

MQL5语言参考手册

MetaTrader 5客户端专用

研究MQL5和 解决任何任务：

- 创建您自己的任何复杂的技术分析指标
- 使用自动交易 - 自动交易系统
工作在不同金融市场
- 开发你自己的分析工具将实际与传统数字工具
相结合
- 撰写解决信息交易系统的任务（交易，监测范围
围广，警报等）

Content

MQL5参考

38

1 语言基础	39
语法	40
注释	41
标识符	42
关键词	43
数据类型	44
整型	45
字符型，短整型，整型和长整型	46
字符常量	49
日期时间型	52
颜色型	53
布尔类型	54
枚举类型	55
真实型	57
字符串数据	62
架构和类	63
动态数组目标	75
类型转换	77
空型和NULL常量	83
目标指针	84
引用，修饰符&和关键字this	85
运行式和表达式	87
表达式	88
算术运算	89
赋值运算	90
关系运算	91
布尔运算	92
逐位运算	94
其他运算	97
优先规则	101
操作符	103
复合操作符	104
表达式操作符	105
返回操作符	106
假设操作符if-else	107
假设操作符?:	108
切换操作符	110
循环操作符while	112
循环操作符for	113
循环操作符do while	114
嵌入操作符	115
继续操作符	116
对象创建操作符new	117
对象删除操作符delete	118
函数	119
调用函数	121
传递函数	122
重载函数	125
外部函数描述	128
输出函数	129
事件处理函数	130
变量	137

局部变量.....	139
形式参数.....	141
静态变量.....	143
全局变量.....	144
输入变量.....	145
外部变量.....	148
变量初始化.....	149
可见范围和变量使用期.....	151
创建和删除对象.....	153
预处理程序.....	155
常量声明 (#define).....	156
程序属性 (#property).....	158
包括文件 (#include).....	161
输入函数 (#import).....	162
面向对象的程序设计.....	163
类型的密封和扩展.....	164
继承算法.....	166
多态性.....	168
重载.....	172
虚拟函数.....	173
2 标准常量，枚举和架构.....	176
图表常量.....	177
图表事件类型.....	178
图表时间表.....	182
图表性能.....	184
定位常量.....	189
图表陈述.....	190
对象常量.....	192
物件类型.....	193
对象属性.....	195
对象定位方法.....	210
图表角落.....	214
对象可见性.....	216
埃利奥特波动水平.....	219
江恩物件.....	220
网页颜色.....	222
Wingdings.....	224
指标常量.....	225
价格常数.....	226
平滑方式.....	229
指标线.....	230
绘画风格.....	232
自定义指标属性.....	236
指标类型.....	240
数据类型标识符.....	242
环境状态.....	243
客户端属性.....	244
运行MQL5程序属性.....	246
交易品种属性.....	248
账户属性.....	256
.....	258
交易常数.....	266
历史数据库属性.....	267
订单属性.....	268
仓位属性.....	271
交易属性.....	273
交易操作类型.....	275
DOM交易订单.....	276

命名常量	277
预定义大量代替值	278
数学常量	280
数值类型常量	281
不能初始化原因代码	283
检测对象指针	285
其他常量	286
数据结构	289
数据类型结构	290
输入参量结构	291
历史数据结构	292
DOM 结构	293
交易请求结构	294
请求结构测试结果	297
交易请求结构结果	298
当前价格结构	301
错误和警告代码	302
交易服务器返回代码	303
编译器警告	305
编译错误	307
运行时间错误	315
输入/输出常量	320
文件开始标签	321
内部文件位置	323
代码页使用	324
对话框	325
3 MQL5 程序	327
程序运行	328
客户端事件	332
调用输入函数	334
运行时间出错	342
运行时间出错	344
4 预定义变量	345
_Digits	346
_Point	347
_LastError	348
_Period	349
_StopFlag	350
_Symbol	351
_UninitReason	352
5 普通函数	353
Alert	354
CheckPointer	355
Comment	357
DebugBreak	358
ExpertRemove	359
GetPointer	361
GetTickCount	365
MessageBox	366
PeriodSeconds	367
Playsound	368
Print	369
PrintFormat	370
ResetLastError	374
SetUserError	375
SendFTP	376
SendMail	377

Sleep	378
TerminalClose	379
TesterStatistics	381
TesterWithdrawal	382
ZeroMemory	383
6 数组函数	384
ArrayBsearch	385
ArrayCopy	387
ArrayFree	388
ArrayGetAsSeries	389
ArrayInitialize	390
ArrayIsDynamic	391
ArrayIsSeries	392
ArrayMaximum	394
ArrayMinimum	395
ArrayRange	396
ArrayResize	397
ArraySetAsSeries	398
ArraySize	401
ArraySort	402
7 函数转换	403
CharToString	404
CharArrayToString	405
ColorToString	406
DoubleToString	407
EnumToString	408
IntegerToString	410
ShortToString	411
ShortArrayToString	412
TimeToString	413
NormalizeDouble	414
StringToCharArray	415
StringToColor	416
StringToDouble	417
StringToInteger	418
StringToShortArray	419
StringToTime	420
StringFormat	421
8 数学函数	422
MathAbs	423
MathArccos	424
MathArcsin	425
MathArctan	426
MathCeil	427
MathCos	428
MathExp	429
MathFloor	430
MathLog	431
MathLog10	432
MathMax	433
MathMin	434
MathMod	435
MathPow	436
MathRand	437
MathRound	438
MathSin	439
MathSqrt	440

MathSrand	441
MathTan	442
MathIsValidNumber	443
9 字符串函数	444
StringAdd	445
StringBufferLen	447
StringCompare	448
StringConcatenate	450
StringFill	451
StringFind	452
StringGetCharacter	453
StringInit	454
StringLen	455
StringReplace	456
StringSetCharacter	457
StringSubstr	459
StringToLower	460
StringToUpper	461
StringTrimLeft	462
StringTrimRight	463
10 日期和时间	464
TimeCurrent	465
TimeTradeServer	466
TimeLocal	467
TimeGMT	468
TimeDaylightSaving	469
TimeGMTOffset	470
TimeToStruct	471
StructToTime	472
11 账户信息	473
AccountInfoDouble	474
AccountInfoInteger	475
AccountInfoString	477
12 检测	478
GetLastError	479
IsStopped	480
UninitializeReason	481
TerminalInfoInteger	482
TerminalInfoString	483
MQL5InfoInteger	484
MQL5InfoString	485
Symbol	486
Period	487
Digits	488
Point	489
13 市场信息	490
SymbolsTotal	491
SymbolName	492
SymbolSelect	493
SymbolSynchronized	494
SymbolInfoDouble	495
SymbolInfoInteger	496
SymbolInfoString	497
SymbolInfoTick	498
SymbolInfoSessionQuote	499
SymbolInfoSessionTrade	500

MarketBookAdd	501
MarketBookRelease	502
MarketBookGet	503
14 时间序列和指标访问	504
数组，缓冲器和时间序列中索引方向	507
组织数据存储	510
SeriesInfoInteger	518
Bars	520
BarsCalculated	522
IndicatorCreate	524
IndicatorRelease	526
CopyBuffer	528
CopyRates	532
CopyTime	535
CopyOpen	537
CopyHigh	539
CopyLow	542
CopyClose	544
CopyTickVolume	546
CopyRealVolume	550
CopySpread	552
15 图表操作	556
ChartApplyTemplate	558
ChartSaveTemplate	561
ChartWindowFind	566
ChartOpen	568
ChartFirst	569
ChartNext	570
ChartClose	571
ChartSymbol	572
ChartPeriod	573
ChartRedraw	574
ChartSetDouble	575
ChartSetInteger	576
ChartSetString	577
ChartGetDouble	578
ChartGetInteger	579
ChartGetString	580
ChartNavigate	582
ChartID	585
ChartIndicatorAdd	586
ChartIndicatorDelete	587
ChartIndicatorName	589
ChartIndicatorsTotal	591
ChartWindowOnDropped	592
ChartPriceOnDropped	593
ChartTimeOnDropped	594
ChartXOnDropped	595
ChartYOnDropped	596
ChartSetSymbolPeriod	597
ChartScreenShot	598
16 交易函数	599
OrderCalcMargin	601
OrderCalcProfit	602
OrderCheck	603
OrderSend	604
PositionsTotal	608

PositionGetSymbol	609
PositionSelect	610
PositionGetDouble	611
PositionGetInteger	612
PositionGetString	613
OrdersTotal	614
OrderGetTicket	615
OrderSelect	617
OrderGetDouble	618
OrderGetInteger	619
OrderGetString	620
HistorySelect	621
HistorySelectByPosition	623
HistoryOrderSelect	624
HistoryOrdersTotal	625
HistoryOrderGetTicket	626
HistoryOrderGetDouble	628
HistoryOrderGetInteger	629
HistoryOrderGetString	630
HistoryDealSelect	631
HistoryDealsTotal	632
HistoryDealGetTicket	633
HistoryDealGetDouble	636
HistoryDealGetInteger	637
HistoryDealGetString	638
17 程序端全局变量	639
GlobalVariableCheck	640
GlobalVariableTime	641
GlobalVariableDel	642
GlobalVariableGet	643
GlobalVariableName	644
GlobalVariableSet	645
GlobalVariablesFlush	646
GlobalVariableTemp	647
GlobalVariableSetOnCondition	648
GlobalVariablesDeleteAll	649
GlobalVariablesTotal	650
18 文件函数	651
FileFindFirst	653
FileFindNext	655
FileFindClose	656
FileIsExist	657
FileOpen	658
FileClose	660
FileCopy	661
FileDelete	662
FileMove	663
FileFlush	664
FileIsEnding	665
FileIsLineEnding	666
FileReadArray	667
FileReadBool	668
FileReadDatetime	669
FileReadDouble	670
FileReadFloat	671
FileReadInteger	672
FileReadLong	673
FileReadNumber	674

FileReadString	675
FileSeek	676
FileReadStruct	677
FileSize	678
FileTell	679
FileWrite	680
FileWriteArray	681
FileWriteDouble	682
FileWriteFloat	683
FileWriteInteger	684
FileWriteLong	685
FileWriteString	686
FileWriteStruct	687
FolderCreate	688
FolderDelete	689
FolderClean	690
19 自定义指标	691
	694
DRAW_NONE	709
DRAW_LINE	712
DRAW_SECTION	716
DRAW_HISTOGRAM	721
DRAW_HISTOGRAM2	725
DRAW_ARROW	730
DRAW_ZIGZAG	735
DRAW_FILLING	740
DRAW_BARS	745
DRAW_CANDLES	751
DRAW_COLOR_LINE	757
DRAW_COLOR_SECTION	763
DRAW_COLOR_HISTOGRAM	769
DRAW_COLOR_HISTOGRAM2	775
DRAW_COLOR_ARROW	780
DRAW_COLOR_ZIGZAG	786
DRAW_COLOR_BARS	792
DRAW_COLOR_CANDLES	799
指标属性和函数间的连接	806
SetIndexBuffer	808
IndicatorSetDouble	811
IndicatorSetInteger	812
IndicatorSetString	813
PlotIndexSetDouble	814
PlotIndexSetInteger	815
PlotIndexSetString	819
PlotIndexGetInteger	820
20 物件函数	823
ObjectCreate	824
ObjectName	826
ObjectDelete	827
ObjectsDeleteAll	828
ObjectFind	829
ObjectGetTimeByValue	830
ObjectGetValueByTime	831
ObjectMove	832
ObjectsTotal	833
ObjectSetDouble	834
ObjectSetInteger	837
ObjectSetString	838

	ObjectGetDouble	839
	ObjectGetInteger	840
	ObjectGetString	841
21	技术指标	843
	iAC	845
	iAD	850
	iADX	855
	iADXWilder	860
	iAlligator	866
	iAMA	873
	iAO	878
	iATR	883
	iBearsPower	888
	iBands	893
	iBullsPower	899
	iCCI	904
	iChaikin	909
	iCustom	914
	iDEMA	917
	iDeMarker	922
	iEnvelopes	927
	iForce	933
	iFractals	938
	iFrAMA	943
	iGator	948
	iIchimoku	955
	iBWMFI	962
	iMomentum	967
	iMFI	972
	iMA	977
	iOsMA	982
	iMACD	987
	iOBV	993
	iSAR	998
	iRSI	1003
	iRVI	1008
	iStdDev	1013
	iStochastic	1018
	iTEMA	1024
	iTriX	1029
	iWPR	1034
	iVidya	1039
	iVolumes	1044
22	工作事件	1049
	EventSetTimer	1050
	EventKillTimer	1051
	EventChartCustom	1052
23	标准程序库	1057
	Basic Class CObject	1058
	Prev	1059
	Prev	1060
	Next	1061
	Next	1062
	Compare	1063
	Save	1065
	Load	1067
	Type	1069

Classes of data	1070
CArray	1071
Step	1073
Step	1074
Total	1075
Available	1076
Max	1077
IsSorted	1078
SortMode	1079
Clear	1080
Sort	1081
Save	1082
Load	1083
CArrayChar	1084
Reserve	1086
Resize	1087
Shutdown	1088
Add	1089
AddArray	1090
AddArray	1091
Insert	1093
InsertArray	1094
InsertArray	1095
AssignArray	1097
AssignArray	1098
Update	1100
Shift	1101
Delete	1102
DeleteRange	1103
At	1104
CompareArray	1106
CompareArray	1107
InsertSort	1108
Search	1109
SearchGreat	1110
SearchLess	1111
SearchGreatOrEqual	1112
SearchLessOrEqual	1113
SearchFirst	1114
SearchLast	1115
Save	1116
Load	1117
Type	1119
CArrayShort	1120
Reserve	1122
Resize	1123
Shutdown	1124
Add	1125
AddArray	1126
AddArray	1127
Insert	1129
InsertArray	1130
InsertArray	1131
AssignArray	1133
AssignArray	1134
Update	1136
Shift	1137
Delete	1138

DeleteRange.....	1139
At	1140
CompareArray.....	1142
CompareArray.....	1143
InsertSort	1144
Search	1145
SearchGreat	1146
SearchLess.....	1147
SearchGreatOrEqual.....	1148
SearchLessOrEqual	1149
SearchFirst.....	1150
SearchLast.....	1151
Save	1152
Load	1154
Type	1156
CArrayInt.....	1157
Reserve	1159
Resize	1160
Shutdown	1161
Add	1162
AddArray	1163
AddArray	1164
Insert	1166
InsertArray	1167
InsertArray	1168
AssignArray.....	1170
AssignArray.....	1171
Update	1173
Shift	1174
Delete	1175
DeleteRange.....	1176
At	1177
CompareArray.....	1179
CompareArray.....	1180
InsertSort	1181
Search	1182
SearchGreat	1183
SearchLess.....	1184
SearchGreatOrEqual.....	1185
SearchLessOrEqual	1186
SearchFirst.....	1187
SearchLast.....	1188
Save	1189
Load	1191
Type	1193
CArrayLong.....	1194
Reserve	1196
Resize	1197
Shutdown	1198
Add	1199
AddArray	1200
AddArray	1201
Insert	1203
InsertArray	1204
InsertArray	1205
AssignArray.....	1207
AssignArray.....	1208
Update	1210

Shift	1211
Delete	1212
DeleteRange	1213
At	1214
CompareArray	1216
CompareArray	1217
InsertSort	1218
Search	1219
SearchGreat	1220
SearchLess	1221
SearchGreatOrEqual	1222
SearchLessOrEqual	1223
SearchFirst	1224
SearchLast	1225
Save	1226
Load	1228
Type	1230
CArrayFloat	1231
Delta	1233
Reserve	1234
Resize	1235
Shutdown	1236
Add	1237
AddArray	1238
AddArray	1239
Insert	1241
InsertArray	1242
InsertArray	1243
AssignArray	1245
AssignArray	1246
Update	1248
Shift	1249
Delete	1250
DeleteRange	1251
At	1252
CompareArray	1254
CompareArray	1255
InsertSort	1256
Search	1257
SearchGreat	1258
SearchLess	1259
SearchGreatOrEqual	1260
SearchLessOrEqual	1261
SearchFirst	1262
SearchLast	1263
Save	1264
Load	1266
Type	1268
CArrayDouble	1269
Delta	1271
Reserve	1272
Resize	1273
Shutdown	1274
Add	1275
AddArray	1276
AddArray	1277
Insert	1279
InsertArray	1280

InsertArray	1281
AssignArray.....	1283
AssignArray.....	1284
Update	1286
Shift	1287
Delete	1288
DeleteRange.....	1289
At	1290
CompareArray.....	1292
CompareArray.....	1293
InsertSort	1294
Search	1295
SearchGreat	1296
SearchLess.....	1297
SearchGreatOrEqual.....	1298
SearchLessOrEqual	1299
SearchFirst.....	1300
SearchLast.....	1301
Save	1302
Load	1304
Type	1306
CArrayString.....	1307
Reserve	1309
Resize	1310
Shutdown	1311
Add	1312
AddArray	1313
AddArray	1314
Insert	1316
InsertArray	1317
InsertArray	1318
AssignArray.....	1320
AssignArray.....	1321
Update	1323
Shift	1324
Delete	1325
DeleteRange.....	1326
At	1327
CompareArray.....	1329
CompareArray.....	1330
InsertSort	1331
Search	1332
SearchGreat	1333
SearchLess.....	1334
SearchGreatOrEqual.....	1335
SearchLessOrEqual	1336
SearchFirst.....	1337
SearchLast.....	1338
Save	1339
Load	1341
Type	1343
CArrayObj	1344
FreeMode	1349
FreeMode	1350
Reserve	1352
Resize	1353
Clear	1355
Shutdown	1356

CreateElement	1357
Add	1359
AddArray	1360
Insert	1363
InsertArray	1365
AssignArray	1367
Update	1369
Shift	1370
Detach	1371
Delete	1373
DeleteRange	1374
At	1375
CompareArray	1376
InsertSort	1377
Search	1378
SearchGreat	1379
SearchLess	1380
SearchGreatOrEqual	1381
SearchLessOrEqual	1382
SearchFirst	1383
SearchLast	1384
Save	1385
Load	1386
Type	1388
CList	1389
FreeMode	1391
FreeMode	1392
Total	1393
IsSorted	1394
SortMode	1395
CreateElement	1396
Add	1397
Insert	1398
DetachCurrent	1400
DeleteCurrent	1401
Delete	1402
Clear	1403
IndexOf	1404
GetNodeAt Index	1405
GetFirstNode	1406
GetPrevNode	1407
GetCurrentNode	1408
GetNextNode	1409
GetLastNode	1410
Sort	1411
MoveToIndex	1412
Exchange	1413
CompareList	1414
Search	1415
Save	1416
Load	1418
Type	1420
CTreeNode	1421
Owner	1426
Left	1427
Right	1428
Balance	1429
BalanceL	1430

BalanceR	1431
CreateSample.....	1432
RefreshBalance.....	1433
GetNext	1434
SaveNode	1435
LoadNode	1436
Type	1437
CTree	1438
Root	1443
CreateElement.....	1444
Insert	1445
Detach	1446
Delete	1447
Clear	1448
Find	1449
Save	1450
Load	1451
Type	1452
Classes for Graphic Objects	1453
CChartObject.....	1454
ChartId	1457
Window	1458
Name	1459
NumPoints	1460
Attach	1461
SetPoint	1462
Delete	1463
Detach	1464
ShiftObject.....	1465
ShiftPoint	1466
Time	1467
Price	1469
Color	1471
Style	1472
Width	1473
Background.....	1474
Selected	1475
Selectable	1476
Description.....	1477
Tooltip	1478
Timeframes.....	1479
CreateTime.....	1480
LevelsCount.....	1481
LevelColor	1482
LevelStyle	1484
LevelWidth.....	1486
LevelValue	1488
LevelDescription.....	1490
GetInteger.....	1492
SetInteger	1494
GetDouble	1496
SetDouble	1498
GetString	1500
SetString	1502
Save	1504
Load	1505
Type	1506
Objects Lines.....	1507

CChartObjectVLine.....	1508
Create	1509
Type	1510
CChartObjectHLine.....	1511
Create	1512
Type	1513
CChartObjectTrend.....	1514
Create	1515
RayLeft	1516
RayRight	1517
Save	1518
Load	1519
Type	1520
CChartObjectTrendByAngle.....	1521
Create	1522
Angle	1523
Type	1524
CChartObjectCycles.....	1525
Create	1526
Type	1527
Objects Channels.....	1528
CChartObjectChannel.....	1529
Create	1530
Type	1531
CChartObjectRegression.....	1532
Create	1533
Type	1534
CChartObjectStdDevChannel.....	1535
Create	1536
Deviations	1537
Save	1538
Load	1539
Type	1540
CChartObjectPitchfork.....	1541
Create	1542
Type	1543
Gann Tools.....	1544
CChartObjectGannLine.....	1545
Create	1546
PipsPerBar	1547
Save	1548
Load	1549
Type	1550
CChartObjectGannFan.....	1551
Create	1552
PipsPerBar	1553
Downtrend.....	1554
Save	1555
Load	1556
Type	1557
CChartObjectGannGrid.....	1558
Create	1559
PipsPerBar	1560
Downtrend.....	1561
Save	1562
Load	1563
Type	1564
Fibonacci Tools.....	1565

CChartObjectFibo.....	1566
Create	1567
Type	1568
CChartObjectFiboTimes.....	1569
Create	1570
Type	1571
CChartObjectFiboFan	1572
Create	1573
Type	1574
CChartObjectFiboArc.....	1575
Create	1576
Scale	1577
Ellipse	1578
Save	1579
Load	1580
Type	1581
CChartObjectFiboChannel.....	1582
Create	1583
Type	1584
CChartObjectFiboExpansion.....	1585
Create	1586
Type	1587
Elliott Tools.....	1588
CChartObjectElliottWave3.....	1589
Create	1590
Degree	1591
Lines	1592
Save	1593
Load	1594
Type	1595
CChartObjectElliottWave5.....	1596
Create	1597
Type	1599
Objects Shapes.....	1600
CChartObjectRectangle.....	1601
Create	1602
Type	1603
CChartObjectTriangle	1604
Create	1605
Type	1606
CChartObjectEllipse.....	1607
Create	1608
Type	1609
Objects Arrows.....	1610
CChartObjectArrow.....	1611
Create	1612
ArrowCode.....	1614
Anchor	1616
Save	1618
Load	1619
Type	1620
Arrows with fixed code.....	1621
Create	1623
ArrowCode.....	1625
Type	1626
Objects Controls.....	1627
CChartObjectText	1628
Create	1629

Angle	1630
Font	1631
FontSize	1632
Anchor	1633
Save	1634
Load	1635
Type	1636
CChartObjectLabel	1637
Create	1638
X_Distance	1639
Y_Distance	1640
X_Size	1641
Y_Size	1642
Corner	1643
Time	1644
Price	1645
Save	1646
Load	1647
Type	1648
CChartObjectEdit	1649
Create	1650
X_Size	1651
Y_Size	1652
BackColor	1653
Angle	1654
Save	1655
Load	1656
Type	1657
CChartObjectButton	1658
State	1659
Save	1660
Load	1661
Type	1662
CChartObjectSubChart	1663
Create	1665
X_Distance	1666
Y_Distance	1667
Corner	1668
X_Size	1669
Y_Size	1670
Symbol	1671
Period	1672
Scale	1673
DateScale	1674
PriceScale	1675
Time	1676
Price	1677
Save	1678
Load	1679
Type	1680
CChartObjectBitmap	1681
Create	1682
BmpFile	1683
X_Offset	1684
Y_Offset	1685
Save	1686
Load	1687
Type	1688

CChartObjectBmpLabel.....	1689
Create	1691
X_Distance.....	1692
Y_Distance.....	1693
X_Offset	1694
Y_Offset	1695
Corner	1696
X_Size	1697
Y_Size	1698
BmpFileOn	1699
BmpFileOff.....	1700
State	1701
Time	1702
Price	1703
Save	1704
Load	1705
Type	1706
CChartObjectRectLabel.....	1707
Create	1708
X_Size	1709
Y_Size	1710
BackColor	1711
Angle	1712
BorderType.....	1713
Save	1714
Load	1715
Type	1716
Class for working with chart	1717
ChartID.....	1722
Mode	1723
Foreground.....	1724
Shift	1725
ShiftSize.....	1726
AutoScroll.....	1727
Scale	1728
ScaleFix	1729
ScaleFix_11.....	1730
FixedMax.....	1731
FixedMin.....	1732
PointsPerBar.....	1733
ScalePPB.....	1734
ShowOHLC.....	1735
ShowLineBid.....	1736
ShowLineAsk.....	1737
ShowLastLine.....	1738
ShowPeriodSep.....	1739
ShowGrid.....	1740
ShowVolumes.....	1741
ShowObjectDescr.....	1742
ShowDateScale.....	1743
ShowPriceScale.....	1744
ColorBackground.....	1745
ColorForeground.....	1746
ColorGrid.....	1747
ColorBarUp.....	1748
ColorBarDown.....	1749
ColorCandleBull.....	1750
ColorCandleBear.....	1751

ColorChartLine	1752
ColorVolumes	1753
ColorLineBid	1754
ColorLineAsk	1755
ColorLineLast	1756
ColorStopLevels	1757
VisibleBars	1758
WindowsTotal	1759
WindowIsVisible	1760
WindowHandle	1761
FirstVisibleBar	1762
WidthInBars	1763
WidthInPixels	1764
HeightInPixels	1765
PriceMin	1766
PriceMax	1767
Attach	1768
FirstChart	1769
NextChart	1770
Open	1771
Detach	1772
Close	1773
BringToTop	1774
EventObjectCreate	1775
EventObjectDelete	1776
IndicatorAdd	1777
IndicatorDelete	1778
IndicatorsTotal	1779
IndicatorName	1780
Navigate	1781
Symbol	1782
Period	1783
Redraw	1784
GetInteger	1785
SetInteger	1786
GetDouble	1787
SetDouble	1788
GetString	1789
SetString	1790
SetSymbolPeriod	1791
ApplyTemplate	1792
ScreenShot	1793
WindowOnDropped	1794
PriceOnDropped	1795
TimeOnDropped	1796
XOnDropped	1797
YOnDropped	1798
Save	1799
Load	1800
Type	1801
Classes for file operations	1802
CFile	1803
Handle	1805
Filename	1806
Flags	1807
SetUnicode	1808
SetCommon	1809
Open	1810

Close	1811
Delete	1812
IsExist	1813
Copy	1814
Move	1815
Size	1816
Tell	1817
Seek	1818
Flush	1819
IsEnding	1820
IsLineEnding	1821
FolderCreate	1822
FolderDelete	1823
FolderClean	1824
FileFindFirst	1825
FileFindNext	1826
FileFindClose	1827
CFileBin	1828
Open	1830
WriteChar	1831
WriteShort	1832
WriteInteger	1833
WriteLong	1834
WriteFloat	1835
WriteDouble	1836
WriteString	1837
WriteCharArray	1838
WriteShortArray	1839
WriteIntegerArray	1840
WriteLongArray	1841
WriteFloatArray	1842
WriteDoubleArray	1843
WriteObject	1844
ReadChar	1845
ReadShort	1846
ReadInteger	1847
ReadLong	1848
ReadFloat	1849
ReadDouble	1850
ReadString	1851
ReadCharArray	1852
ReadShortArray	1853
ReadIntegerArray	1854
ReadLongArray	1855
ReadFloatArray	1856
ReadDoubleArray	1857
ReadObject	1858
CFileTxt	1859
Open	1860
WriteString	1861
ReadString	1862
Class for String operations	1863
CString	1864
Str	1866
Len	1867
Copy	1868
Fill	1869
Assign	1870

Append	1871
Insert	1872
Compare	1873
CompareNoCase	1874
Left	1875
Right	1876
Mid	1877
Trim	1878
TrimLeft	1879
TrimRight	1880
Clear	1881
ToUpper	1882
ToLower	1883
Reverse	1884
Find	1885
FindRev	1886
Remove	1887
Replace	1888
Classes for working with Indicators	1889
Base classes	1890
CSpreadBuffer	1891
Size	1892
SetSymbolPeriod	1893
At	1894
Refresh	1895
RefreshCurrent	1896
CTimeBuffer	1897
Size	1898
SetSymbolPeriod	1899
At	1900
Refresh	1901
RefreshCurrent	1902
CTickVolumeBuffer	1903
Size	1904
SetSymbolPeriod	1905
At	1906
Refresh	1907
RefreshCurrent	1908
CRealVolumeBuffer	1909
Size	1910
SetSymbolPeriod	1911
At	1912
Refresh	1913
RefreshCurrent	1914
CDoubleBuffer	1915
Size	1916
SetSymbolPeriod	1917
At	1918
Refresh	1919
RefreshCurrent	1920
Minimum	1921
Maximum	1922
COpenBuffer	1923
Refresh	1924
RefreshCurrent	1925
CHighBuffer	1926
Refresh	1927
RefreshCurrent	1928

CLowBuffer	1929
Refresh	1930
RefreshCurrent	1931
CCloseBuffer	1932
Refresh	1933
RefreshCurrent	1934
CIndicatorBuffer	1935
Offset	1936
Name	1937
At	1938
Refresh	1939
RefreshCurrent	1940
CSeries	1941
Name	1942
BuffersTotal	1943
Timeframe	1944
Symbol	1945
Period	1946
RefreshCurrent	1947
BufferResize	1948
Refresh	1949
PeriodDescription	1950
CPriceSeries	1951
BufferResize	1952
GetData	1953
Refresh	1954
MinIndex	1955
MinValue	1956
MaxIndex	1957
MaxValue	1958
CIndicator	1959
Handle	1962
Status	1963
FullRelease	1964
Create	1965
BufferResize	1966
GetData	1967
Refresh	1970
Minimum	1971
MinValue	1972
Maximum	1973
MaxValue	1974
MethodDescription	1975
PriceDescription	1976
VolumeDescription	1977
AddToChart	1978
DeleteFromChart	1979
CIndicators	1980
Create	1981
Refresh	1982
Timeseries classes	1983
CiSpread	1984
Create	1985
BufferResize	1986
GetData	1987
Refresh	1989
CiTime	1990
Create	1991

BufferResize.....	1992
GetData	1993
Refresh	1995
CiTickVolume.....	1996
Create	1997
BufferResize.....	1998
GetData	1999
Refresh	2001
CiRealVolume	2002
Create	2003
BufferResize.....	2004
GetData	2005
Refresh	2007
CiOpen	2008
Create	2009
GetData	2010
CiHigh	2012
Create	2013
GetData	2014
CiLow	2016
Create	2017
GetData	2018
CiClose	2020
Create	2021
GetData	2022
Trend Indicators.....	2024
CiADX	2025
MaPeriod	2026
Create	2027
Main	2028
Plus	2029
Minus	2030
Type	2031
CiADXWilder.....	2032
MaPeriod	2033
Create	2034
Main	2035
Plus	2036
Minus	2037
Type	2038
CiBands	2039
MaPeriod	2040
MaShift	2041
Deviation	2042
Applied	2043
Create	2044
Base	2045
Upper	2046
Lower	2047
Type	2048
CiEnvelopes.....	2049
MaPeriod	2050
MaShift	2051
MaMethod	2052
Deviation	2053
Applied	2054
Create	2055
Upper	2056

Lower	2057
Type	2058
Cilchimoku	2059
TenkanSenPeriod	2060
KijunSenPeriod	2061
SenkouSpanBPeriod	2062
Create	2063
TenkanSen	2064
KijunSen	2065
SenkouSpanA	2066
SenkouSpanB	2067
ChinkouSpan	2068
Type	2069
CIMA	2070
MaPeriod	2071
MaShift	2072
MaMethod	2073
Applied	2074
Create	2075
Main	2076
Type	2077
CISAR	2078
SarStep	2079
Maximum	2080
Create	2081
Main	2082
Type	2083
CiStdDev	2084
MaPeriod	2085
MaShift	2086
MaMethod	2087
Applied	2088
Create	2089
Main	2090
Type	2091
CIDEMA	2092
MaPeriod	2093
IndShift	2094
Applied	2095
Create	2096
Main	2097
Type	2098
CITEMA	2099
MaPeriod	2100
IndShift	2101
Applied	2102
Create	2103
Main	2104
Type	2105
CiFrAMA	2106
MaPeriod	2107
IndShift	2108
Applied	2109
Create	2110
Main	2111
Type	2112
CiAMA	2113
MaPeriod	2114

FastEmaPeriod	2115
SlowEmaPeriod	2116
IndShift	2117
Applied	2118
Create	2119
Main	2120
Type	2121
CiMDyA	2122
CmoPeriod	2123
EmaPeriod	2124
IndShift	2125
Applied	2126
Create	2127
Main	2128
Type	2129
Oscillators	2130
CiATR	2131
MaPeriod	2132
Create	2133
Main	2134
Type	2135
CiBearsPower	2136
MaPeriod	2137
Create	2138
Main	2139
Type	2140
CiBullsPower	2141
MaPeriod	2142
Create	2143
Main	2144
Type	2145
CiCCI	2146
MaPeriod	2147
Applied	2148
Create	2149
Main	2150
Type	2151
CiChaikin	2152
FastMaPeriod	2153
SlowMaPeriod	2154
MaMethod	2155
Applied	2156
Create	2157
Main	2158
Type	2159
CiDeMarker	2160
MaPeriod	2161
Create	2162
Main	2163
Type	2164
CiForce	2165
MaPeriod	2166
MaMethod	2167
Applied	2168
Create	2169
Main	2170
Type	2171
CiMACD	2172

FastEmaPeriod	2173
SlowEmaPeriod	2174
SignalPeriod	2175
Applied	2176
Create	2177
Main	2178
Signal	2179
Type	2180
CiMomentum	2181
MaPeriod	2182
Applied	2183
Create	2184
Main	2185
Type	2186
CiOsMA	2187
FastEmaPeriod	2188
SlowEmaPeriod	2189
SignalPeriod	2190
Applied	2191
Create	2192
Main	2193
Type	2194
CiRSI	2195
MaPeriod	2196
Applied	2197
Create	2198
Main	2199
Type	2200
CiRM	2201
MaPeriod	2202
Create	2203
Main	2204
Signal	2205
Type	2206
CiStochastic	2207
Kperiod	2208
Dperiod	2209
Slowing	2210
MaMethod	2211
PriceField	2212
Create	2213
Main	2214
Signal	2215
Type	2216
CiTriX	2217
MaPeriod	2218
Applied	2219
Create	2220
Main	2221
Type	2222
CiWPR	2223
CalcPeriod	2224
Create	2225
Main	2226
Type	2227
Volume Indicators	2228
CiAD	2229
Applied	2230

Create	2231
Main	2232
Type	2233
CiMFI	2234
MaPeriod	2235
Applied	2236
Create	2237
Main	2238
Type	2239
CiOBV	2240
Applied	2241
Create	2242
Main	2243
Type	2244
CiVolumes	2245
Applied	2246
Create	2247
Main	2248
Type	2249
Bill Williams Indicators	2250
CiAC	2251
Create	2252
Main	2253
Type	2254
CiAlligator	2255
JawPeriod	2256
JawShift	2257
TeethPeriod	2258
TeethShift	2259
LipsPeriod	2260
LipsShift	2261
MaMethod	2262
Applied	2263
Create	2264
Jaw	2265
Teeth	2266
Lips	2267
Type	2268
CiAO	2269
Create	2270
Main	2271
Type	2272
CiFractals	2273
Create	2274
Upper	2275
Lower	2276
Type	2277
CiGator	2278
JawPeriod	2279
JawShift	2280
TeethPeriod	2281
TeethShift	2282
LipsPeriod	2283
LipsShift	2284
MaMethod	2285
Applied	2286
Create	2287
Upper	2288

Lower	2289
Type	2290
CiBWMFI	2291
Applied	2292
Create	2293
Main	2294
Type	2295
Custom indicators.....	2296
NumBuffers.....	2297
NumParams.....	2298
ParamType.....	2299
ParamLong.....	2300
ParamDouble.....	2301
ParamString.....	2302
Type	2303
Trade Classes	2304
CAccountInfo.....	2305
Login	2307
TradeMode.....	2308
TradeModeDescription.....	2309
Leverage	2310
MarginMode.....	2311
MarginModeDescription.....	2312
TradeAllowed.....	2313
TradeExpert	2314
Limit Orders.....	2315
Balance	2316
Credit	2317
Profit	2318
Equity	2319
Margin	2320
FreeMargin.....	2321
MarginLevel.....	2322
MarginCall	2323
MarginStopOut	2324
Name	2325
Server	2326
Currency	2327
Company	2328
InfoInteger	2329
InfoDouble.....	2330
InfoString	2331
OrderProfitCheck.....	2332
MarginCheck.....	2333
FreeMarginCheck.....	2334
MaxLotCheck	2335
CSymbolInfo.....	2336
Refresh	2339
RefreshRates.....	2340
Name	2341
Select	2342
IsSynchronized.....	2343
Volume	2344
VolumeHigh.....	2345
VolumeLow.....	2346
VolumeBid	2347
VolumeAsk	2348
Time	2349

Spread	2350
SpreadFloat	2351
TickBookDepth	2352
StopsLevel	2353
FreezeLevel	2354
Bid	2355
BidHigh	2356
BidLow	2357
Ask	2358
AskHigh	2359
AskLow	2360
Last	2361
LastHigh	2362
LastLow	2363
TradeCalcMode	2364
TradeCalcModeDescription	2365
TradeMode	2366
TradeModeDescription	2367
TradeExecution	2368
TradeExecutionDescription	2369
SwapMode	2370
SwapModeDescription	2371
SwapRollover3days	2372
SwapRollover3daysDescription	2373
MarginInitial	2374
MarginMaintenance	2375
MarginLong	2376
MarginShort	2377
MarginLimit	2378
MarginStop	2379
MarginStopLimit	2380
TradeTimeFlags	2381
TradeFillFlags	2382
Digits	2383
Point	2384
TickValue	2385
TickValueProfit	2386
TickValueLoss	2387
TickSize	2388
ContractSize	2389
LotsMin	2390
LotsMax	2391
LotsStep	2392
LotsLimit	2393
SwapLong	2394
SwapShort	2395
CurrencyBase	2396
CurrencyProfit	2397
CurrencyMargin	2398
Bank	2399
Description	2400
Path	2401
InfoInteger	2402
InfoDouble	2403
InfoString	2404
NormalizePrice	2405
COrderInfo	2406
Ticket	2408

TimeSetup	2409
OrderType	2410
TypeDescription	2411
State	2412
StateDescription	2413
TimeExpiration	2414
TimeDone	2415
TypeFilling	2416
TypeFillingDescription	2417
TypeTime	2418
TypeTimeDescription	2419
Magic	2420
PositionId	2421
VolumeInitial	2422
VolumeCurrent	2423
PriceOpen	2424
StopLoss	2425
TakeProfit	2426
PriceCurrent	2427
PriceStopLimit	2428
Symbol	2429
Comment	2430
InfoInteger	2431
InfoDouble	2432
InfoString	2433
StoreState	2434
CheckState	2435
Select	2436
SelectByIndex	2437
CHistoryOrderInfo	2438
TimeSetup	2440
OrderType	2441
TypeDescription	2442
State	2443
StateDescription	2444
TimeExpiration	2445
TimeDone	2446
TypeFilling	2447
TypeFillingDescription	2448
TypeTime	2449
TypeTimeDescription	2450
Magic	2451
PositionId	2452
VolumeInitial	2453
VolumeCurrent	2454
PriceOpen	2455
StopLoss	2456
TakeProfit	2457
PriceCurrent	2458
PriceStopLimit	2459
Symbol	2460
Comment	2461
InfoInteger	2462
InfoDouble	2463
InfoString	2464
Ticket	2465
SelectByIndex	2466
CPositionInfo	2467

Time	2469
PositionType	2470
TypeDescription	2471
Magic	2472
Identifier	2473
Volume	2474
PriceOpen	2475
StopLoss	2476
TakeProfit	2477
PriceCurrent	2478
Commission	2479
Swap	2480
Profit	2481
Symbol	2482
InfoInteger	2483
InfoDouble	2484
InfoString	2485
Select	2486
SelectByIndex	2487
StoreState	2488
CheckState	2489
CDealInfo	2490
Order	2492
Time	2493
DealType	2494
TypeDescription	2495
Entry	2496
EntryDescription	2497
Magic	2498
PositionId	2499
Volume	2500
Price	2501
Commission	2502
Swap	2503
Profit	2504
Symbol	2505
Comment	2506
InfoInteger	2507
InfoDouble	2508
InfoString	2509
Ticket	2510
SelectByIndex	2511
CTrade	2512
LogLevel	2515
SetExpertMagicNumber	2516
SetDeviationInPoints	2517
SetTypeFilling	2518
OrderOpen	2519
OrderModify	2521
OrderDelete	2522
PositionOpen	2523
PositionModify	2524
PositionClose	2525
Buy	2526
Sell	2527
BuyLimit	2528
BuyStop	2529
SellLimit	2530

SellStop	2531
Request	2532
RequestAction	2533
RequestActionDescription	2534
RequestMagic	2535
RequestOrder	2536
RequestSymbol	2537
RequestVolume	2538
RequestPrice	2539
RequestStopLimit	2540
RequestSL	2541
RequestTP	2542
RequestDeviation	2543
RequestType	2544
RequestTypeDescription	2545
RequestTypeFilling	2546
RequestTypeFillingDescription	2547
RequestTypeTime	2548
RequestTypeTimeDescription	2549
RequestExpiration	2550
RequestComment	2551
Result	2552
ResultRetcode	2553
ResultRetcodeDescription	2554
ResultDeal	2555
ResultOrder	2556
ResultVolume	2557
ResultPrice	2558
ResultBid	2559
ResultAsk	2560
ResultComment	2561
CheckResult	2562
CheckResultRetcode	2563
CheckResultRetcodeDescription	2564
CheckResultBalance	2565
CheckResultEquity	2566
CheckResultProfit	2567
CheckResultMargin	2568
CheckResultMarginFree	2569
CheckResultMarginLevel	2570
CheckResultComment	2571
PrintRequest	2572
PrintResult	2573
FormatRequest	2574
FormatRequestResult	2575
Trading Strategy Classes	2576
Base classes for Expert Advisors	2579
CExpert	2580
Init	2584
InitSignal	2585
InitTrailing	2586
InitMoney	2587
Deinit	2588
MaxOrders	2589
OnTick	2590
OnTrade	2591
OnTimer	2592
InitParameters	2593

InitIndicators.....	2594
InitTrade	2595
DeinitTrade	2596
DeinitSignal	2597
DeinitTrailing.....	2598
DeinitMoney	2599
DeinitIndicators	2600
Refresh	2601
Processing	2602
CheckOpen	2604
CheckOpenLong.....	2605
CheckOpenShort.....	2606
OpenLong	2607
OpenShort	2608
CheckClose	2609
CheckCloseLong.....	2611
CheckCloseShort.....	2612
CloseAll	2613
Close	2614
CloseLong	2615
CloseShort	2616
CheckReverse.....	2617
CheckReverseLong.....	2618
CheckReverseShort	2619
ReverseLong	2620
ReverseShort.....	2621
CheckTrailingStop.....	2622
CheckTrailingStopLong.....	2623
CheckTrailingStopShort	2624
TrailingStopLong	2625
TrailingStopShort.....	2626
CheckTrailingOrderLong.....	2627
CheckTrailingOrderShort.....	2628
TrailingOrderLong.....	2629
TrailingOrderShort	2630
CheckDeleteOrderLong	2631
CheckDeleteOrderShort	2632
DeleteOrders.....	2633
DeleteOrder	2634
DeleteOrderLong.....	2635
DeleteOrderShort.....	2636
LotOpenLong.....	2637
LotOpenShort	2638
LotReverse	2639
PrepareHistoryDate	2640
HistoryPoint.....	2641
CheckTradeState	2642
WaitEvent	2643
NoWaitEvent	2644
IsWaitingPositionOpened.....	2645
IsWaitingPositionVolumeChanged	2646
IsWaitingPositionModified.....	2647
IsWaitingPositionClosed.....	2648
IsWaitingPositionStopTake	2649
IsWaitingOrderPlaced.....	2650
IsWaitingOrderModified.....	2651
IsWaitingOrderDeleted.....	2652
IsWaitingOrderTriggered	2653

TradeEvent PositionStopTake	2654
TradeEvent OrderTriggered	2655
TradeEvent PositionOpened	2656
TradeEvent PositionVolumeChanged	2657
TradeEvent PositionModified	2658
TradeEvent PositionClosed	2659
TradeEvent OrderPlaced	2660
TradeEvent OrderModified	2661
TradeEvent OrderDeleted	2662
TradeEvent Not Identified	2663
TimeframeAdd	2664
TimeframesFlags	2665
CExpert Signal	2666
Init	2667
Init Indicators	2668
ValidationSettings	2669
CheckOpenLong	2670
CheckCloseLong	2671
CheckOpenShort	2672
CheckCloseShort	2673
CheckReverseLong	2674
CheckReverseShort	2675
CheckTrailingOrderLong	2676
CheckTrailingOrderShort	2677
CExpert Trailing	2678
Init	2679
Init Indicators	2680
ValidationSettings	2681
CheckTrailingStopLong	2682
CheckTrailingStopShort	2683
CExpert Money	2684
Percent	2685
Init	2686
Init Indicators	2687
ValidationSettings	2688
CheckOpenLong	2689
CheckOpenShort	2690
CheckReverse	2691
CheckClose	2692
Modules of Trade Signals	2693
Signals of the Indicator Accelerator Oscillator	2696
Signals of the Indicator Adaptive Moving Average	2699
Signals of the Indicator Awesome Oscillator	2703
Signals of the Oscillator Bears Power	2707
Signals of the Oscillator Bulls Power	2709
Signals of the Oscillator Commodity Channel Index	2711
Signals of the Oscillator DeMarker	2715
Signals of the Indicator Double Exponential Moving Average	2719
Signals of the Indicator Envelopes	2723
Signals of the Indicator Fractal Adaptive Moving Average	2726
Signals of the Intraday Time Filter	2730
Signals of the Oscillator MACD	2732
Signals of the Indicator Moving Average	2738
Signals of the Indicator Parabolic SAR	2742
Signals of the Oscillator Relative Strength Index	2744
Signals of the Oscillator Relative Vigor Index	2750
Signals of the Oscillator Stochastic	2752
Signals of the Oscillator Triple Exponential Average	2757

Signals of the Indicator Triple Exponential Moving Average	2761
Signals of the Oscillator Williams Percent Range	2765
Trailing Stop Classes	2768
CTrailingFixedPips	2769
StopLevel	2770
ProfitLevel	2771
ValidationSettings	2772
CheckTrailingStopLong	2773
CheckTrailingStopShort	2774
CTrailingMA	2775
Period	2776
Shift	2777
Method	2778
Applied	2779
Init Indicators	2780
ValidationSettings	2781
CheckTrailingStopLong	2782
CheckTrailingStopShort	2783
CTrailingNone	2784
CheckTrailingStopLong	2785
CheckTrailingStopShort	2786
CTrailingPSAR	2787
Step	2788
Maximum	2789
Init Indicators	2790
CheckTrailingStopLong	2791
CheckTrailingStopShort	2792
Money Management Classes	2793
CMoneyFixedLot	2794
Lots	2795
ValidationSettings	2796
CheckOpenLong	2797
CheckOpenShort	2798
CMoneyFixedMargin	2799
CheckOpenLong	2800
CheckOpenShort	2801
CMoneyFixedRisk	2802
CheckOpenLong	2803
CheckOpenShort	2804
CMoneyNone	2805
ValidationSettings	2806
CheckOpenLong	2807
CheckOpenShort	2808
CMoneySizeOptimized	2809
DecreaseFactor	2810
ValidationSettings	2811
CheckOpenLong	2812
CheckOpenShort	2813
24 来自 MQL4	2814

MQL5 参考

MQL5是一种内置式计算机语言，用于设计交易的策略。这种语言是基于[MetaQuotes Software Corp.](#) 长期的网上交易平台经验开发的。通过这种语言，可以创建你自己的智能交易，使自己的交易策略能够完全自动地执行。而且，MQL5还能自定义客户指标，脚本和数据库。

MQL5内包含了大量可以分析当前及历史报价所必须的函数，并内置的内基本指标和函数来管理和支配这些交易。

MetaEditor 5(文本编辑器) 集合了编写 MQL5程序代码的各种语句。它能帮助使用者方便地写出规范的代码。MetaQuotes Language Dictionary 是 MQL5 语言的帮助工具。

简要指南包括功能、操作、储备词库，和其他语言结构分类，以便查找所使用的要素之相关语言描述。

MQL5 可以编写不同作用的程序代码：

- **EA交易** 运行处理它：加载函数和卸载函数，项目铃声提醒，定时项目，深度变化的市场事件，图标事件和自定义事件。

EA交易能够在提醒用户可以交易的同时，将交易订单自动送到交易服务器。EA交易储存在terminal_directory\MQL5\Experts 中。

- **自定义指标** 可用来编写新的技术指标，和内置的指标一样，它不能用来进行自动交易，只能作为分析数据的工具。自定义指标储存在terminal_directory\MQL5\Indicators中。
- **脚本** 是执行单一功能的一段程序，和EA交易不同，脚本不处理任何行动，除了开始事件（需要在脚本中亲自处理函数）。脚本是储存terminal_directory\MQL5\Scripts。
- **数据库** 被使用的自定义函数的集合，用来储存和分发常用的自定义程序块。数据库储存在terminal_directory\MQL5\Libraries。
- **包含文件** 常被使用的程序块源代码，这些文件能够被包含在EA交易，脚本，客户指标和数据库 的源代码中。使用包含文件比调用资料库更灵活快捷。包含可以存储在与源文件相同的目录—在这种情况下，指令“[#include](#)”。另一个储存包含文件是terminal_directory\MQL5\Include，指令<#include>。

2000-2011, [MetaQuotes Software Corp.](#)

语言基础

MetaQuotes Language 5(MQL5) 是一种面向对象的高水平的程序语言，它用来自动录入交易战略，为金融市场的各种分析定制智能指标。它不仅允许录入各种智能系统，更致力于实践操作，还能建立专属的图表工具帮您制定交易决策。

MQL5是以最流行的程序C++语言为蓝本的，相比与MQL4，新语言拥有[计数](#), [结构](#), [分类](#) 和 [事件处理](#)功能。通过增加嵌入式主要标签的数量，在MQL5的可执行程序的相互作用下，其他应用的运行就相对容易得多，MQL5的语法与C++的语法相似，这样一来，把现代程序语言译成它自己的语言就很容易。

为帮助您学习MQL系列语言，所有主题可分为如下步骤：

- [语法](#)
- [数据类型](#)
- [运行式和表达式](#)
- [运算符](#)
- [函数](#)
- [变量](#)
- [预处理程序](#)
- [面向对象的程序设计](#)

语法

至于语法, MQL5语言为程序交易策略语言与C++编程语言十分类似, 除了如下特点:

- 没有运算地址;
- 没有goto语句;
- 匿名计算无法删除;
- 级的构造函数没有常量;
- 无成倍继承.

查看相关

[计数](#), [结构和类](#), [继承](#)

注释

多行注释使用 /* 作为开始到 */ 结束，在这之间不能够嵌套。单行注释使用 // 作为开始到新的一行结束，可以被嵌套到多行注释之中。

示例:

```
//--- 单行线注释
/* Multi-
   line      // 嵌入式单行线注释
   comment
*/
```

标识符

标识符用来给变量和函数进行命名，长度不能超过63个字节。

特属性能够输入到标识符中，你可以使用数字0-9、拉丁字母大写A-Z和小写a-z(大小写有区分的) 还有下划线(_)。此外首字母不可以是数字，标识符不能和 [保留字](#)冲突。

示例:

```
NAME1 name1 Total_5 Paper
```

查看相关

[变量](#), [函数](#)

关键词

下面列出的是固定的保留字标识符，每个人标识符相当于一个动作，不能用来操作其他命令。

数据类型

bool	enum	struct
char	float	uchar
class	int	uint
color	long	ulong
datetime	short	ushort
double	string	void

访问分类符

const	private	protected
public	virtual	

存储类型

extern	input	static
------------------------	-----------------------	------------------------

操作符

break	do	return
case	else	sizeof
continue	for	switch
default	if	while
delete	new	

其他

false	#define	#property
this	#import	
true	#include	

数据类型

任何程序都依靠数据来运作，数据因目的不同可以有不同的类型。比如，访问数组可以用整型数据，价格可以用双精度的浮点型数据。在 MQL 5 中没有专门用来标记货币值的数据类型。

不同的数据类型有不同的处理速度，整型数据是最快的。双精度的数据处理需要特殊的协同处理器，然而，因为处理浮点型数据比较复杂，所以它比处理整型数据慢一些。

字符串是处理速度最慢的，因为它要存取动态内存。

基本数据类型：

- 整型数据（[字符型](#)，[短整型](#)，[整型](#)，[长型](#)，[无符字符型](#)，[无符短整型](#)，[无符号整型](#)，[无符长整型](#)）；
- 逻辑数据（[布尔型](#)）；
- [字符数据](#)（无符短整型）；
- 字符串数据（[字符串](#)）；
- 浮点型数据（[双精度型](#)，[浮点型](#)）；
- 颜色（[颜色](#)）；
- 日期和时间数据（[日期时间](#)）；
- 计数（[计数](#)）。

复杂的数据类型是：

- [结构数据](#)；
- [级层数据](#)。

根据OOP复杂数据类型被称作抽象数据类型。

颜色 和 时间日期 从外观上可以使我们更清楚的区分图表中的内容和参数的录入，在EA交易和自定义指标中经常使用这些数据类型（[Inputs](#) 标签）。颜色和日期时间数据用整数来表示。整型数据和浮点数据都属于数值（数字）型。

只有在 [公式](#) 中使用[类型分类](#)，除非提供指定强制类。

另见

[类型分类](#)

整型数据

在MQL5中整数有11个类型，如果逻辑程序需要，一些类型能与另一些类型一起使用，但是在此种情况下，要记住[类型转换](#) 规则。

下面列表中显示了每一类型的特性，此外，每一类型的最后一列均与C++程序类型相同。

类型	字节大小	最小值	最大值	C++ 类比
char	1	-128	127	char
uchar	1	0	255	unsigned char, BYTE
bool	1	0(false)	1(true)	bool
short	2	-32 768	32 767	short, wchar__t
ushort	2	0	65 535	unsigned short, WORD
int	4	- 2 147 483 648	2 147 483 647	int
uint	4	0	4 294 967 295	unsigned int, DWORD
color	4	-1	16 777 215	int, COLORREF
long	8	-9 223 372 036 854 775 808	9 223 372 036 854 775 807	__int64
ulong	8	0	18 446 744 073 709 551 615	unsigned __int64
datetime	8	0 (1970.01.01 0:00:00)	32 535 244 799 (3000.12.31 23:59:59)	__time64__t

整型值也可视为数字常数，颜色值和，日期-时间值，[字符常量](#) 和[计数](#)。

另见

[数据变换](#), [数值类型常量](#)

字符型，短整型，整型和长整型

字符型

字符型在内存里存储1字节（8位元组），允许二进制记录法 $2^8=256$ 表达值。字符型包括正值和负值，范围在-128-127之间。

无符号字符型

无字符型和字符型一样，也占据1字节内存，但与之不同的是，它只有正值。最小值是0，最大值是255，无字符型的第一个字母u是unsigned的缩写。

短整型

短型数据2字节（16位元组），所以它可以表达等于2的值和16: $2^{16} = 65\,536$ ，因此短型还是一个符号，包括正值和负值，范围在-32768到32767之间。

无符号短整型

无符短型是ushort，也占用2字节，最小值是0，最大值是65535。

整型

整型占用4字节内存（32元组），最小值是-2 147 483 648，最大值是2 147 483 647。

无符号整型

无符号整型就是uint。它占用4字节内存，区间在0到4 294 967 295之间。

长整型

长型占用8字节（64元组），最小值是-9 223 372 036 854 775 808，最大值是9 223 372 036 854 775 807。

无符号长整型

无符长型也占用8字节，能存储从0到18 446 744 073 709 551 615之间的值。

示例:

```
char   ch=12;
short  sh=-5000;
int     in=2445777;
```

无符长型不代表短型的负值，建立负值会导致意外的结果，该脚本会无限循环：

```
//--- 无限循环
void OnStart()
{
    uchar u_ch;
```

```

for(char ch=-128;ch<128;ch++)
{
    u_ch=ch;
    Print("ch = ",ch," u_ch = ",u_ch);
}

```

正确的转化是：

```

//--- 正确的变量
void OnStart()
{
    uchar u_ch;

    for(char ch=-128;ch<=127;ch++)
    {
        u_ch=ch;
        Print("ch = ",ch," u_ch = ",u_ch);
        if(ch==127) break;
    }
}

```

结果：

```

ch= -128 u_ch= 128
ch= -127 u_ch= 129
ch= -126 u_ch= 130
ch= -125 u_ch= 131
ch= -124 u_ch= 132
ch= -123 u_ch= 133
ch= -122 u_ch= 134
ch= -121 u_ch= 135
ch= -120 u_ch= 136
ch= -119 u_ch= 137
ch= -118 u_ch= 138
ch= -117 u_ch= 139
ch= -116 u_ch= 140
ch= -115 u_ch= 141
ch= -114 u_ch= 142
ch= -113 u_ch= 143
ch= -112 u_ch= 144
ch= -111 u_ch= 145
...

```

示例：

```

//--- 负值不能存储在无符类型中
uchar u_ch=-120;
ushort u_sh=-5000;
uint u_in=-401280;

```

十六进制：数字0-9，字母a-f，或A-F表示值10-15，从0x或0X开始。

示例：

```
0x0A, 0x12, 0X12, 0x2f, 0xA3, 0xa3, 0X7C7
```

另见：

[类型转换](#)

字符常量

字符作为MQL5中的字符串要素是统一字符设置的指数。它们是可以转换成整数的十六进制值，可以像加减法定理一样进行整数操作。

引号里的任何单一特质和十六进制的 ASCII 代码'\x10' 都能当做字符常量无符号短整型 型，例如，0型记录的数值是30，相当于在图标字符中代表调整归零。

示例：

```
void OnStart()
{
//--- 定义字符常量
int symbol_0='0';
int symbol_9=symbol_0+9; // 获得符号 '9'
//--- 输出常量值
printf("In a decimal form: symbol_0 = %d, symbol_9 = %d",symbol_0,symbol_9);
printf("In a hexadecimal form: symbol_0 = 0x%x, symbol_9 = 0x%x",symbol_0,symbol_9);
//--- 输入常量成字符串
string test="";
StringSetCharacter(test,0,symbol_0);
StringSetCharacter(test,1,symbol_9);
//--- 这是字符串中看起来像的东西
Print(test);
}
```

在程序源文本与常量型进行处理时，反斜线符号是编译器的控制字符，，一些符号，例如，单引号(')，双引号(")，反斜杠(\) 和控制字符都能被当做以反斜杠(\) 为起点的符号的集合，如下表：

字符名称	助记码或者图像	MQL5中的记录	数值
新线 (换行)	LF	'\n'	13
水平制表符	HT	'\t'	9
回车	CR	'\r'	10
反斜杠	\	'\\'	92
单引号	'	'\"'	39
双引号	"	'\"'	34
十六进制代码	hhhh	'\xhhhh'	1 to 4 十六进制字符
十进制代码	d	'\d'	从 0 到 65535的十进制代码

如果反斜杠后面跟着另一种字符而不是以上描述的类型，结果是未知的。

示例

```
void OnStart()
{
```

```

//--- 声明字符常量
int a='A';
int b='$';
int c='©';      // 代码 0xA9
int d='\xAE';   // 符号代码
//--- 输出打印常量
Print(a,b,c,d);
//--- 添加字符到字符串
string test="";
StringSetCharacter(test,0,a);
Print(test);
//--- 成串的替换字符
StringSetCharacter(test,0,b);
Print(test);
//--- 成串的替换字符
StringSetCharacter(test,0,c);
Print(test);
//--- 成串的替换字符
StringSetCharacter(test,0,d);
Print(test);
//--- 字符表示为数字
int a1=65;
int b1=36;
int c1=169;
int d1=174;
//--- 添加字符到字符串
StringSetCharacter(test,1,a1);
Print(test);
//--- 添加字符到字符串
StringSetCharacter(test,1,b1);
Print(test);
//--- 添加字符到字符串
StringSetCharacter(test,1,c1);
Print(test);
//--- 添加字符到字符串
StringSetCharacter(test,1,d1);
Print(test);
}

```

综上所述，常量（或变量）字符的值是字符表中的指标，指标存在整型，能以不同的方式输出。

```

void OnStart()
{
//---
int a=0xAE;      // 代码 符合于 the '\xAE' 文字
int b=0x24;      // 代码 $ 符合于 the '\x24' 文字
int c=0xA9;      // 代码 符合于 to the '\xA9' 文字
int d=0x263A;    // 代码 O 符合于 the '\x263A' 文字
//--- 显示值

```

```
Print(a,b,c,d);
//--- 添加字符到字符串
string test="";
StringSetCharacter(test,0,a);
Print(test);
//--- 成串的替换字符
StringSetCharacter(test,0,b);
Print(test);
//--- 成串的替换字符
StringSetCharacter(test,0,c);
Print(test);
//--- 成串的替换字符
StringSetCharacter(test,0,d);
Print(test);
//--- 合适的代码
int a1=0x2660;
int b1=0x2661;
int c1=0x2662;
int d1=0x2663;
//--- 添加黑桃字符
StringSetCharacter(test,1,a1);
Print(test);
//--- 添加红心字符
StringSetCharacter(test,2,b1);
Print(test);
//--- 添加方片字符
StringSetCharacter(test,3,c1);
Print(test);
//--- 添加梅花字符
StringSetCharacter(test,4,d1);
Print(test);
//--- 成串的字符文字示例
test="Queen\x2660Ace\x2662";
printf("%s",test);
}
```

字符型的内部表示法是 [无符号短整型](#)，字符常量能够接受从0到65535的值。

另见

[StringSetCharacter\(\)](#), [StringGetCharacter\(\)](#), [ShortToString\(\)](#), [ShortArrayToString\(\)](#),
[StringToShortArray\(\)](#)

日期时间型

日期时间型是为存储日期时间型预留的，开始日期是1970年1月1日，占8字节内存。

日期时间型常量可被当做数字串，由 6 个部分的字符组成：年、月、日（或是日、月、年）、时、分、秒，数据以 D 开头，用单引号括起。

日期（年、月、日）、时间（时、分、秒），或是一起被省略，值起于1970年1月1日，止于3000年12月31日。

示例：

```
D'2004.01.01 00:00'      // 新年
D'1980.07.19 12:30:27'
D'19.07.1980 12:30:27'
D'19.07.1980 12'         // 等同于 D' 1980. 07. 19 12: 00: 00'
D'01.01.2004'           // 等同于 D' 01. 01. 2004 00: 00: 00'
D'12:30:27'             // 等同于 D [ 编辑日期] 12: 30: 27'
D''                     // 等同于 D [ 编辑日期] 00: 00: 00'
```

另见

[日期类型结构](#), [日期和时间](#), [时间到字符串](#), [字符串到时间](#)

颜色型

颜色型是为了存储颜色信息的，占用4个字节，头1个字节忽略不计，其他3个字节包括红绿蓝3个数据。

颜色数据可以用三种方法表示：字符数据、整型数据或者是颜色名（只能是已命名的 [网页-颜色](#)）。

字符数据的表达方法是用三个数字来表示三种主要颜色：红、绿、蓝的比例。数据以 C 开头，用单引号括住。数字的值在 0 ~ 255 之间按比例选取。

整数数据的表达方法使用十六进制或十进制数字。十六进制数字如 0x00BBGGRR，其中 RR 是红色元素的比例，GG 是绿色的比例，BB 是蓝色的比例。十进制数不能直接体现红绿蓝的比例，而是十六进制数字的十进制表示方式。

特殊颜色名可以参考 [网页-颜色](#) 表。

示例：

```
// --- 字面值
C'128,128,128'    // 灰色
C'0x00,0x00,0xFF' // 蓝色
//color names
clrRed            // 红色
clrYellow         // 黄色
clrBlack          // 黑色
// --- 整数表示
0xFFFFFFFF        // 白色
16777215          // 白色
0x008000          // 绿色
32768             // 绿色
```

另见

[网站色彩](#)，[颜色到字符串](#)，[字符串到颜色](#)，[类型转换](#)

布尔类型

布尔类型是用来存储true或者false的逻辑值的，它们的数字表示法分别是1和0。

示例：

```
bool a = true;
bool b = false;
bool c = 1;
```

内置法最多就是1字节，在逻辑表达式里可以注释，各种表达式都能用其他整数——编译器不会产生任何错误，在这种情况下，0值表示成false,其他值表示true。

示例：

```
int i=5;
double d=-2.5;
if(i) Print("i = ",i," and is set to true");
else Print("i = ",i," and is set to false");

if(d) Print("d = ",d," and has the true value");
else Print("d = ",d," and has the false value");

i=0;
if(i) Print("i = ",i," and has the true value");
else Print("i = ",i," and has the false value");

d=0.0;
if(d) Print("d = ",d," and has the true value");
else Print("d = ",d," and has the false value");

//--- 执行结果
//   i = 5 且有真值
//   d = -2.5 且有真值
//   i = 0 且有错误值
//   d = 0 且有错误值
```

另见

[布尔体系操作](#), [优先规则](#)

枚举类型

枚举类型数据属于数据集合的限制额，最典型的是枚举类型：

```
enum name of enumerable type
{
    list of values
};
```

该值列表是分割逗号命名的标识符常量列表。

示例：

```
enum months // 已命名常量的计算
{
    January,
    February,
    March,
    April,
    May,
    June,
    July,
    August,
    September,
    October,
    November,
    December
};
```

计算清楚之后，结果是新的4字节整数数据类型值。新数据类型的描述可以严格地编译到通过的常量控制类型里，因为列举介绍了新命名的数据。在上述例子中，一月常量的值是0，二月是-1，十二月是-11。

规则：如果某一确定值并没有列举到命名常量-计算的一员，它的新值将会自动形成。如果是计算左边的，就会出现0值，在随后出现的值中，将会以先前计算的值为基准递增。

示例：

```
enum intervals // 已命名常量的计算
{
    month=1, // 间隔一个月
    two_months, // 两个月
    quarter, // 三个月 - 四分之一
    halfyear=6, // 半年
    year=12, // 一年 - 12个月
};
```

注释

- 不像C++，在MQL5里程序段的大小代表了计数类型，通常为4字节，也就是运算符（月）的返回值是4。
- 不像C++，在MQL5里，匿名运算无法显示，在列举关键字后，需要指定独立名称。

另见

类型转换

真实型（双精度型，浮点型）

真实型（或浮点型）以小数部分为代表值，在MQL5语言里，浮点型数据有两种类型，在内存中实型数据的表示方法由IEEE 754水平规定，它并不依赖平台、操作系统和程序语言。

类型	字节大小	最小正值	最大值	精确表示	C++ 类似物
浮点型	4	1.175494351e-38	3.402823466e+38	7位有效数字	浮点型
双型	8	2.2250738585072014e-308	1.7976931348623158e+308	15位有效数字	双型

双精度名称是为了表示这些**浮点**型数据的双倍准确率，在大多数情况下，**双精度**型是最方便的，**浮点**型数据的精度限制是不够的，原因就在于**浮点**型数据还要节约内存（这就是真实型数据庞大的重要性）。

浮点型数据由整数部分、小数点(.)和小数部分组成,其中整数部分和小数部分为一系列十进制数字。

示例：

```
double a=12.111;  
double b=-956.1007;  
float c =0.0001;  
float d =16;
```

有更科学的方法输入实常数，通常这些方法比传统方法更简洁。

示例：

[illegible]

在二进制系统中，实型数据以限制精确度来存储，而常用作十进制计数法。这就是在十进制系统中，许多被取代的数字在二进制系统中被输出为无数小数点的原因。

例如，0.3和0.7的小数部分被取代，而0.25却被精确保留，因为它的有效数字是两位。

就这一点而言，不要实际地区对比两个真实数据，因为对比是不精确的。

示例：

```
void OnStart()  
{  
    //---
```

```

double three=3.0;
double x,y,z;
x=1/three;
y=4/three;
z=5/three;
if(x+y==z) Print("1/3 + 4/3 == 5/3");
else Print("1/3 + 4/3 != 5/3");
// 结果: 1/3 + 4/3 != 5/3
}

```

如果你仍需要对比两个真实型数据，有两种方法，第一种，在同样的小数位对比他们之间的不同。

示例：

```

bool EqualDoubles(double d1,double d2,double epsilon)
{
    if(epsilon<0) epsilon=-epsilon;
    //---
    if(d1-d2>epsilon) return false;
    if(d1-d2<-epsilon) return false;
    //---
    return true;
}
void OnStart()
{
    double d_val=0.7;
    float f_val=0.7;
    if(EqualDoubles(d_val,f_val,0.0000000000000001)) Print(d_val," equals ",f_val);
    else Print("Different: d_val = ",DoubleToString(d_val,16),
               " f_val = ",DoubleToString(f_val,16));
    // 结果: 不同: d_val = 0.7000000000000000    f_val = 0.6999999880790710
}

```

上例中第五位比DBL_ EPSILON多，值是2.2204460492503131e-016，与浮点型数据相一致的是FLT_ EPSILON = 1.192092896e-07。这些值有如下意义：满足条件的最低值 1.0 + DBL_ EPSILON! = 1.0（大量的浮点型数值 1.0 + FLT_ EPSILON! = 1.0）。

第二种方法通过0将两种真实型数据进行了标准对比，它是无意义的，因为任何标准化操作都能给出非标准的结果。

示例：

```

bool CompareDoubles(double number1,double number2)
{
    if(NormalizeDouble(number1-number2,8)==0) return(true);
    else return(false);
}
void OnStart()
{
    double d_val=0.3;
    float f_val=0.3;
}

```

```

    if(CompareDoubles(d_val,f_val)) Print(d_val," equals ",f_val);
    else Print("Different: d_val = ",DoubleToString(d_val,16),
               " f_val = ",DoubleToString(f_val,16));
// 结果: 不同: d_val = 0.3000000000000000    f_val = 0.3000000119209290
}

```

一些数字协同处理器的操作能够导致无效的真实型数字，不能运用到数字操作和对比中，因为用无效真实型数据的操作结果是不能定义的。例如，当想要计算2的反正弦，结果可能无穷负。

示例：

```

double abnormal = MathArcsin(2.0);
Print("MathArcsin(2.0) =",abnormal);
// 结果:  Mat hArcsi n( 2. 0)  = - 1. #I ND

```

除了负无穷大也还有正无穷大和NaN（不是数字），确定数字是否是无效的，可以运用`MathIsValidNumber()`。功能，根据IEEE标准，可以用专用机描述。例如，双精度型正无穷代表小的0x7FF0 0000 0000 0000。

示例：

```

struct str1
{
    double d;
};
struct str2
{
    long l;
};

//--- 开始
str1 s1;
str2 s2;
//---
s1.d=MathArcsin(2.0);          // 获得无效数据 -1.#IND
s2=s1;
printf("1.   %f %I64X",s1.d,s2.l);
//---
s2.l=0xFFFFF0000000000000;    // 无效数据 -1.#QNAN
s1=s2;
printf("2.   %f %I64X",s1.d,s2.l);
//---
s2.l=0x7FF7000000000000;      // 最大 nonnumber SNaN
s1=s2;
printf("3.   %f %I64X",s1.d,s2.l);
//---
s2.l=0x7FF8000000000000;      // 最小 nonnumber QNaN
s1=s2;
printf("4.   %f %I64X",s1.d,s2.l);
//---

```

```

s2.l=0x7FFF000000000000;    // 最大 nonnumber QNaN
s1=s2;
printf("5.    %f %I64X",s1.d,s2.l);
//---
s2.l=0x7FF0000000000000;    // 正无穷大 1.#I NF 和最小 nonnumber SNaN
s1=s2;
printf("6.    %f %I64X",s1.d,s2.l);
//---
s2.l=0xFFFF000000000000;    // 负无穷大 -1.#I NF
s1=s2;
printf("7.    %f %I64X",s1.d,s2.l);
//---
s2.l=0x8000000000000000;    // 负零 -0.0
s1=s2;
printf("8.    %f %I64X",s1.d,s2.l);
//---
s2.l=0x3FE0000000000000;    // 0.5
s1=s2;
printf("9.    %f %I64X",s1.d,s2.l);
//---
s2.l=0x3FF0000000000000;    // 1.0
s1=s2;
printf("10.   %f %I64X",s1.d,s2.l);
//---
s2.l=0x7FEFFFFFFFFFFFFFFF;    // 最大的规格化数字 (MAX_DBL)
s1=s2;
printf("11.   %.16e %I64X",s1.d,s2.l);
//---
s2.l=0x0010000000000000;    // 最小的正规格化 (MIN_DBL)
s1=s2;
printf("12.   %.16e %.16I64X",s1.d,s2.l);
//---
s1.d=0.7;                    // 显示数字0.7-无限循环部分
s2=s1;
printf("13.   %.16e %.16I64X",s1.d,s2.l);
/*
1.  -1.#IND00 FFF8000000000000
2.  -1.#QNaN0 FFFF000000000000
3.   1.#SNaN0 7FF7000000000000
4.   1.#QNaN0 7FF8000000000000
5.   1.#QNaN0 7FFF000000000000
6.   1.#INF00 7FF0000000000000
7.  -1.#INF00 FFF0000000000000
8.  -0.000000 8000000000000000
9.   0.500000 3FE0000000000000
10.  1.000000 3FF0000000000000
11.  1.7976931348623157e+308 7FEFFFFFFFFFFFFFFF
12.  2.2250738585072014e-308 0010000000000000

```



```
13. 6.9999999999999996e-001 3FE6666666666666  
*/
```

另见

[双精度型到字符串](#)， [标准化双精度型](#) - [数字类型常量](#)

字符串数据

字符串数据是用来存储文本串的，文本串是以编码的形式末尾为0的字符序列，每串常数分配给一个变量，每串数据都用双引号引出写着“这是一串字符常量”。

如果字符串中包括双引号(")，那么反斜杠字符(\) 必须放其前面。如果反斜杠字符放其前面的话，任何特殊字符常量都可以写入字符串。

示例：

```
string svar="This is a character string";
string svar2=StringSubstr(svar,0,4);
Print("Copyright symbol\t\x00A9");
FileWrite(handle,"This string contains a new line symbols \n");
string MT5path="C:\\Program Files\\MetaTrader 5";
```

```
//--- объявим длинную константную строку
string HTML_head="<!DOCTYPE html PUBLIC \"-//W3C//DTD XHTML 1.0 Transitional//EN\"
    \"http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd\">\n"
    "<html xmlns=\"http://www.w3.org/1999/xhtml\">\n"
    "<head>\n"
    "<meta http-equiv=\"Content-Type\" content=\"text/html; charset=u"
    "<title>Trade Operations Report</title>\n"
    "</head>";

//--- выведем в журнал константную строку
Print(HTML_head);
}
```

另见

[转换函数](#)， [字符串函数](#)， [打开文件](#)， [文件阅读字符串](#)， [文件输入字符串](#)

架构和类

架构

架构是设定任何类型的元素（除了 [空型](#)），因此，架构要组合不同类型的逻辑相关资料。

架构说明

以下描述定义结构类型数据：

```
struct structure_name
{
    elements_description
};
```

架构名称不能用作标识符（变量或功能的名称），应该直接记录在MQL5架构元素中，没有定位，在C++中需要遵循以下指令来定义这些命令：

```
#pragma pack(1)
```

如果想在架构中执行另一定位，运用辅助程序，“fillers”填充正确大小。

示例：

```
struct trade_settings
{
    uchar   slippage;      // 许可的下降值-1字节大小
    char    reserved1;    // 跳过 1 字节
    short   reserved2;    // 跳过 2 字节
    int     reserved4;    // 跳过另4个字节。确保定位8字节界限
    double  take;         // 固定利润价格值
    double  stop;         // 受保护的止损价格值
};
```

线性构造的描述只对动态传输功能入口有效。

注意：该例子列举的是错误的设计数据，在[双精度](#)型数据中，首先声明take和stop的数据大小比较好，然后是slippage，在此情况下，数据的内部表达式与 `#pragma pack()` 无关，都是一样的。

如果该构造包含[字符串](#)型数据变量，和/或[目标动态数组](#)，编译器给架构分派隐含的构造函数。构造函数设置所有[字符串](#)型和初始化目标动态数组的所有结构。

简单构造

架构并不包括字符串或动态数组，这些被叫做简单构造；该构造的变量可以互相[自由复制](#)，不同结构也可以，简单结构的变量和数组，都能从DLL以参数的形式传递到[输入功能](#)。

接入架构会员入口

架构是一种新的数据类型用来声明这种类型变量的，只有在一个项目开始是架构可以运行，架构会员可以通过 [操作点](#)（.）进入接口。

示例：

```
struct trade_settings
{
    double take;           // 利润固定价格值
    double stop;           // 受保护的止损价格值
    uchar slippage;        // 可接受的下降值
};
//--- 创建和初始化交易设置类型的变量
trade_settings my_set={0.0,0.0,5};
if (input_TP>0) my_set.take=input_TP;
```

类

类与架构的不同之处在于：

- 在声明中使用关键字类；
- 默认情况下，所有类成员都能通过独自通过指示符，除非另行表明的，架构里的数据成员都能通过默认类型，除非是另行标注的。
- 甚至在类中没有虚函数的情况下，类目标总是有一个 [虚函数](#) 图表，结构类没有虚函数；
- [新功能](#)操作也能应用到类目标中，但不能应用到结构类中；
- 类只能从类中 [继承](#)，结构也只能从架构中继承。

类和架构都有明确的构造函数和解构方法，如果构造函数定义明确，架构或类别变量的初始化进行初始化序列是不可能的。

示例：

```
struct trade_settings
{
    double take;           // 利润固定价格值
    double stop;           // 受保护的止损价格值
    uchar slippage;        // 可接受的下降值
    //--- 构造函数
    trade_settings() { take=0.0; stop=0.0; slippage=5; }
    //--- 析构函数
    ~trade_settings() { Print("This is the end"); }
};
//--- 编译器生成一个无法初始化的错误信息
trade_settings my_set={0.0,0.0,5};
```

构造函数和析构函数

([void](#)) .

```
//+-----+
//|  класс для работы с датой  |
//+-----+
class MyDateClass
{
private:
    int          m_year;          // год
    int          m_month;         // месяц
    int          m_day;           // день месяца
    int          m_hour;          // час в сутках
    int          m_minute;        // минуты
    int          m_second;        // секунды
public:
    ///--- конструктор по умолчанию
    MyDateClass(void);

    ///--- конструктор с параметрами
    MyDateClass(int h,int m,int s);
};
```

MyDateClass:

```
//+-----+
//|  конструктор по умолчанию  |
//+-----+
MyDateClass::MyDateClass(void)
{
    ///---
    MqlDateTime mdt;
    datetime t=TimeCurrent(mdt);
```

```

    m_year=mdt.year;
    m_month=mdt.mon;
    m_day=mdt.day;
    m_hour=mdt.hour;
    m_minute=mdt.min;
    m_second=mdt.sec;
    Print(__FUNCTION__);
}

//+-----+
//|  конструктор с параметрами  |
//+-----+
MyDateClass::MyDateClass(int h,int m,int s)
{
    MqlDateTime mdt;
    datetime t=TimeCurrent(mdt);
    m_year=mdt.year;
    m_month=mdt.mon;
    m_day=mdt.day;
    m_hour=h;
    m_minute=m;
    m_second=s;
    Print(__FUNCTION__);
}

```

TimeCurrent() ,

(m_year, m_month m_day)

```

//+-----+
//|  класс с конструктором по умолчанию  |
//+-----+
class CFoo
{
    datetime m_call_time; // время последнего обращения к объекту
public:
    ///--- конструктор с параметром, имеющим значение по умолчанию, не является конструктором
    CFoo(datetime t=0){m_call_time=t;};
    string ToString(){return(TimeToString(m_call_time,TIME_DATE|TIME_SECONDS));};
};

//+-----+
//|  Script program start function  |
//+-----+

```

```

void OnStart()
{
    CFoo foo; // такой вариант использовать можно - будет вызван конструктор с параметром
    //--- допустимые варианты создания объекта CFoo
    CFoo foo1(TimeCurrent()); // первый способ автоматического создания объекта
    CFoo foo2(); // второй способ автоматического создания объекта
    CFoo foo3=TimeTradeServer(); // третий способ автоматического создания объекта
    //--- допустимые варианты создания указателей CFoo с помощью оператора new
    CFoo *foo4=new CFoo();
    CFoo *foo5=new CFoo(TimeTradeServer());
    CFoo *foo6=GetPointer(foo5); // теперь foo5 и foo6 указывают на один и тот же объект
    CFoo *foo7,*foo8;
    foo7=new CFoo(TimeCurrent());
    foo8=GetPointer(foo7); // foo7 и foo8 указывают на один и тот же объект
    //CFoo foo_array[3]; // такой вариант использовать нельзя - конструктор по умолчанию не определен
    //CFoo foo_dyn_array[]; // такой вариант использовать нельзя - конструктор по умолчанию не определен
    //--- выведем значения m_call_time
    Print("foo.m_call_time=",foo1.ToString());
    Print("foo1.m_call_time=",foo1.ToString());
    Print("foo2.m_call_time=",foo2.ToString());
    Print("foo3.m_call_time=",foo3.ToString());
    Print("foo4.m_call_time=",foo4.ToString());
    Print("foo5.m_call_time=",foo5.ToString());
    Print("foo6.m_call_time=",foo6.ToString());
    Print("foo7.m_call_time=",foo7.ToString());
    Print("foo8.m_call_time=",foo8.ToString());
    //--- удалим динамически созданные объекты
    delete foo4;
    delete foo5;
    delete foo7;
}

```

```

//CFoo foo_array[3]; // такой вариант использовать нельзя - конструктор по умолчанию не определен

```

```

//CFoo foo_dyn_array[]; // такой вариант использовать нельзя - конструктор по умолчанию не определен

```

"default constructor is not defined".

```

//+-----+

```

```

//| класс без конструктора по умолчанию |
//+-----+
class CFoo
{
    string      m_name;
public:
    CFoo(string name) { m_name=name;}

};
//+-----+
//| Script program start function |
//+-----+
void OnStart()
{
    //--- при компиляции получим ошибку "default constructor is not defined"
    CFoo badFoo[5];
}

```

CFoo

-

.

/

/

/

.

.

(
)

-

/

/

(

).

:

•

_____;

•

_____;

•

(_____)

```

//+-----+
//| класс для хранения фамилии и имени персонажа |
//+-----+
class CPerson
{
    string      m_first_name;    // имя
    string      m_second_name;   // фамилия
public:
    ///--- пустой конструктор по умолчанию
    CPerson() {Print(__FUNCTION__);};

    ///--- параметрический конструктор
    CPerson(string full_name);

    ///--- конструктор со списком инициализации
    CPerson(string surname,string name): m_second_name(surname), m_f
    void PrintName() {PrintFormat("Name=%s Surname=%s",m_first_name,m_second_name);};
};
//+-----+
//|
//+-----+
CPerson::CPerson(string full_name)
{
    int pos=StringFind(full_name," ");
    if(pos>=0)
    {
        m_first_name=StringSubstr(full_name,0,pos);
        m_second_name=StringSubstr(full_name,pos+1);
    }
}
//+-----+
//| Script program start function |
//+-----+
void OnStart()
{
    ///--- получим ошибку "default constructor is not defined"
    CPerson people[5];
    CPerson Tom="Tom Sawyer";           // Том Сойер
    CPerson Huck("Huckleberry","Finn"); // Гекльберри Финн
    CPerson *Pooh = new CPerson("Whinnie","Pooh"); // Винни Пух
    ///--- выведем значения
    Tom.PrintName();
    Huck.PrintName();
    Pooh.PrintName();

    ///--- удалим динамически созданный объект
    delete Pooh;
}

```

CPerson

:

1. _____ ;
2. _____ ;
3. _____ - m_second_name
(surname) m_first_name(name) .

член_класса (список выражений)

m_first_name, m_second_name.

```
//+-----+
//| базовый класс |
//+-----+
class CFoo
{
    string m_name;
public:
    ///--- конструктор со списком инициализации
    CFoo(string name) : m_name(name) { Print(m_name); }
};
//+-----+
//| потомок класса CFoo |
//+-----+
class CBar : CFoo
{
    CFoo m_member; // член класса является объектом предка
public:
```

```

//--- конструктор по умолчанию в списке инициализации вызывает конструктор предка
CBar(): m_member(_Symbol), CFoo("CBAR") {Print(__FUNCTION__);}

};
//+-----+
//| Script program start function |
//+-----+
void OnStart()
{
    CBar bar;
}

```

```

                                bar
                                CBar( ),
                                CFoo,
                                m_member.

```

析构函数是一种特殊功能，当类目标被破坏时自动调用，析构函数的名称用波浪字符(~) 以分类名输入。串型数据、动态数组和目标函数，不管破坏函数是否出现，无论如何都不会初始化。如果存在破坏函数，该行为在召回破坏者后会执行。

破坏函数总是 [虚拟的](#)，无论[虚拟](#)值存在与否。

规定的分类方法

分类功能方式既能被内在分类定义也能被外在分类定义，如果该方法被分类定义，主题会正确显示。

示例：

```

class CTetrisShape
{
protected:
    int          m_type;
    int          m_xpos;
    int          m_ypos;
    int          m_xsize;
    int          m_ysize;
    int          m_prev_turn;
    int          m_turn;
    int          m_right_border;
public:
    void          CTetrisShape();
    void          SetRightBorder(int border) { m_right_border=border; }
    void          SetYPos(int ypos)         { m_ypos=ypos;           }
    void          SetXPos(int xpos)         { m_xpos=xpos;           }
    int           GetYPos()                  { return(m_ypos);        }
    int           GetXPos()                  { return(m_xpos);        }
    int           GetYSize()                 { return(m_ysize);       }
    int           GetXSize()                 { return(m_xsize);       }
    int           GetType()                  { return(m_type);        }
    void          Left()                     { m_xpos-=SHAPE_SIZE;    }
    void          Right()                    { m_xpos+=SHAPE_SIZE;    }
    void          Rotate()                   { m_prev_turn=m_turn; if(++m_turn>3) }
    virtual void  Draw()                     { return;                }
    virtual bool  CheckDown(int& pad_array[]);
    virtual bool  CheckLeft(int& side_row[]);
    virtual bool  CheckRight(int& side_row[]);
};

```

SetRightBorder (整数边框) 功能直接被内置的CTetrisShape 定义。

CTetrisShape 构造函数和CheckDown(int& pad_array[]) , CheckLeft(int& side_row[]) 、 CheckRight (int& side_row[]) 方法在未被定义的情况下只能被内置分类定义，这些功能的定义会在代码中进一步体现。为了定义外在分类方法，使用[范围解析操作](#)功能，该分类名称与范围名称一样使用。

示例：

```
//+-----+
//| 基本类的构造函数 |
//+-----+
void CTetrisShape::CTetrisShape()
{
    m_type=0;
    m_ypos=0;
    m_xpos=0;
    m_xsize=SHAPE_SIZE;
    m_ysize=SHAPE_SIZE;
    m_prev_turn=0;
    m_turn=0;
    m_right_border=0;
}
//+-----+
//| 检测向下的能力（为竖条和方块） |
//+-----+
bool CTetrisShape::CheckDown(int& pad_array[])
{
    int i,xsize=m_xsize/SHAPE_SIZE;
    //---
    for(i=0; i<xsize; i++)
    {
        if(m_ypos+m_ysize>=pad_array[i]) return(false);
    }
    //---
    return(true);
}
```

公用，受保护的和私用编辑器接入

当一个新的类进行时，会从外部限制类进入，关键字`private`或者`protected`此时就起到了作用，在此情况下，隐藏数据只能从相同分类的功能方法通路计入，如果保护关键字使用了，隐藏数据只能从分类方法中接入，就是该分类的[继承接入](#)，限制接入到分类方法中时也可以使用相同方法。

如果需要完全打开此链接，使用`public`关键字。

示例：

```
class CTetrisField
{
private:
    int          m_score;                // 得分
    int          m_ypos;                 // 图形当前位置
    int          m_field[FIELD_HEIGHT][FIELD_WIDTH]; // DCM矩阵
    int          m_rows[FIELD_HEIGHT];   // DCM行的编号
    int          m_last_row;             // 最后一个自由行
    CTetrisShape *m_shape;               // 俄罗斯方块图形
    bool         m_bover;                // 游戏结束
public:
    void         CTetrisField() { m_shape=NULL; m_bover=false; }
    void         Init();
    void         Deinit();
    void         Down();
    void         Left();
}
```

```
void      Right();  
void      Rotate();  
void      Drop();  
private:  
void      NewShape();  
void      CheckAndDeleteRows();  
void      LabelOver();  
};
```

任何类成员和方法都在公开说明符后声明：（在下一个指示符输入之前）可以通过程序计入目标，示例中需要如下按键功能：CTetrisField()，Init()，Deinit()，Down()，Left()，Right()，Rotate() 和 Drop()。

在接入到说明符元素后，每个成员都要自动声明：（在接入下一个说明符之前）只能通过该类成员功能接入，接入到元素的分类通常在冒号(:) 后，多次出现在类别定义中。

基本分类的接入成员在派生类通过[继承](#)进行定义。

另见

[面向对象的程序设计](#)

动态数组目标

动态数组

动态数组占用40字节内容，最大的四次元数组也可被声明。

```
struct MqlArrayObject
{
    ushort type;           // 编码的数据类型
    ushort flags;          // 决定数组状态的标记
    uint item_len;         // 字节数组元素的大小
    int allocated;         // 为数组分配的实际大小
    int range0;            // 第一维尺寸大小（开始等于0）
    int range1;            // 第二维尺寸大小，若有的话，否则0
    int range2;            // 第三维尺寸大小，若有的话，否则0
    int range3;            // 第四维尺寸大小，若有的话，否则0
    int reserved0;         // 适用的数据
    int reserved1;         // 内部
    int reserved2;         // 用法
};
```

当定义一个动态数组，（方括号里第一对未知值数组），编译器自动编译上述结构变量（动态数组变量）为正确的初始化提供代码。

当声明程序在能见度之上数组自动释放。

示例：

```
double matrix[][10][20]; // 三维动态数组
ArrayResize(matrix,5);    // 设置第一维大小
```

静态数组

当有效阵列维数被定义后，编译器会再分配必要的内存，这种数组叫做静态数组。然而，编译器会为动态数组目标分配额外内存，目标和再分配的静态缓冲器（存储数组的内存部分）有关。

创建动态数组目标是需要经过静态数组确定参数和部分功能的。

Examples:

```
double stat_array[5]; // 1维静态数组
some_function(stat_array);
...
bool some_function(double& array[])
{
    if(ArrayResize(array,100)) return(false);
    ...
    return(true);
}
```

在架构中的数组

当静态数组被描述成架构中的一员，动态数组目标不能创建，在windows API中是为了确保数据结构的兼容性。

然而，静态数组都能被描述成结构成员，也能通过MQL5功能，在此情况下，当通过一个动态数组临时目标的常量时，与静态数组连接—架构成员就会产生。

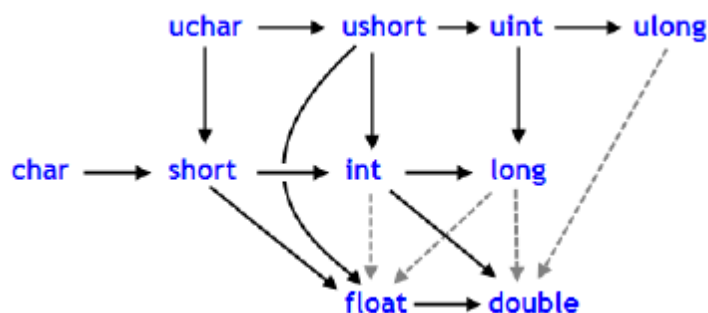
另见

[数组函数](#), [初始变量](#), [虚拟范围和时间变量](#), [创建和删除目标](#)

类型转换

创建数字符

有必要把一组数字类型变化成另一种数字类型，但并非作用数字类型都能转换，下面是允许转换的模式：



箭头指明表示转换方向，期间没有任何损失信息，[布尔型](#)可以取代字符类型（只占用1字节），[颜色型](#)可以取代整型（4字节），[日期时间型](#)可以取代长型（占用8字节）。四条灰色虚线，也带有箭头，在精确度缺失时，表示转化。例如，与123456789相等的整数（[int](#)）就高于[浮点型](#)表示的数字。

```

int n=123456789;
float f=n;      // f 内容同于 1.234567892E8
Print("n = ",n,"    f = ",f);
// 结果 n= 123456789    f= 123456792.00000

```

转化成浮点型的数字有同样的命令，但是精确度比较低。与黑色箭头不同的是，转化允许部分数据丢失。字符型和无符号字符型间的转化，短整型和无符号短整型间的转化，整型及无符号整型的转化，长整型和无符号长整型的转化（双向转化），都可能导致数据丢失。

因此浮点值转化成整数型的结果就是，经常删除小数部分。如果想把浮点转成最近的整数（任何情况下都很有用），应该使用 [MathRound\(\)](#)。

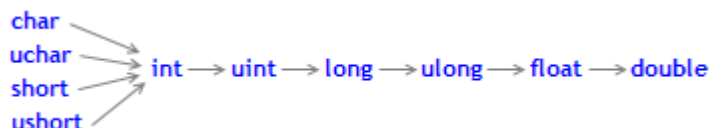
示例：

```

// --- 重力加速度
double g=9.8;
double round_g=(int)g;
double math_round_g=MathRound(g);
Print("round_g = ",round_g);
Print("math_round_g = ",math_round_g);
/*
Result:
round_g = 9
math_round_g = 10
*/

```

如果用二进制合并两个值，执行操作前，需要按照下表的先后顺序，把低类型转化成高类型。



数字类型字符型，无符号字符型，短整型，和无符号短整型，无条件的转化成整型。

示例：

```

char c1=3;
//--- 第一示例
double d2=c1/2+0.3;
Print("c1/2 + 0.3 = ",d2);
// 结果:   c1/2+0.3 = 1.3

//--- 第二示例
d2=c1/2.0+0.3;
Print("c1/2.0 + 0.3 = ",d2);
// 结果:   c1/2.0+0.3 = 1.8
  
```

计算的表达式由两种操作构成。示例一，字符型变量c1转化成整型临时变量，因为除法操作中的第二运算对象，常量2，是高类型整型。因此3/2的整数我们取整数值，1。

在示例一中的第二步中，第二运算对象是常量0.3，双精度型，那么结果就是第一运算对象转化成1.0双精度型临时变量。

示例二中，字符型c1变量转化成双精度型临时变量，因为除法操作的第二运算对象，常量2.0，是双精度型；无进一步转化。

数型类型转换

在MQL5语言表达式中，使用明确和含蓄两种类型转换。明确类型转换如下：

```
var_1 = (type)var_2;
```

表达式或者函数执行的结果可用作 var_2 变量。明确类型转换的函数记录也可以如此：

```
var_1 = type(var_2);
```

基于第一示例，考虑下明确类型转换。

```

//--- 第三示例
double d2=(double)c1/2+0.3;
Print("(double)c1/2 + 0.3 = ",d2);
// 结果:   (双精度) c1/2+0.3 = 1.80000000
  
```

做除法前，c1变量明确为双精度型。现在整型常量2，转换成双精度型2.0，因为转换造成第一运算对象成为双精度型。实际上，明确类型转换时是一种一元运算操作。

此外，当尝试转换类型时，结果可能超出允许范围内。在这个情况下，容易发生截断。例如：

```

char c;
uchar u;
c=400;
  
```

```
u=400;
Print("c = ",c); // 结果 c=-112
Print("u = ",u); // 结果 u=144
```

在运算完成之前（除了数据已被定义的），数据会根据优先级被转换。当定义数据的操作完成前，数据会转换成被定义的数据类型。

示例：

```
int i=1/2; // 无类型转换，结果是 0
Print("i = 1/2 ",i);

int k=1/2.0; // 表达式转换到双精度型，
Print("k = 1/2 ",k); // 那么就是到整型的目标类型，结果是0

double d=1.0/2.0; // 无类型转换，结果是 0.5
Print("d = 1/2.0; ",d);

double e=1/2.0; // 表达式转换到双精度型，
Print("e = 1/2.0; ",e); // 同于目标类型，结果为0.5

double x=1/2; // 整型表达式转换到双精度目标类型，
Print("x = 1/2; ",x); // 结果是 0.0
```

字符串类型转换

字符串类型是几种简单类型中的最高级别。因此，如果操作的运算对象之一为字符串，第二运算对象自动转换成字符串。注意的是，对于字符串，添加独立二元操作是可以的。允许任何字符串明确转换成数字类型。

示例：

```
string s1=1.0/8; // 表达式转换到双精度型，
Print("s1 = 1.0/8; ",s1); // 那么就是到字符串的目标类型，
// 结果 is "0.12500000"（包括10个字符的字符串）

string s2=NULL; // 字符串无法初始化
Print("s2 = NULL; ",s2); // 结果是空值字符串
string s3="Ticket N"+12345; // 表达式转换到字符串类型
Print("s3 = \"Ticket N\"+12345",s3);

string str1="true";
string str2="0,255,0";
string str3="2009.06.01";
string str4="1.2345e2";
Print(bool(str1));
Print(color(str2));
Print(datetime(str3));
Print(double(str4));
```

简单结构类型的类型转换

只要两个结构所有成员都是数字类型，那么简单结构类型数据彼此从属。这种情况下的分配操作运算对象一定是结构类型。不执行成员类型转换，只做简单复制。如果结构大小不同，就复制小的字节数。因此MQL5中联盟缺乏可以补偿。

示例：

```
struct str1
{
    double d;
};
//---
struct str2
{
    long l;
};
//---
struct str3
{
    int low_part;
    int high_part;
};
//---
struct str4
{
    string s;
};
//+-----+
void OnStart()
{
    str1 s1;
    str2 s2;
    str3 s3;
    str4 s4;
    //---
    s1.d=MathArcsin(2.0); // 获得无效数据 -1. # IND
    s2=s1;
    printf("1. %f %I64X",s1.d,s2.l);
    //---
    s3=s2;
    printf("2. high part of long %.8X low part of long %.8X",
        s3.high_part,s3.low_part);
    //---
    s4.s="some constant string";
    s3=s4;
    printf("3. buffer len is %d constant string address is 0x%.8X",
        s3.low_part,s3.high_part);
}
```

其他示例列举出从颜色类型表达里接收RGB（红，绿，蓝），组织自定义函数的方法。创建不同内容，同样大小的

两个结构。为了方便，我们加入一个RGB颜色表达式函数作为字符串。

```
#property script_show_inputs
input color          testColor=clrBlue;// 为测试设置颜色
//--- 以RGB表示颜色的结构
struct RGB
{
    uchar          blue;          // 蓝色组件
    uchar          green;        // 绿色组件
    uchar          red;           // 红色组件
    uchar          empty;        // 不使用这个字节
    string          toString();   // 为接收字符串的函数
};
//--- 显示颜色为字符串的函数
string RGB::toString(void)
{
    string out="("+(string)red+": "+(string)green+": "+(string)blue+") ";
    return out;
}
//--- 存储嵌入颜色类型的结构
struct builtColor
{
    color          c;
};
//+-----+
//| 脚本程序启动函数                                     |
//+-----+
void OnStart()
{
    //--- RGB格式存储的变量
    RGB colorRGB;
    //--- 存储颜色类型的变量
    builtColor test;
    test.c=testColor;
    //--- 通过复制内容转换2个结构
    colorRGB=test;
    Print("color ",test.c,"=",colorRGB.toString());
    //---
}
```

基本类指针到派生类指针的类型转换

[打开生成](#)分类目标也可以看做相关基本类目标。这将导致一些有趣的影响。例如，即使一个基本类生成的不同类目标彼此明显不同，我们仍然可以创建它们的链接列表(List) ，因为我们将它们全部视为基本类型的目标。但是反过来却不可以：基本类目标不能自动成为派生类的目标。

您可以使用明确转换，将基本类指针转化成派生类[指针](#)。但是对这种转化要有足够的资格，因为否则的话，会导致危险的运行错误而mq15程序会停止。

另见

数据类型

空型和NULL常量

依照语法来说，[空型](#)是一种基本类型，附带图表、非图表、布尔型、短整型、无符号短整型、整型、无符号整型、长整型、无符号长整型、日期时间型、浮点型、双精度型和字符串型。类型被指示出来，该功能不返回任何值，而一个功能参数表示一个出席参数。

预先参数变量NULL是空型值，能无需转换标识任何其他基本类型的常量，允许比较基本类型变量与NULL值。

示例：

```
//--- 如果字符串没有初始化，那么会分配预先定义的值给它  
if(some_string==NULL) some_string="empty";
```

NULL也能与指示目标中的 [新操作](#) 对比。

另见

[变量](#)， [函数](#)

目标指针

在MQL5中，有可能动态地创造复杂形势的目标，被[新操作](#)执行，创造目标返回一个描述符号，描述符号占用8字节。根据语法来说，在MQL6中的描述符号目标中与C++程序类似。

示例：

```
MyObject* hobject= new MyObject();
```

与C++不同的是，上述示例变量hobject并不指向内存，只是目标的描述符号。

另见

[变量](#)， [初始变量](#)， [变量可见范围和使用期限](#)， [创建和删除目标](#)

引用，修饰符&和关键字this

通过引用传送参数

在MQL5的[简单](#)类型参数能够通过值和引用传送，而[复合](#)类型参数经常通过引用传送。若使编译器了解参数是通过引用传送，需要在参数前加上 & 符号。

通过引用传送参数意味着传送变量地址，这就是通过引用传送的参数的所有改变能够马上在变量源中反映出来的原因。使用通过引用传送的参数，一个函数可以同时得出几种结果。为防止引用传送的参数改变，使用[常量](#)修饰符。

因此，函数输入参数是一个[数组](#)，结构或者类，符号&'放在函数表头，变量类型后，名字前。

示例：

```
class CDemoClass
{
private:
    double          m_array[];

public:
    void            setArray(double &array[]);
};
//+-----+
//|  填充数组  |
//+-----+
void CDemoClass::setArray(double &array[])
{
    if(ArraySize(array)>0)
    {
        ArrayResize(m_array,ArraySize(array));
        ArrayCopy(m_array, array);
    }
}
```

上面示例中，包括[私人](#)成员的类CDemoClass被声明-[双精度](#)型数组 m_array[]。函数 setArray() 被声明，通过引用传送array[]到这里。如果函数表头不包括通过引用传送的指示，例如，不包括&符号，并且编译这个符号时会生成错误信息。

即使数组是通过引用传送的，我们仍不能把一个数组定义到另一个。我们需要将数组源里逐元素复制成可接受的数组。函数描述中&是数组和结构传送成函数参数必要的状态。

关键字 this

类型 (object) 变量可以通过引用和[指针](#)传送。如引用一样，指针允许访问目标。目标指针声明后，就会应用[新](#)操作去创建和初始化。

关键字this意在获得目标的引用，可以在类或者结构函数里得到。this经常引用到目标中，并在此函数中使用，表达式 [GetPointer\(this\)](#) 给出目标指针，其成员为完成GetPointer() 的函数。在MQL5中，函数不能返回对象，但是它们能返回目标指针。

因此，如果我们需函数返回对象，我们能够在GetPointer(this) 形式下返回目标指针。我们将这个类的目标指针函数getDemoClass() 添加到CDemoClass描述。

```

class CDemoClass
{
private:
    double          m_array[];

public:
    void            setArray(double &array[]);
    CDemoClass      *getDemoClass();
};
//+-----+
//|  填充数组                                     |
//+-----+
void CDemoClass::setArray(double &array[])
{
    if(ArraySize(array)>0)
    {
        ArrayResize(m_array,ArraySize(array));
        ArrayCopy(m_array,array);
    }
}
//+-----+
//|  返回它自己的指针                             |
//+-----+
CDemoClass *CDemoClass::getDemoClass(void)
{
    return(GetPointer(this));
}

```

结构没有指针，*新建和删除*操作不能应用其上，不可以用GetPointer(this)。

另见

[目标指针](#)，[创建和删除对象](#)，[可见范围和变量使用期](#)

运行式和表达式

一些数字和字符的组合是特别重要的，它们被称为运算符，例如：

+ - * / %	算术运算符
&&	逻辑运算符
= += *=	负值运算符

运算符应用在表达式中实现特定的作用。需要特别注意标点符号如圆括号、方括号、逗号、冒号、分号。

运算符，标点符号，空格用于分割语句的不同部分。

这部分包括以下主题：

- [表达式](#)
- [算术运算](#)
- [指定运算](#)
- [关系运算](#)
- [布尔运算](#)
- [逐位运算](#)
- [其他运算](#)
- [优先运算命令](#)

表达式

一个表达式可以拥有多个字符和操作符。一个表达式可以写在几行里面。

示例：

```
a++; b = 10;           // 几个表达式位于一行
// --- 一个表达式被分成几行
x = (y * z) /
    (w + 2) + 127;
```

一个表达式的最后一个分号 (;) 操作符。

另见

[优先规则](#)

算术运算

算术运算符包括加法和乘法运算：

Sum of variables	<code>i = j + 2;</code>
Difference of variables	<code>i = j - 3;</code>
Changing the sign	<code>x = - x;</code>
Product of variables	<code>z = 3 * x;</code>
Division quotient	<code>i = j / 5;</code>
Divisions reminder	<code>minutes = time % 60;</code>
Adding 1 to the variable value	<code>i++;</code>
Adding 1 to the variable value	<code>++i;</code>
Subtracting 1 from the variable value	<code>k--;</code>
Subtracting 1 from the variable value	<code>--k;</code>

加减运算只用于变量中，不能用于常量里。加减前缀（`++i`）和（`-k`）只在变量用在表达式前才用于这个变量。

加减后缀（`i++`）和（`k-`）只在变量用在表达式后才用于这个变量。

示例：

```
int a=3;
a++;           // 有效表达式
int b=(a++)*3; // 无效表达式
```

另见

[优先规则](#)

赋值运算

包括给出运算式的表达式值就是赋值后留下的运算对象：

Assignment the x value to the y variable	<code>y = x;</code>
--	---------------------

下面的运算式将赋值运算式与算法和逐位运算相结合：

Adding x to the y variable	<code>y += x;</code>
Subtracting x from the y variable	<code>y -= x;</code>
Multiplying the y variable by x	<code>y *= x;</code>
Dividing the y variable by x	<code>y /= x;</code>
Reminder of division of the y variable by x	<code>y %= x;</code>
The shift of the binary representation of y to the right by x bit	<code>y >>= x;</code>
The shift of the binary representation of y to the left by x bit y	<code>y <<= x;</code>
AND bitwise operation of binary representations of y and x	<code>y &= x;</code>
OR bitwise operation of binary representations of y and x	<code>y = x;</code>
Excluding OR bitwise operation of binary representations of y and x	<code>y ^= x;</code>

逐位算法只能用于整数。当y与右/左x完成逻辑移位时，x使用最小的二进制值5，放弃最大值，例如0-31间的移位。

By `%=` 运算式 (x与y组件) ，结果等于分开数字符。

赋值操作符在一个表达式中可以使用多次。这种情况下，表达式从左到右执行：

<code>y=x=3;</code>

首先，x变量赋值3，y变量赋予x值，也是3。

另见

[优先规则](#)

关系运算

布尔运算 FALSE 代表整数零值，而布尔运算 TRUE 代表不同于零的任何值。

用返回 FALSE (0) 或者 TRUE (1) 来表示[逻辑值](#) 两个量之间的关系。

等于b	<code>a == b;</code>
不等于b	<code>a != b;</code>
小于b	<code>a < b;</code>
大于b	<code>a > b;</code>
小于等于b	<code>a <= b;</code>
大于等于b	<code>a >= b;</code>

相等的两个[真实数字](#)不能比较。大部分情况下，两个看起来相似的数字不能相同，因为15小数位的值不同。为了正确比较两个真实数字，用零比较这些不同。

示例：

```
bool CompareDoubles(double number1, double number2)
{
    if(NormalizeDouble(number1-number2, 8) == 0) return(true);
    else return(false);
}

void OnStart()
{
    double first=0.3;
    double second=3.0;
    double third=second-2.7;
    if(first!=third)
    {
        if(CompareDoubles(first, third))
            printf("%.16f and %.16f are equal", first, third);
    }
}

// 结果: 0.3000000000000000 0.2999999999999998 是平等的
```

另见

[优先规则](#)

布尔运算

否定运算符 (!)

否定运算符(!)，用来表示真假的反面的结果。如果运算值是 FALSE (0) 结果为TRUE (1)；如果运算不同于FALSE (0) 等于FALSE (0)。

```
if(!a) Print("不是 'a'");
```

逻辑运算符或 OR (||)

x和y值的逻辑运算符或OR(||)用来表示两个表达式只要有一个成立即可。如果x和y值为真的，表达式值为TRUE (1)。否则，值为FALSE (0)。

```
if(x<0 || x>=max_bars) Print("超出范围");
```

逻辑运算符 AND (&&)

x和y值的逻辑运算符 AND (&&)。如果x和y值为真的(not null)，表达式值为TRUE (1)。否则，值为FALSE (0)。

布尔运算的摘要评估

所谓的“摘要评估”模式是应用到布尔型操作系统中的，i.e.当表达式可以被精确的终止时，该表达计算就会停止。

```
//+-----+
//| 脚本程序启动函数 |
//+-----+
void OnStart()
{
//--- 简单判断的第一示例
    if(func_false() && func_true())
    {
        Print("Operation &&: You will never see this expression");
    }
    else
    {
        Print("Operation &&: Result of the first expression is false, so the second was");
    }
//--- 简单判断的第二示例
    if(!func_false() || !func_true())
    {
        Print("Operation ||: Result of the first expression is true, so the second wasn");
    }
    else
    {
        Print("Operation ||: You will never see this expression");
    }
}
//+-----+
```



```
///| 函数总是返回false |
//+-----+
bool func_false()
{
    Print("Function func_false()");
    return(false);
}
//+-----+
///| 函数总是返回 true |
//+-----+
bool func_true()
{
    Print("Function func_true()");
    return(true);
}
```

另见

[优先规则](#)

逐位运算

补码

补充变量值，表达值包含1，可变量值包含0，表达值包含0，可变量值包含1。

```
b = ~n;
```

示例：

```
char a='a',b;  
b=~a;  
Print("a = ",a, " b = ",b);  
// 结果将会是：  
// a = 97   b = -98
```

右移

运算符x向右移动到数字y代表二进制代码。如果移动的值是无符号类型，进行逻辑右移，即左侧将被零填满。

如果移动的值是符号类型，进行算术右移，即左侧将被符号填满（如果数字是正值，符号为零值；如果数字为负值，符号值为1）。

```
x = x >> y;
```

示例：

```
char a='a',b='b';  
Print("Before: a = ",a, " b = ",b);  
// --- 右移  
b=a>>1;  
Print("After: a = ",a, " b = ",b);  
// 结果会是：  
// 之前: a = 97   b = 98  
// 以后: a = 97   b = 48
```

左移

运算符x向左移动到数字y代表二进制代码，即右侧将被零填满。

```
x = x << y;
```

示例：

```
char a='a',b='b';  
Print("Before: a = ",a, " b = ",b);  
// --- 左移  
b=a<<1;  
Print("After: a = ",a, " b = ",b);  
// 结果会是：  
// 之前: a = 97   b = 98  
// 之后: a = 97   b = -62
```

不建议移动大的数字或者与移动变量长度相等的数字，因为运算结果无法确定。

位逻辑运算符 AND

二进制的x和y代表位逻辑运算符AND。在所有数组中x和y的值都不含有零表达式的值包含1(TRUE) ；在所有其他数字中包含 0 (FALSE) 。

```
b = (x & y) != 0;
```

示例：

```
char a='a',b='b';
//--- AND 操作
char c=a&b;
Print("a = ",a," b = ",b);
Print("a & b = ",c);
// 结果会是：
// a = 97   b = 98
// a & b = 96
```

位逻辑运算符OR

二进制的x和y代表位逻辑运算符OR。在所有数字中x和y的值都不等于零表达值包含1 并且在所有其他数字中包含0。

```
b = x | y;
```

示例：

```
char a='a',b='b';
//--- 或者操作
char c=a|b;
Print("a = ",a," b = ",b);
Print("a | b = ",c);
// 结果将会是：
// a = 97   b = 98
// a | b = 99
```

位逻辑运算符Exclusive

二进制的x和y代表位逻辑运算符EXCLUSIVE (eXclusive OR) 。在所有数字中x和y的值都不同于二进制值表达值包含1并且在所有其他数字中包含0。

```
b = x ^ y;
```

示例：

```
char a='a', b='b';
//--- 免除或者执行
char c=a^b;
Print("a = ",a," b = ",b);
Print("a ^ b = ",c);
// 结果会是：
```

```
// a = 97    b = 98  
// a ^ b = 99
```

位逻辑运算符只作用于[integers](#)。

另见

[优先规则](#)

其他运算

指数 ([])

在数组第一元素的位置，表达式值为i的系列数变量值。

示例：

```
array[i] = 3; // 数组的3的计算值到第i个元素。
```

只有整数能够成为数组指数。四维以下的数组是禁止的。每组的检测是从0到测量大小-1。特定情况下，对于维数组由50个元素组成，参照的第一个数组将为[0]，这样最后一个数组将是[49]。

访问超出数组，正在执行的子系统生成危险错误，程序停止。

调用x1, x2 ,..., xn自变数函数

每一个自变数可以显示一个常数，一个变量和相应类型表达式。自变数的通过必须根据通道命令。

用此函数返回表达式值。如果返回的表达式值为空，一些函数不能进行中转。请确认表达式x1,x2,...,xn 是按照命令执行的。

示例：

```
int length=1000000;
string a="a",b="b",c;
//---
int start=GetTickCount(),stop;
long i;
for(i=0;i<length;i++)
{
    c=a+b;
}
stop=GetTickCount();
Print("time for 'c = a + b' = ",(stop-start)," milliseconds, i = ",i);
```

逗号操作符 (,)

从左到右的表达式用逗号分开。所有表达式的计算是从左至右的。结果类型和值相互吻合，说明表达式是正确的。参量列表可以作为范例被通过（参阅上文）。

示例：

```
for(i=0,j=99; i<100; i++,j--) Print(数组[i][j]);
```

点操作符 (.)

对于直接访问结构和类的[公共成员](#)时，使用点操作符。语法：

```
Variable_name_of_structure_type.Member_name
```

示例：

```
struct SessionTime
{
    string sessionName;
    int    startHour;
    int    startMinutes;
    int    endHour;
    int    endMinutes;
} st;
st.sessionName="Asian";
st.startHour=0;
st.startMinutes=0;
st.endHour=9;
st.endMinutes=0;
```

范围解析操作符 (::)

mql5中每个函数都有其执行范围。例如，[Print\(\)](#)系统函数可以在全局范围使用。[Imported](#)函数导入时使用。[classes](#)函数用在类的范围。范围解析操作语法如下：

```
[Scope_name]::Function_name(parameters)
```

如果没有范围名，就属于全局范围函数。如果没有范围解析操作，可以在最近范围内找到函数。如果本地范围没有函数n，需要全局搜寻函数。

范围解析操作也用在 [确定函数](#)-类成员。

```
type Calss_name::Function_name(parameters_description)
{
    // function body
}
```

示例：

```
#property script_show_inputs
#import "kernel32.dll"
    int GetLastError(void);
#import

class CCheckContext
{
    int    m_id;
public:
    CCheckContext() { m_id=1234; }
protected:
    int    GetLastError() { return(m_id); }
};
class CCheckContext2 : public CCheckContext
{
    int    m_id2;
```

```

public:
    CCheckContext2() { m_id2=5678; }

    void Print();
protected:
    int GetLastError() { return(m_id2); }
};

void CCheckContext2::Print()
{
    ::Print("Terminal GetLastError", ::GetLastError());
    ::Print("kernel32 GetLastError", kernel32::GetLastError());
    ::Print("parent GetLastError", CCheckContext::GetLastError());
    ::Print("our GetLastError", GetLastError());
}

//+-----+
//| 脚本程序启动函数 |
//+-----+
void OnStart()
{
    //---
    CCheckContext2 test;
    test.Print();
}

//+-----+

```

获得数据类型大小或者任何类型数据对象大小的运算 (sizeof)

使用sizeof运算，标识符或者类型大小可以确定。sizeof 运算有以下格式：

示例：

```
sizeof(expression)
```

任何标识符，或者类型名都在括号中，可用作表达式。注意空类型名不可以使用，标识符不属于二进制，也不是函数名。

如果表达式为静态数组名（例如给出第一尺寸），那么结果就是整个数组大小（例如类型长度产品）。如果表达式是动态数组名（没有确定第一尺寸），结果就是[动态数组](#)对象大小。

当sizeof用在结构或者类型名时，或者结构或者类型标识符时，结果就是结构或者类的真实大小。

示例：

```

struct myStruct
{
    char   h;
    int    b;
    double f;
} str;

Print("sizeof(str) = ", sizeof(str));
Print("sizeof(myStruct) = ", sizeof(myStruct));

```

在编辑步骤计算大小。

另见

[优先规则](#)

优先规则

下面是从上到下的运算优先规则，优先级高的将先被运算。

注意：MQL5语言运算优先依据C++优先规则，不同于MQL4语言。

运算	描述	执行顺序
() [] .	函数调用 数组元素参考 引用结构元素	从左到右
! ~ - ++ -- (type) sizeof	真假运算符 位逻辑运算符（补码） 改变运算符 增量 减量 类型转换 决定字节大小	从右到左
* / %	乘法 除法 百分比	从左到右
+ -	加法 减法	从左到右
<< >>	左移 右移	从左到右
< <= > >=	小于 小于等于 大于 大于等于	从左到右
== !=	等于 不等于	从左到右
&	位逻辑运算符AND	从左到右
^	位逻辑运算符 OR	从左到右
	位逻辑运算符 OR	从左到右
&&	逻辑AND	从左到右
	逻辑OR	从左到右
?:	假设运算	从右到左
= *= /= %= += -=	值 乘法值 除法值 百分比值 加法值 减法值	从右到左

<<=	左移值	
>>=	右移值	
&=	位逻辑运算符AND值	
^=	位逻辑运算符OR值	
=	位逻辑运算符OR值	
,	逗号	从左到右

若要改变运算操作顺序，使用更高一级的圆括号。

操作符

语言操作符必须对执行完成任务的一些运算法操作进行描述。程序本身是这样的序列语句。语句逐个随后以分号分离。

操作符	描述
复合操作符 { }	花括号 { } 括起来的一个或者多个任一类型操作符
表达操作符 (;)	分号 (;) 结尾的表达式
返回 操作符	终止当前函数和返回控制访问程序
if-else 假设操作符	需要作出选择时使用
?: 假设操作符	if-else假设操作符的简单类似物
切换 选择操作符	控制与表达式值一致的操作符
while 循环操作符	执行操作符直到表达式变成错误。每次重复前都要检查表达式。
for 循环操作符	执行操作符直到表达式变成错误。每次重复前都要检查表达式。
do-while 循环操作符	执行操作符直到表达式变成错误。每次循环后进行检测。循环主体至少执行一次。
嵌入 操作符	终止执行最近的外部操作符， while， do-while 或者 for
继续 操作符	控制最近的外部循环操作符 while， do-while 或者for的起点。
new 操作符	创建大小合适的对象以及返回创建对象的描述符。
delete 操作符	删除新操作符创建的对象

一个语句能占领一条或几条线。二个或更多语句可能位于同样线。控制执行顺序的语句(if, if-else, switch, while and for) 可以相互插入。

示例：

```
if(Month() == 12)
    if(Day() == 31) Print("Happy New Year!");
```

另见

[变量初始化](#)， [可见范围和变量使用期](#)， [创建和删除对象](#)

复合操作符

一个复合操作符 (a block) 有一个(一个区段) 和由一个或多个任何类型的操作符组成的的附件{ } . 每个表达式使用分号作为结束(;) 。

示例：

```
if(x==0)
{
    Print("无效位置 x = ",x);
    return;
}
```

另见

[变量初始化](#) , [可见范围和变量使用期](#) , [创建和删除对象](#)

表达式操作符

任何以分号(;) 结束的表达式都被视为是一个操作符。这里是一些表达式操作符得范例:

赋值操作符

标识符 =表达式 ;

```
x=3;  
y=x=3;  
bool equal=(x==y);
```

赋值操作符在表达式操作符中只限一次使用。

函数调用操作符

Function_name (argument1,..., argumentN) ;

```
FileClose(file);
```

空操作符

它是由分号(;) 组成并且使用在一个检测操作符中。

另见

[变量初始化](#) , [可见范围和变量使用期](#) , [创建和删除对象](#)

返回操作符

返回操作符终止当前[函数](#)操作，返回控制访问程序。表达式计算结果返回调用函数。表达式可以包括返回操作符。

示例：

```
int CalcSum(int x, int y)
{
    return(x+y);
}
```

函数中，[空](#)返回类型函数，必须使用无表达式的[return](#)操作符：

```
void SomeFunction()
{
    Print("Hello!");
    return;    // 操作符可以被移除
}
```

函数右括号意思是无表达式返回操作符的隐性执行。

能返回的有：[简单类型](#)，[简单结构](#)，[目标指针](#)。[返回](#)操作符您不能返回任何数组，类对象，复合结构类型变量。

另见

[变量初始化](#)，[可见范围和变量使用期](#)，[创建和删除对象](#)

If-Else 假设操作符

需要选择时使用IF - ELSE 操作符。语法形式如下：

```
if (expression)
    operator1
else
    operator2
```

如果表达式是真实的，操作符1执行，并按照操作符2给出的检测（操作符2不执行）。如果表达式是错误的，操作符2被执行。

if操作符else部分可能被忽略。if操作符忽略else部分，显示分歧可能会嵌入。这种情况下，else位置在先前if操作符的最近部位，这样不会出现else部分。

示例：

```
// --- else 部分提及到第二个if操作符：
if(x>1)
    if(y==2) z=5;
else      z=6;
// --- else部分提及到第一个if操作符：
if(x>1)
{
    if(y==2) z=5;
}
else      z=6;
// --- 嵌入操作符
if(x=='a')
{
    y=1;
}
else if(x=='b')
{
    y=2;
    z=3;
}
else if(x=='c')
{
    y=4;
}
else Print("ERROR");
```

另见

[变量初始化](#)， [可见范围和变量使用期](#)， [创建和删除对象](#)

假设操作符?:

三进制操作符一般形式如下：

```
expression1? expression2:expression3
```

对于第一个操作-"expression1"-任何表达式在布尔型中值都可以用。如果结果是true，那么第二个操作的操作符，例如 "expression2"，就被执行。

如果第一操作是false，第三操作-"expression3"-就被执行。第二和第三操作，例如 "expression2"和"expression3"应该返回第一类型值，不应该是空类型。执行假设操作符的结果就是expression2的结果或者expression3的结果，都取决于 expression1。

```
// --- 一天中开盘价和收盘价的标准化不同点
double true_range = (High==Low)?0:(Close-Open)/(High-Low);
```

输入等同于如下：

```
double true_range;
if (High==Low) true_range=0;           // 如果最高价等于
else true_range=(Close-Open)/(High-Low); // 如果范围无效
```

示例：

```
void func(double d) { Print("double argument: ",d); }
void func(string s) { Print("string argument: ",s); }

bool Expression1=true;
double Expression2=M_PI;
string Expression3="3.1415926";

void OnStart()
{
    func(Expression2);
    func(Expression3);

    func(Expression1?Expression2:Expression3); // получим предупреждение компилятора
    func(!Expression1?Expression2:Expression3); // получим предупреждение компилятора
}

// Результат:
```



```
// double argument: 3.141592653589793  
// string argument: 3.1415926  
// string argument: 3.141592653589793  
// string argument: 3.1415926
```

另见

[变量初始化](#)， [可见范围和变量使用期](#)， [创建和删除对象](#)

切换操作符

在case全部变量和相应表达式值检测的操作符之内比较常数表达式值。每一个case变量会在[整数常数](#)表达式内标注。常数表达式不包含函数变量调用。switch表达式操作符必须是整数类型。

```
switch(expression)
{
    case constant: operators
    case constant: operators
    ...
    default: operators
}
```

如果在case操作符等于表达式值，操作符default标签连接将会执行。此default变量无需在最后。如果相应表达式值和default变量没有获取，不会有任何执行。

关键词case和常数被标注，并且if 操作符执行 case变量，程序将执行以下所有操作符直至[break](#)操作符生成。

一个常数表达式的计算是在编译期间。在一个switch操作符内部存在两个相同值的常数。

示例：

```
//--- 第一示例
switch(x)
{
    case 'A':
        Print("CASE A");
        break;
    case 'B':
    case 'C':
        Print("CASE B or C");
        break;
    default:
        Print("NOT A, B or C");
        break;
}

//--- 第二示例
string res="";
int i=0;
switch(i)
{
    case 1:
        res=i;break;
    default:
        res="default";break;
    case 2:
        res=i;break;
    case 3:
        res=i;break;
```

```
    }  
    Print(res);  
    /*  
    结果  
    默认  
    */
```

另见

[变量初始化](#)， [可见范围和变量使用期](#)， [创建和删除对象](#)

循环操作符While

while操作符是由一个已检测的表达式和操作符构成，需要满足以下条件：

```
while (expression)
    operator;
```

如果表达式为true，操作符执行直至表达式变成false。如果表达式为false，将检测最近操作符。在操作符执行前，一个表达式值已经被指定。不过，如果开始表达式为 false，操作符根本不会执行。

[IsStopped\(\)](#).

示例

```
while (k<n  && !IsStopped())
{
    y=y*x;
    k++;
}
```

另见

[变量初始化](#)， [可见范围和变量使用期](#)， [创建和删除对象](#)

循环操作符For

for操作符由三个表达式和一个执行操作符组成：

```
for(expression1; expression2; expression3)
    operator;
```

用表达式1Expression来定义初始变量,当表达式2Expression2 为真的时候执行操作运算符for,在每次循环结束后执行表达式3 Expression3。如果true, 运算符 for 将被执行。循环重复直至Expression2变为false。如果false, 循环将会被中断并且检测运算符文本。稍候执行。

此 for运算符下列运算符成功：

```
expression1;
while(expression2)
{
    operator;
    expression3;
};
```

for操作符中可以缺少任何三个或者全部三个表达式，但是分隔它们的分号(;) 必须省略。如果表达式2省略，则意味着不变的true。for(;;) 操作符是持续循环，与 while(1) 操作符一样。每一个表达式1或者3都由几个逗号','组成的表达式构成。

[IsStopped\(\)](#).

示例：

```
for(x=1;x<=7000; x++)
{
    if(IsStopped())
        break;
    Print(MathPower(x,2));
}
//--- 另一个示例
for(;!IsStopped();)
{
    Print(MathPower(x,2));
    x++;
    if(x>10) break;
}
//--- 第三示例
for(i=0,j=n-1;i<n && !IsStopped();i++,j--) a[i]=a[j];
```

另见

[变量初始化](#)， [可见范围和变量使用期](#)， [创建和删除对象](#)

循环操作符 do while

[for](#) 和 [while](#) 从起点循环检测终止，不在循环末端。第三种循环操作符 [do - while](#) 每次循环重复后，在最后检测终止状态。循环主体至少执行一次。

```
do
    operator;
while (expression)
```

首先执行操作符，然后计算表达式。如果是true，那么操作符再次执行。如果表达式变成false，循环终止。

[IsStopped\(\)](#).

示例：

```
// -- 计算斐波纳契数列
int counterFibonacci=15;
int i=0, first=0, second=1;
int currentFibonacciNumber;
do
{
    currentFibonacciNumber=first+second;
    Print("i = ", i, "   currentFibonacciNumber = ", currentFibonacciNumber);
    first=second;
    second=currentFibonacciNumber;
    i++; // 没有这个操作符会出现一个无限循环!
}
while(i<counterFibonacci && !IsStopped());
```

另见

[变量初始化](#)， [可见范围和变量使用期](#)， [创建和删除对象](#)

嵌入操作符

`break`操作符终止最近外部嵌入操作符 [switch](#), [while](#), [do-while](#) 或者 [for](#) 的执行。在终止操作符之后给出检测操作符。这个操作符的目的之一：当中心值指定为变量时，操作符完成循环执行。

示例：

```
// --- 搜索第一个零元素  
for(i=0;i<array_size;i++)  
    if(array[i]==0)  
        break;
```

另见

[变量初始化](#) , [可见范围和变量使用期](#) , [创建和删除对象](#)

继续操作符

一个[继续](#)操作符。我们将其放在嵌套内的指定位置,用来在指定情况下跳过接下来的运算,直接跳入下一次的循环 [while](#), [do-while](#) 或者 [for](#)操作符。操作符[break](#)位置与此操作符相反。

示例：

```
//--- 所有非零元素总和
int func(int array[])
{
    int array_size=ArraySize(array);
    int sum=0;
    for(int i=0;i<array_size; i++)
    {
        if(a[i]==0) continue;
        sum+=a[i];
    }
    return(sum);
}
```

另见

[变量初始化](#), [可见范围和变量使用期](#), [创建和删除对象](#)

对象创建操作符new

new 操作符自动创建一个相应大小的对象，称为对象构造函数并回转[已经创建的对象描述符](#)。失败的情况下，操作符返回一个与常量**NULL**相比较的null描述符。

new操作符仅能用于[类](#)对象。不能应用于结构。

操作符不用于创建对象数组。若要做这个，使用 [ArrayResize\(\)](#)。

示例：

```
//+-----+
//| 图形创建                                     |
//+-----+
void CTetrisField::NewShape()
{
    m_ypos=HORZ_BORDER;
//--- 随机创建7个可能形状中的一个
    int nshape=rand()%7;
    switch(nshape)
    {
        case 0: m_shape=new CTetrisShape1; break;
        case 1: m_shape=new CTetrisShape2; break;
        case 2: m_shape=new CTetrisShape3; break;
        case 3: m_shape=new CTetrisShape4; break;
        case 4: m_shape=new CTetrisShape5; break;
        case 5: m_shape=new CTetrisShape6; break;
        case 6: m_shape=new CTetrisShape7; break;
    }
//--- 绘画
    if(m_shape!=NULL)
    {
        //--- 预置
        m_shape.SetRightBorder(WIDTH_IN_PIXELS+VERT_BORDER);
        m_shape.SetYPos(m_ypos);
        m_shape.SetXPos(VERT_BORDER+SHAPE_SIZE*8);
        //--- 绘画
        m_shape.Draw();
    }
//---
}
```

注意对象描述符不是内存指标。

用new操作符创建的对象不能用[delete](#)操作符直接移动。

另见

[变量初始化](#)， [可见范围和变量使用期](#)， [创建和删除对象](#)

对象删除操作符 delete

`delete`操作符删除通过[new](#)操作符创建的对象，称为相关的类析构函数并释放由对象占据的内存。现存对象的真实析构函数用作操作对象。`delete`操作执行后，[对象析构函数无效](#)。

示例：

```
//--- 删除图形
delete m_shape;
m_shape=NULL;
//--- 创建一个新图形
NewShape();
```

另见

[变量初始化](#)， [可见范围和变量使用期](#)， [创建和删除对象](#)

函数

每个任务都可以分解成子任务，既可以用代码形式直接表示，也可以分成子任务。这个方法称为逐步求精。函数用来输入要解决的子任务的代码。描述函数的代码称为函数定义。

```
function_header
{
    instructions
}
```

第一个括号前是函数定义的表头，括号中间是函数定义主体。函数表头包括返回值类型描述，名称(标识符)和正式参数。通过函数的参数有限制，不能超过64。

函数不能从程序的其他部分多次调用。实际上，返回类型，函数标识符和参数类型组成函数原型。

函数原型是函数说明，但是不是定义。由于明确说明返回类型和自变量类型列表，调用函数时严格检测类型和隐藏的类型转换。经常使用函数说明提高代码可读性。

函数定义必须与其说明相匹配。每个说明的函数都要下定义。

示例：

```
double                // 返回值类型
linfunc (double a, double b) // 函数名和参量列表
{
    // 组合操作符
    return (a + b);      // 返回值
}
```

返回操作符能够返回位于这个操作符的表达式值。如果有必要，表达式值可以转换为函数结果类型。返回到零值的函数描述为空型。

示例：

```
void errmesg(string s)
{
    Print("error: "+s);
}
```

通过函数的参量可能存在由特定类型常数指定的默认值。

示例：

```
int somefunc(double a,
             double d=0.0001,
             int n=5,
             bool b=true,
             string s="passed string")
{
    Print("Required parameter a = ",a);
    Print("Pass the following parameters: d = ",d," n = ",n," b = ",b," s = ",s);
}
```

```
    return(0);  
}
```

如果任何参数有默认值，所有的子参数也一定有默认值。

错误说明示例：

```
int somefunc(double a,  
             double d=0.0001,    // 声明默认值  
             int n,              // 没有指定默认值 !  
             bool b,            // 没有指定默认值 !  
             string s="passed string")  
{  
}
```

另见

[重载](#), [虚拟函数](#), [多态](#)

调用函数

如果显示的文件没有描述，它将考虑上下文的联系作为函数名称。

```
function_name (x1, x2, ..., xn)
```

自变数（[形式参数](#)）以值的形式通过。计算每一个表达式 x_1, \dots, x_n 并将其值发送到函数。表达式计算命令值是被保证的。在执行系统测试数字和自变数类型期间会给出函数。这种形式的函数调用被称作调用值。

调用函数是一个通过函数返回的表达式值。描述函数类型必须相应类型返回的值。[全球范围](#)内程序的任何一个部分函数是被声明或描述的，即其他函数之外。在另外函数里，函数不能被声明或描述。

示例：

```
int start()
{
    double some_array[4]={0.3, 1.4, 2.5, 3.6};
    double a=linfunc(some_array, 10.5, 8);
    //...
}
double linfunc(double x[], double a, double b)
{
    return (a*x[0] + b);
}
```

函数的调用是默认参量，通过参量的列表是被限定的，但不是之前的第一默认参量。

示例：

```
void somefunc(double init,
              double sec=0.0001, // 设置默认值
              int level=10);
//...
somefunc();           // 错误调用。必须存在第一参量
somefunc(3.14);       // 正确调用
somefunc(3.14,0.0002); // 正确调用
somefunc(3.14,0.0002,10); // 正确调用
```

当我们调用一个函数时，不可以忽略参量，存在默认值：

```
somefunc(3.14, , 10); // 错误调用 -> 跳过第二参量
```

另见

[重载](#), [虚拟函数](#), [多态](#)

传递函数

有两类函数，计算机语言可以用其传送自变量到子程序（函数）。第一类函数是通过值发送参数。这个函数将自变量值复制到形式函数参数。因此，函数中这个参数的任何改变都不会对自变量的相应调用产生影响。

```
//+-----+
//| 通过值传递参数 |
//+-----+
double FirstMethod(int i,int j)
{
    double res;
//---
    i*=2;
    j/=2;
    res=i+j;
//---
    return(res);
}
//+-----+
//| 脚本程序启动函数 |
//+-----+
void OnStart()
{
//---
    int a=14,b=8;
    Print("a and b before call:",a," ",b);
    double d=FirstMethod(a,b);
    Print("a and b after call:",a," ",b);
}
//--- 执行脚本结果
// 调用前a 和 b : 14 8
// 调用后a 和 b : 14 8
```

第二类函数是通过引用传送。在这种情况下，参数引用（不是它的值）被传送到函数参数。函数内，被用来连接调用指定的实际参数。这意味着参数的改变将会影响用来调用函数的自变量。

```
//+-----+
//| 通过引用传递参数 |
//+-----+
double SecondMethod(int &i,int &j)
{
    double res;
//---
    i*=2;
    j/=2;
    res=i+j;
//---
    return(res);
}
```

```
//+-----+
//| 脚本程序启动函数 |
//+-----+
void OnStart()
{
//---
    int a=14,b=8;
    Print("a and b before call:",a," ",b);
    double d=SecondMethod(a,b);
    Print("a and b after call:",a," ",b);
}
//+-----+
//--- 执行脚本结果
// 调用前a 和 b: 14 8
// 调用后a 和 b: 28 4
```

MQL5使用这两种函数，有一个例外就是：数组和结构类型变量（类的对象）总是通过引用传送。为了在真实参数中排除改变（函数调用传送的自变量）使用[常数](#)访问说明符。若尝试改变说明常量说明符的变量，编译程序会生成错误。

注解

注意用倒序传送参数到函数，例如，首先是最近的函数被计算和传送，然后倒数第二。最后计算和传送的参数是括号后的第一个。

示例：

```
void OnStart()
{
//---
    int a[]={0,1,2};
    int i=0;

    func(a[i],a[i++],"First call (i = "+string(i)+" )");
    func(a[i++],a[i],"Second call (i = "+string(i)+" )");
// 结果:
// 首先调用 (i = 0) : par1 = 1    par2 = 0
// 第二调用 (i = 1) : par1 = 1    par2 = 1

}
//+-----+
//| |
//+-----+
void func(int par1,int par2,string comment)
{
    Print(comment,": par1 = ",par1,"    par2 = ",par2);
}
```

首先上面的示例中先调用变量*i*参与合并字符串。

```
"First call (i = "+string(i)+" )"
```

这里值不可更改。变量 `i` 加入 `a[i++]` 数组的运算，例如加入数组元素 `i`，变量 `i` 就会[增加](#)。之后，仅计算变量 `i` 改变值第一个参数。

第二步，计算所有三个参数，使用调用函数第一步计算的和的同值，仅计算第一参数后变量 `i` 会再次更改。

另见

[可见范围和变量使用期](#)、[重载](#)、[虚拟函数](#)、[多态](#)

重载函数

通常函数名反映它的作用。一般说来，可读程序包括各种挑选出来的标识符。有时不同的函数用于同样的作用。例如，计算双精度数数组的平均值的函数，也会操作整数数组。都可以称为数组平均数AverageFromArray：

```
//+-----+
//| 双精度类型数组的平均数计算      |
//+-----+
double AverageFromArray(const double array[],int size)
{
    if(size<=0) return 0.0;
    double sum=0.0;
    double aver;
//---
    for(int i=0;i<size;i++)
    {
        sum+=array[i];    // 添加到双精度型
    }
    aver=sum/size;        // 正好总数被数字除
//---
    Print("Calculation of the average for an array of double type");
    return aver;
}
//+-----+
//| 整型数组的平均数计算            |
//+-----+
double AverageFromArray(const int array[],int size)
{
    if(size<=0) return 0.0;
    double aver=0.0;
    int sum=0;
//---
    for(int i=0;i<size;i++)
    {
        sum+=array[i];    // 添加到双精度
    }
    aver=(double)sum/size;// 给予双精度类型总值，并且相除
//---
    Print("Calculation of the average for an array of int type");
    return aver;
}
```

每个函数通过 `Print()` 函数输出信息；

```
Print("Calculation of the average for an array of int type");
```

程序选择与自变量类型及其数量一致的必需的函数。选择依据的规则，称为署名匹配算法。署名就是用于说明函数类型的列表。

示例：

```
//+-----+
//| 脚本程序启动函数 |
//+-----+
void OnStart()
{
//---
    int    a[5]={1,2,3,4,5};
    double b[5]={1.1,2.2,3.3,4.4,5.5};
    double int_aver=AverageFromArray(a,5);
    double double_aver=AverageFromArray(b,5);
    Print("int_aver = ",int_aver,"    double_aver = ",double_aver);
}
//--- 脚本结果
// 为整型数组计算平均值
// 为双精度型数组计算平均值
// 整型 平均值= 3.00000000    双精度 平均值= 3.30000000
```

重载就是一个函数定义多值。根据函数接收的自变量类型选择特定的值。基于调用函数时对应的自变量列表选择特定的函数，到函数说明参数列表。

当调用重载的函数，程序必需有一个运算法则选择合适的函数。该算法是依据现有类型的[转化](#)完成这个选择。最佳关联是唯一的。至少对于一个自变量来说，是最好的，对其他相关的自变量也是一样。

以下就是每个自变量匹配的算法。

选择一个重载函数的运算法则

1. 匹配严格（尽可能）。
2. 尝试标准类型提高。
3. 尝试标准类型转换。

标准类型提高好于其他标准转换。提高就是[浮点型](#)向[双精度型](#)转换，[布尔型](#) - [字符型](#) - [短整型](#)或者[计数](#)向[整型](#)转换。类似[整型](#)数组的类型转换也属于类型转换。相似类型有：布尔型，字符型，无符号字符型，这三种都是单字节整数；双字节整数是短整型和无符号短整型；4字节整数是整型，无字符整型，和颜色；长整型，无符号长整型，和日期时间。

当然，匹配严格是最好的。若要达到这样的一致性，就要使用[类型转换](#)。编译程序不能处理含糊状况。因此不能依靠类型的微妙不同点和使重载函数不明确的隐式变换。

如果怀疑，请使用显式转换确保严格相符。

MQL5中重载的函数可以在[ArrayInitialize\(\)](#) 函数示例中看到。

函数重载规则用于[类函数重载](#)。

允许系统函数重载，但是要注意到编译程序可以准确的选择必要函数。例如，用三种不同方法重载系统函数 [fmax\(\)](#)，但是仅有两个变量是正确的。

示例：

```
// 允许重载 - 在参数数字上不同
double fmax(double a,double b,double c);
```

```
// 错误重载
// 参数是不同的，但是最后一个有默认值
// 这就导致系统函数的隐藏，不可接受调用
double fmax(double a, double b, double c=DBL_MIN);

// 通过参数类型正常重载
int fmax(int a, int b);
```

另见

[重载](#), [虚拟函数](#), [多态](#)

外部函数描述

其他模块中定义的外部函数类型，被明确指出。如果没有这样的描述，将导致编译，创建，或者执行程序时的错误。当描述外部对象时，使用象征模块的关键字 `#import`。

示例：

```
#import "user32.dll"
    int      MessageBoxW(int hWnd ,string szText,string szCaption,int nType);
    int      SendMessageW(int hWnd,int Msg,int wParam,int lParam);
#import "lib.ex5"
    double   round(double value);
#import
```

导入帮助下，使描述外部DLL文件调用的函数或者编译EX5函数库变得容易。EX5函数库编译成带有[程序库](#)属性的ex5文件。只有[导出模块](#)描述的函数可以用EX5函数库导入。

另见

[重载](#), [虚拟函数](#), [多态](#)

输出函数

mql5 [输出](#)后置修饰符发布的函数可以用在其他mql5程序中。这种函数称为可输出函数，编辑后可以从其他程序调用。

```
int Function() export
{
}
```

该修饰符使编译器将函数添加进ex5文件导出的EX5函数表格中。只有带这个修饰符的函数可以从其他mql5程序中接入（“可见”）。

[程序库](#)属性通知编译程序EX5-文件是一个程序库，而编译程序会在EX5表头显示它。

所有计划输出的函数都要用 [输出](#)修饰符标记。

另见

[重载](#), [虚拟函数](#), [多态](#)

事件处理函数

MQL5语言提供[预定义事件](#)处理。处理这些事件的函数由MQL5程序确定；函数名，返回类型，参数组成（有几个参数情况下）以及类型必须严格遵照事件处理程序函数。

客户端的事件处理程序通过返回值类型和参数类型确定函数，来处理这个或者那个事件。如果其他参数，与下面描述不一致，但已被指定了相关的函数或者另一个返回类型，那么这个函数不能用于事件处理程序。

OnStart

OnStart() 函数就是 [启动](#) 事件处理程序，运行脚本时自动生成。一定是[空类型](#)，无参数：

```
void OnStart();
```

对于OnStart() 函数，指定[整型](#) 返回类型。

OnInit

OnInit() 函数是 [初始化](#) 事件处理程序。必须是[空型](#)或者[整型](#)，无参数：

```
void OnInit();
```

初始化事件处理程序在EA交易或者指标下载后即时生成；它不生成脚本。OnInit() 函数用于初始化。如果OnInit() 返回值为整型，非零结果意味着初始化失败，生成[初始化失败原因代码](#)[REASON__INITFAILED](#)。

空型OnInit() 函数代表初始化成功。

OnDeinit

OnDeinit() 函数称为失败初始化，是[初始化失败](#)事件处理程序。必须是[空型](#)且有一个包括[初始化失败原因代码](#)的常量整型参数。如果声明不同类型，编译程序会发出警告，但函数不可调用。对于脚本来说不会生成初始化失败事件，因此OnDeinit() 函数不用于脚本。

```
void OnDeinit(const int reason);
```

在以下情况下EA交易和指标生成初始化失败：The Deinit event is generated for Expert Advisors and indicators in the following cases:

- 再次初始化前，mql5程序下的交易品种和图表周期发生变化；
- 再次初始化前，[输入参数](#)发生变化；
- mql5程序卸载前。

OnTick

仅仅EA交易依附的图表中，交易品种收到新订单号时EA交易会生成 [新订单号](#) 事件。自定义指标或者脚本中确定OnTick() 函数是无效的，因为订单号事件不为它们而生。

订单号事件只为EA交易而生，但是却不意味着EA交易需要OnTick() 函数，因为EA交易不仅需要生成订单号，也需要生成计时器，预定事件和图表事件。这些都需要[空型](#)，无参数：

```
void OnTick();
```

OnTimer

[计时器](#)事件发生时调用OnTimer() 函数，由系统计时器生成，仅用于EA交易-不能用于脚本或者指标。函数EventSetTimer() 接收事件，当同意该事件声明时，设置事件发生频率。

可以用函数EventKillTimer() 为特殊EA交易注销计时器事件。函数必须为空值，没有参数：

```
void OnTimer();
```

建议在OnInit() 函数中调用EventSetTimer() 函数，而EventKillTimer() 函数可以在OnDeinit() 中调用。

每个EA交易，每个指标都有其独自の计时器且仅通过它来接收事件。如果创建了计时器但没有用EventKillTimer() 函数禁止，那么一旦mql5程序停止，计时器被强制破坏。

OnTrade

[交易](#)发生时调用这个函数，改变[下订单](#)和[持仓](#)列表，[订单历史记录](#)和[交易历史记录](#)时会出现。当交易活动执行挂单，持仓/平仓，停止设置，启动挂单等等，订单和交易历史记录或者仓位和当前订单列表也会相应改变。

```
void OnTrade();
```

接到这个事件时（需要交易策略状态）用户必须独自确认交易账户状态。如果函数OrderSend() 成功调用，且返回了真实值，意味着交易服务器已经提交订单并确定了订单号。服务器一旦处理这个订单，交易事件就生成了。如果用户记住了订单号值，OnTrade() 函数执行时，用这个值可以看到订单情况。

OnTester

函数OnTester() 是外部EA交易历史测试结束后自动生成的[测试](#)事件处理程序。用双型确定函数，无参数：

```
double OnTester();
```

这个函数在OnDeinit() 之前调用，且有同样的双型返回值。只在EA交易测试时使用OnTester() 。其主要作用是计算某个值，该值用作遗传最优化自定义标准。

遗传优化中，生成结果用降序，例如从优化标准查看点，最佳结果是那些最大值（OnTester函数带进账户的最大自定义优化标准值）。这种情况下，最差值放在最后，或者排除，不参与下一步生成。

OnBookEvent

OnBookEvent() 函数是[BookEvent](#)处理程序。BookEvent只在市场深度改变时，转为EA交易而生成。它必须是空值，且有一个字符串类型参数：

```
void OnBookEvent (const string& symbol);
```

若要接收任一交易品种的BookEvent，需要事先同意用函数MarketBookAdd() 接收这个交易品种的BookEvent。如果不同接收某个特定交易品种的BookEvent，可以调用 [MarketBookRelease\(\)](#)。

与其他事件不同的是BookEvent是直播。意味着如果EA交易同意用MarketBookAdd接收BookEvent，其他有OnBookEvent() 处理程序的EA交易也会收到这个事件。因此分析交易品种名称是必须的，可以const string& symbol参数传到处理程序。

OnChartEvent

OnChartEvent() 是[ChartEvent](#)的处理程序：

- CHARTEVENT_ KEYDOWN — 击键，图表窗口定格；

_OBJECT_DELETE=true)				
鼠标单击图表	CHARTEVENT_CLICK	X坐标	Y 坐标	—
鼠标单击属于图表的图解物件	CHARTEVENT_OBJECT_CLICK	X坐标	Y 坐标	事件发生的图解物件名称
用鼠标移动图解物件	CHARTEVENT_OBJECT_DRAG	—	—	移动的图解物件名称
图解物件标签编辑输入框中完成文本编辑	CHARTEVENT_OBJECT_EDIT	—	—	文本编辑完成的图解物件名称
	CHARTEVENT_CHART_CHANGE	—	—	—
N下的用户ID	CHARTEVENT_CUSTOM+N	通过函数 EventChartCustom() 设置值	通过函数 EventChartCustom() 设置值	通过函数 EventChartCustom() 设置值

OnCalculate

OnCalculate() 函数只在自定义指标中调用，通过[Calculate](#)计算指标值是必须的。通常在接到指标计算的交易品种新订单号时发生。这个指标不需要附在交易品种的价格图表上。

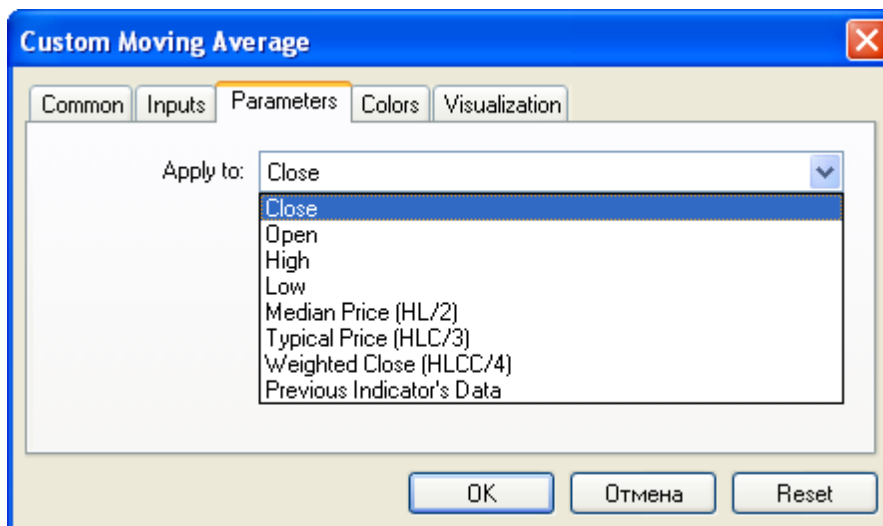
OnCalculate() 函数有个返回值int。有两个可能定义。一个指标中不可以有两个函数版本。

一种是用于单数据缓冲中计算的指标。例如，自定义移动平均数指标。

```
int OnCalculate (const int rates_total,      // 价格[] 数组的大小
                 const int prev_calculated, // 前一次调用处理的柱
                 const int begin,           // 有效数据起始位置
                 const double& price[]      // 计算的数组
                 );
```

价格[]数组中，可以传送时间序列和计算的一些指标缓冲。[ArrayGetAsSeries\(\)](#) 函数确定价格[] 数组索引方向。为了不依赖默认值，需要无条件的调用[ArraySetAsSeries\(\)](#) 函数用于工作的数组。

价格[]数组中，在“参数”标签启动指标时，选择适当的时间序列或者指标。所以，需要在“应用于”字段的下拉列表中指定必要的项目。



从其他mql5程序中接收自定义指标值，要使用*iCustom()* 函数，返回嵌入指标处理程序。可以指定适当的价格[]数组或者另一个指标处理程序。这个参数在自定义指标输入变量列表中最后传送。

示例：

```
void OnStart ()
{
//---
string terminal_path=StatusInfoString(STATUS_TERMINAL_PATH);
int handle_customMA=iCustom(Symbol(),PERIOD_CURRENT, "Custom Moving Average",13,0,
if(handle_customMA>0)
    Print("handle_customMA = ",handle_customMA);
else
    Print("Cannot open or not EX5 file '"+terminal_path+"\\MQL5\\Indicators\\"+"Cus
}
```

这个示例中，通过的最后参数是PRICE_ TYPICAL值（从ENUM_ APPLIED_ PRICE计数开始），指出自定义指标可以用获得的典型价格建立（高价+低价+平仓）/3。如果没有确定这个参数，指标基于PRICE_ CLOSE 值建立，例如每栏平仓价。

另一个示例显示依照指定价格[]数组的最后一个参数传送指标处理程序，由函数*iCustom()* 所描述。

另一种形式意在所有其他指标，计算更多的时间序列。

```
int OnCalculate (const int rates_total,      // 输入时间序列大小
                const int prev_calculated,   // 前一次调用处理的柱
                const datetime& time[],      // 时间
                const double& open[],        // 开盘价
                const double& high[],        // 最高价
                const double& low[],         // 最低价
                const double& close[],       // 收盘价
                const long& tick_volume[],   // 订单交易量
                const long& volume[],        // 真实交易量
                const int& spread[]          // 点差
                );
```

开盘价[], 最高价[], 最低价[]和收盘价[]参数由当前时间表的开盘价, 最高和最低价和收盘价数组组成。时间参数[]包括开盘时间值数组, 扩展参数[]有一个数组包括扩展历史记录 (如果为交易安全提供扩展)。volume[] 和 tick_volume[] 参数分别包括交易和交易量历史记录。

确定时间[]索引方向, 开盘价[], 最高价[], 最低价[], 收盘价[], 交易量[], 交易量[] 和 扩展[], 需要调用 [ArrayGetAsSeries\(\)](#) 函数。若不想依赖默认值, 需要无条件的调用函数[ArraySetAsSeries\(\)](#)用于工作的数组。

首先rates_total 参数包括栏的数量, 可用来计算指标, 与图表中现存的栏数一致。

需要注意OnCalculate() 返回值和第二输入参数prev_calculated的连接。调用函数时, prev_calculated 参数包括上次调用时OnCalculate() 返回值。这就允许用经济算法计算自定义指标, 避免重复计算。

返回rates_total参数值足够了, 包括当前调用函数的栏数。如果自从上次调用函数OnCalculate(), 价格数据更改了 (下载深度历史记录或者填满历史空白期), 输入参数prev_calculated 值由终端机设置为零。

注: 如果OnCalculate返回零, 那么指标值不能显示在客户端的数据窗口。

为更好的理解, 启动附加以下代码的指标很有用。

指标示例:

```
#property indicator_chart_window
#property indicator_buffers 1
#property indicator_plots 1
//---- 图的线
#property indicator_label1 "Line"
#property indicator_type1 DRAW_LINE
#property indicator_color1 clrDarkBlue
#property indicator_style1 STYLE_SOLID
#property indicator_width1 1
//--- 指标缓冲区
double LineBuffer[];
//+-----+
//| 自定义指标初始化函数 |
//+-----+
int OnInit()
{
//--- 指标缓冲区绘图
SetIndexBuffer(0, LineBuffer, INDICATOR_DATA);
//---
return(0);
}
//+-----+
//| 自定义指标重复函数 |
//+-----+
int OnCalculate(const int rates_total,
               const int prev_calculated,
               const datetime& time[],
               const double& open[],
               const double& high[],
```

```
        const double& low[],
        const double& close[],
        const long& tick_volume[],
        const long& volume[],
        const int& spread[])
    {
//--- 获得当前交易品种和图表周期的有效柱数
        int bars=Bars(Symbol(),0);
        Print("Bars = ",bars," rates_total = ",rates_total," prev_calculated = ",prev_calculated);
        Print("time[0] = ",time[0]," time[rates_total-1] = ",time[rates_total-1]);
//--- 为下次调用返回prev_calculated值
        return(rates_total);
    }
//+-----+
```

另见

[运行程序](#) [客户端事件](#) [工作事件](#)

变量

变量声明

可变量必须在声明之前使用。可变量必须拥有特殊的辨认名。相关可变量的定义描述会显示。

基本类型如下：

- 字符型, 短整型, 整型, 长整型, 无符号字符型, 无符号短整型, 无符号整型, 无符号长整型-整数；
- 颜色-代表RGB-颜色的整数；
- 日期时间-日期和时间, 自1970年1月1日起无符号整数包括秒数；
- 布尔数据-布尔值的true 和 false；
- 双精度数字-带有浮点的双精度数字；
- 浮点型-带有浮点的单精度数字；
- 字符串数据-特殊字符串。

示例：

```
string szInfoBox;
int      nOrders;
double   dSymbolPrice;
bool      bLog;
datetime tBegin_Data   = D'2004.01.01 00:00';
color     cModify_Color = C'0x44,0xB9,0xE6';
```

复合类型：

用其他类型构成的数据类型结构。

```
struct MyTime
{
    int hour;      // 0-23
    int minute;    // 0-59
    int second;    // 0-59
};
...
MyTime strTime; // 预先声明的结构 MyTime MyTime 的变量
```

结构类型声明前不可结构类型变量声明。

数组

相同数列数据被标注序列：

```
int      a[50];      // 50个整数的一维数组
double   m[7][50];   // 7个数组的二维数组，
                      // 每个都由50个数字组成。
MyTime   t[100];     // 数组包含元素例如 MyTime
```

唯一整数可以是数组指数。不允许四唯数列。数组元素开始编号为0。一个一维列阵的最后元素是1的数字比列阵大小。这就意味着, 请求数列的最后元素包括50 个整数将出现作为a[49]。维度被标注从0 到维度大小-1. 一个二

维数组的最后元素从示例将出现作为m[6][49]。

静态数组不能代表时间序列，例如[ArraySetAsSeries\(\)](#)函数从后到前接入数组，不能应用于此。如果想要在[时间序列](#)中接入数组，使用[动态数组对象](#)。

如果访问超出数列范围，执行的子系统将生成严重错误，程序就会停止。

接入说明符

接入说明符表示编译器如何接入变量，结构会员或者类。

常量 说明符声明常数变量，运行期不允许改变这个变量。变量声明时允许独自初始化。常量说明符不能应用于结构会员和类。

样本

```
int OnCalculate (const int rates_total,      // 价格[] 数组的大小
                const int prev_calculated,  // 前一次调用处理的柱
                const int begin,           // 有效数据起始位置
                const double& price[]      // 数组计算
                );
```

若要接入结构会员和类使用以下限定符：

- [公用限定符](#) -允许自由接入变量或者类函数
- [受保护限定符](#) -允许从这个类函数或者[公共继承](#)类函数中接入。不允许其他接入；
- 私人限定符-允许仅从同类函数中接入变量和类函数。
- [虚拟限定符](#) -仅应用于类函数（不是结构函数）并且告诉编译器这个函数置于类虚拟函数表格中。

存储类

有三种存储类：[静态](#) -[输入](#) -[外部](#)。存储类的这些调节器明确表明内存预分配区分配的相关变量的编译器，称为全局池。此外，这些调节器表示特别处理变量数据。如果局部声明的变量不是[静态](#)，这个变量内存自动分配程序堆。非静态数组分配的自由内存也是在超出声明数组可见区时自动完成。

另见

[数据类型](#), [类型密封和扩展](#) -[变量初始化](#), [可见范围和变量使用期](#), [创建和删除对象](#)

局部变量

在函数内可变量的声明是局部的。局部变量在声明的部分里是被限定的。局部变量可以由任意一个表达式结果初始化。每次函数的运行只可以初始化一个局部变量。局部变量储存在相应的存储器上。

示例：

```
int somefunc()
{
    int ret_code=0;
    ...
    return(ret_code);
}
```

变量范围是涉及变量的模块部分。模块中声明的变量（内部），有模块规定其范围。变量声明启动模块范围，以终止程序结束。

开始函数声明的局部变量也有模块范围，和局部变量函数参数。任何模块都可以包括变量声明。如果嵌入模块内外模块有同名标识符，隐藏外部指标，直到内部操作完成。

示例：

```
void OnStart()
{
    //---
    int i=5;      // 函数的局部变量
    {
        int i=10; // 函数变量
        Print("In block i = ",i); // 结果是 i=10;
    }
    Print("Outside block i = ",i); // 结果是 i=5;
}
```

这意味着内部模块运行时，只看到自己的局部指标值，看不到外部同名指标值。

示例：

```
void OnStart()
{
    //---
    int i=5;      // 函数局部变量
    for(int i=0;i<3;i++)
        Print("Inside for i = ",i);
    Print("Outside the block i = ",i);
}
/* Execution result
Inside for i = 0
Inside for i = 1
Inside for i = 2
Outside block i = 5
*/
```

即使从启动模块就存在，公开[静态](#)局部变量仍有模块范围。

另见

[数据类型](#)， [类型密封和扩展](#) - [变量初始化](#)， [可见范围和变量使用期](#)， [创建和删除对象](#)

形式参数

通过函数的变量是**局部**的。范围是在作用块内。在作用之内正式变量的名称必须不同于其他外部定义变量和函数变量。作用块内的正式变量值已经被赋予。如果正式参数以**常量**修饰符声明，其值在函数中不能改变。

示例：

```
void func(const int x[], double y, bool z)
{
    if(y>0.0 && !z)
        Print(x[0]);
    ...
}
```

正式参量可能由常量**初始化**。在这种情况下，初始化的值作为缺省值被考虑。参量，在初始化旁边，必须初始化。

示例：

```
void func(int x, double y = 0.0, bool z = true)
{
    ...
}
```

这样作用显现时，初始化的参量可能被省去，缺省值会代替它们。

示例：

```
func(123, 0.5);
```

简易类型参量值通过。在任何情况下，这种**局部变量**的修改将不会显示在功能板块内。任何数组和结构类型数据都是通过引用传送。如果需要禁止修改数组和结构内容，这些类型参数必须用关键字 const 声明。

有机会通过引用传送简单类型参数。在这种情况下，修改的这样参量将影响调用函数对应的变量。为指出引用传送参数，数据类型后放符号&。

示例：

```
void func(int& x, double& y, double& z[])
{
    double calculated_tp;
    ...
    for(int i=0; i<OrdersTotal(); i++)
    {
        if(i==ArraySize(z)) break;
        if(OrderSelect(i)==false) break;
        z[i]=OrderOpenPrice();
    }
    x=i;
    y=calculated_tp;
}
```

引用传送的参量无法通过默认值初始化。

传到函数的最大参量不可以超过64个。

另见

[输入变量](#) - [数据类型](#), [类型密封和扩展](#) - [变量初始化](#), [可见范围和变量使用期](#), [创建和删除对象](#)

静态变量

静止存储类称为静态变量。在数据类型之前指定静态修饰符。

示例：

```
int somefunc ()
{
    static int flag=10;
    ...
    return(flag);
}
```

与简单的仅能用表达式初始化的局部变量不同的是，静态变量可以通过相关的常量和常量表达式[初始化](#)。

静态变量存在于程序执行期，在特别函数[OnInit\(\)](#) 调用前初始化。如果未指定初始化值，静态变量为零初始化值。

关键字 static 声明的[局部变量](#)通过函数[使用期](#)保留值。下次调用函数时，局部变量会包含上次调用的值。

任何变量，除了函数的[形式参数](#)，都称为静态变量。如果局部变量声明不是静态，变量内存会在程序中自动分配。

示例：

```
int Counter()
{
    static int count;
    count++;
    if(count%100==0) Print("Function Counter has been called ",count," times");
    return count;
}

void OnStart()
{
    //---
    int c=345;
    for(int i=0;i<1000;i++)
    {
        int c=Counter();
    }
    Print("c =",c);
}
```

另见

[数据类型](#), [类型密封和扩展](#) - [变量初始化](#), [可见范围和变量使用期](#), [创建和删除对象](#)

全局变量

全局变量通过函数外部描述声明而创建。全局变量作为函数被定义在相同水平，即，不可以局部使用。

示例：

```
int GlobalFlag=10;    // 全局变量
int OnStart()
{
    ...
}
```

全局变量的范围是整个程序。全局变量在所有程序内是被定义的。如果它的值没有被定义，初始化值为零。全局变量只对于相应的常数初始化。

全局变量只可以在程序载入客户端以后初始化。

注解: 变量在全局变量的水平位上不能够与客户端[GlobalVariable...\(\)](#) 函数混淆。

另见

[数据类型](#), [类型密封和扩展](#), [变量初始化](#), [可见范围和变量使用期](#), [创建和删除对象](#)

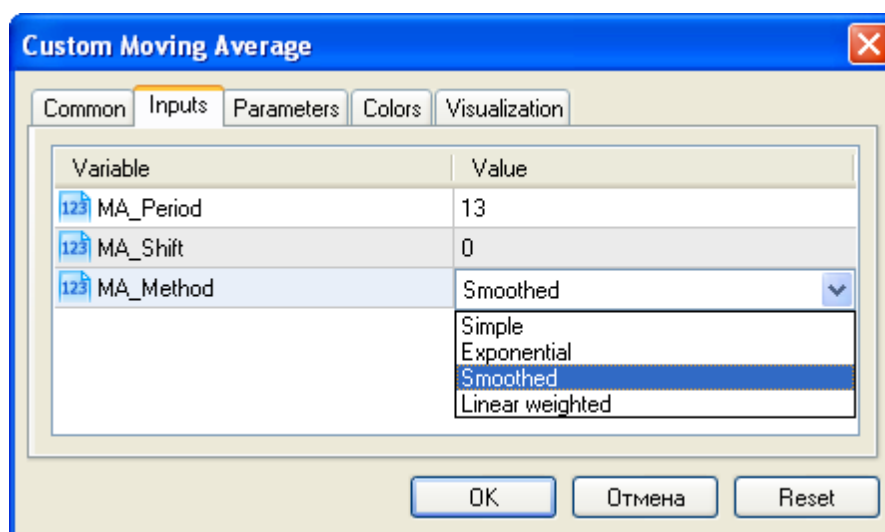
输入变量

输入存储类称作外部变量。input修饰符位于数据类型前。输入修饰符变量不可以在mql5程序中修改，只能只读访问这个变量。用户只能通过程序属性窗口改变输入变量值。

示例：

```
//--- 输入参数
input int      MA_Period=13;
input int      MA_Shift=0;
input ENUM_MA_METHOD MA_Method=MODE_SMMMA;
```

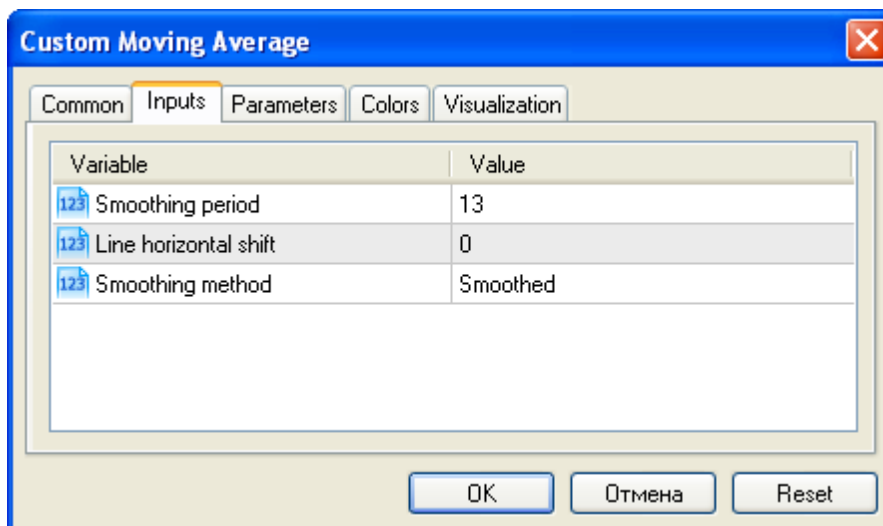
输入变量决定程序的输入参数。可以在程序属性窗口中得到。



输入标签中可以通过其他方法显示输入参数名。使用字符注解，放在同行输入参数描述后。因此对于用户来说，与名字匹配的参数更容易理解。

示例：

```
//--- 输入参数
input int      InpMAPeriod=13;           // 平滑周期
input int      InpMAShift=0;             // 水平移动行
input ENUM_MA_METHOD InpMAMethod=MODE_SMMMA; // 平滑函数
```



注：Note: A [复合型](#)数组和变量不能作为输入变量。

MQL5程序访问自定义指标时传送参数

用 `iCustom()` 函数访问自定义指标。其名前，参数要与该指标声明的输入变量严格一致。如果参数少于变量，缺失的参数要在变量声明时用特殊值填满。

如果自定义指标使用第一型的 `OnCalculate` 函数（例如用同数据数组计算的指标），那么 `ENUM_APPLIED_PRICE` 值或者其他指标处理程序作为访问这个自定义指标的上一个参数被使用。与输入变量一致的所有参数必须清楚指明。

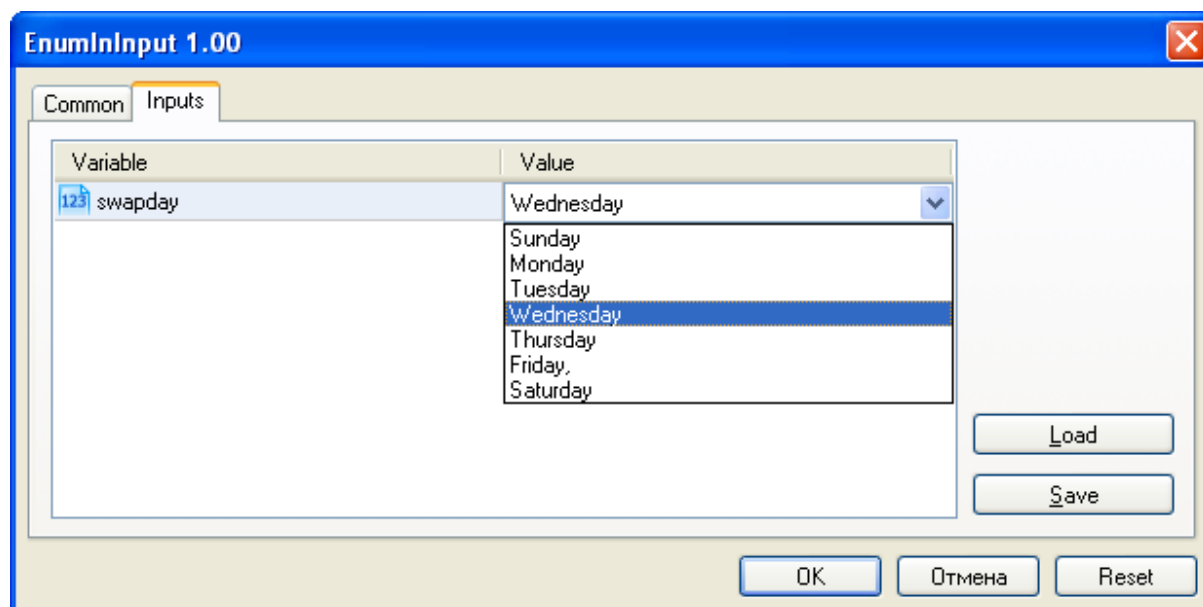
计数作为输入参数

MQL5提供得嵌入计数和用户指定的变量都可以用作输入变量（mql5程序的输入参数）。例如，可以创建 `dayOfWeek` 计数，描述一周的天数，使用输入变量指定某一天，不是数字，用更通常的方法。

示例：

```
#property script_show_inputs
//--- 一周内每天
enum dayOfWeek
{
    S=0,      // 周日
    M=1,      // 周一
    T=2,      // 周二
    W=3,      // 周三
    Th=4,     // 周四
    Fr=5,     // 周五,
    St=6,     // 周六
};
//--- 输入参数
input dayOfWeek swapday=W;
```

脚本启动时，为使用户从属性窗口选择必要的值，使用预处理命令 `#property script_show_inputs`。启动脚本并从列表中选择一个 `dayOfWeek` 枚举值。启动 `EnumInInput` 脚本然后去输入标签。默认情况下，`swapday` 值（三个交换日）是周三（`W=3`），但可以指定另一个值，用这个值改变程序操作。



列举值有限。因此，若要选择输入值，需使用下拉列表。为列表中显示的值使用列举助记名。如果注释和助记名有关，正如示例显示，注释内容代替助记名。

每个dayOfWeek计数值都是0-6，但是参数列表中，显示每个值的注释。明确描述输入参数使编程更灵活。

另见

[i自定义](#) - [计数](#) - [程序属性](#)

外部变量

关键字`extern` 用来声明变量标识符，作为全局[使用期](#)的[静态存储](#)标识符。从程序启动，这些变量即存在，其内存会在程序启动后即时分配和初始化。

创建多源文件程序；这个情况下直接使用预处理[#include](#)。外部声明的同一类型和标识符的变量位于一个项目中的不同源文件。

编译整个项目时，所有同型同标识符的外部变量与全局变量池相连。外部变量用来分离编译源文件。外部变量可以初始化，只能一次-禁止几个同型同标识符外部变量初始化。

另见

[数据类型](#)， [类型密封和扩展](#) - [变量初始化](#)， [可见范围和变量使用期](#)， [创建和删除对象](#)

变量初始化

下定义时可以初始化任何变量。如果变量没有显性初始化，其值可为任何值。不可使用隐性初始化。

全局和静态变量仅可以通过相关的常量或者常量表达式初始化。全局变量仅可以通过表达式初始化（不是常量）。

全局变量和静态变量初始化仅可以执行一次。局部变量每次调用相关函数都进行初始化。

示例：

```
int    n        = 1;
string s        = "hello";
double f[]      = { 0.0, 0.236, 0.382, 0.5, 0.618, 1.0 };
int    a[4][4] = { {1, 1, 1, 1}, {2, 2, 2, 2}, {3, 3, 3, 3}, {4, 4, 4, 4} };
//--- 来自 tetris
int    right[4]={WIDTH_IN_PIXELS+VERT_BORDER,WIDTH_IN_PIXELS+VERT_BORDER,
                  WIDTH_IN_PIXELS+VERT_BORDER,WIDTH_IN_PIXELS+VERT_BORDER};
```

数组元素值列表必须被附寄在括号内。初始化省去的值被考虑为零。初始化至少有一个值：即相关结构和数组第一元素的初始化值，缺失的元素考虑为零。

如果没有指定初始化数组大小，则由基于初始化序列大小的编译器决定。多维数组不同用一维序列初始化（没有另外括号的序列），只有仅指定一个初始化元素时除外（零，通常）。

数组（包括局部变量公开的）只能以常数初始化。

示例：

```
struct str3
{
    int          low_part;
    int          high_part;
};
struct str10
{
    str3         s3;
    double       d1[10];
    int          i3;
};
void OnStart()
{
    str10 s10_1={{1,0},{1.0,2.1,3.2,4.4,5.3,6.1,7.8,8.7,9.2,10.0},100};
    str10 s10_2={{1,0},{0},100};
    str10 s10_3={{1,0},{1.0}};
    //---
    Print("1.  s10_1.d1[5] = ",s10_1.d1[5]);
    Print("2.  s10_2.d1[5] = ",s10_2.d1[5]);
    Print("3.  s10_3.d1[5] = ",s10_3.d1[5]);
    Print("4.  s10_3.d1[0] = ",s10_3.d1[0]);
}
```

另见

[数据类型](#), [类型密封和扩展](#), [可见范围和变量使用期](#), [创建和删除对象](#)

可见范围和变量使用期

有两种基本范围：[局部](#)范围和[全局](#)范围。

所有函数外部变量声明位于全局范围。程序的任何地方都可以接入这个变量。这个变量位于内存全局池，所以其使用期同于程序使用期。

内部变量声明属于局部范围。这种变量外部不可见（因此无法得到）。大部分局部变量声明都是一个函数内声明的。局部变量声明位于存储器中，使用期同于函数使用期。

局部变量范围在存储器中声明，也可以在其他存储器以同名声明；也可以高级别声明，乃至全局范围。

示例：

```
void CalculateLWMA(int rates_total,int prev_calculated,int begin,const double &price[
{
    int          i,limit;
    static int weightsum=0;
    double       sum=0;
    //---
    if(prev_calculated==0)
    {
        limit=MA_Period+begin;
        //--- 为第一限价柱设置空值
        for(i=0; i<limit; i++) LineBuffer[i]=0.0;
        //--- 首先计算明显值
        double firstValue=0;
        for(int i=begin; i<limit; i++)
        {
            int k=i-begin+1;
            weightsum+=k;
            firstValue+=k*price[i];
        }
        firstValue/=(double)weightsum;
        LineBuffer[limit-1]=firstValue;
    }
    else
    {
        limit=prev_calculated-1;
    }

    for(i=limit;i<rates_total;i++)
    {
        sum=0;
        for(int j=0; j<MA_Period; j++) sum+=(MA_Period-j)*price[i-j];
        LineBuffer[i]=sum/weightsum;
    }
    //---
}
```

注意变量*i*，直线声明

```
for(int i=begin; i<limit; i++)  
{  
    int k=i-begin+1;  
    weightsum+=k;  
    firstValue+=k*price[i];  
}
```

其范围只是一个循环；外圈是同名的函数开始就声明的另一个变量。此外，变量*k*在内部声明，其范围就是循环内部。

访问[静态](#)说明符局部变量声明。这种情况下，编译器在内存全局池有一个变量。因此，静态变量使用期等于程序使用期。这就是变量范围的限制。

另见

[数据类型](#), [类型密封和扩展](#) - [变量初始化](#), [创建和删除对象](#)

创建和删除对象

mql5程序执行后，内存会按照变量类型分配。根据接入水平，所有变量分成两类-[全局变量](#)和[局部变量](#)。根据内存等级，它们可以是mql5的[输入参数](#) -[静态的](#)或者自动的。如果需要，每个变量用相应值[初始化](#)。使用后，程序结束，使用的内存回到MQL5执行系统中。

初始化和无法初始化全局变量

mql5程序载入以及调用函数前，全局变量自动初始化。初始化过程是，初始值分配到[简易](#)类型变量中，并且调用构造函数（若有的话）。全局范围声明[输入变量](#)，程序启动时在对话框中设置初始值。

即使通常在局部范围声明[静态](#)变量，但是与[全局](#)变量一样，仍要预分配内存，程序载入后初始化。

初始化命令与程序中声明的变量命令一致。相反就执行无法初始化命令。这个规则只适用于非新操作符创建的变量。载入后，变量自动创建和初始化，程序卸载前，无法初始化。

初始化和无法初始化局部变量

如果声明局部变量不是静态的，内存会自动分配。局部变量与全局变量一样，程序遇到局部变量时自动初始化。因此初始化命令与声明命令一致。

局部变量在程序结束时无法初始化。程序模块是[复合操作符](#)，是选择操作符[切换](#)，循环操作符（[for](#), [while](#), [do-while](#)），[函数主体](#)，或者[if-else 操作符](#)的一部分，

局部变量只在执行程序遇到变量声明时初始化。如果程序执行时，分程序没有执行，那么变量不会初始化。

安置对象的初始化和无法初始化

特殊情况就是[对象指针](#)，因为声明指针无能导致相关对象初始化。动态的安置对象仅当[新操作符](#)创建简单等级时初始化。对象初始化假定调用相关类的构造函数。如果没有类的相关构造函数，[简易类型](#)不会自动初始化；[字符串](#)，[动态数组](#) 和 [复合对象](#) 也会自动初始化。

指针在局部或者全局声明；可以通过空值[NULL](#)或者同指针值或[遗传](#)类型值初始化。如果[新操作符](#)为局部声明指针调用，这个指针的[删除操作符](#)退出前也要执行。否则指针消失，对象显性删除失败。

通过表达式`object_pointer=new Class_name`创建的所有对象一定通过删除操作符(`object__pointer`) 来删除。如果程序完成后因为某种原因无法用[删除 操作符](#)删除变量，“专家”日志中会出现相应输入。可以声明几个变量，然后给它们安排一个指针。

如果动态创建的对象有构造函数，执行[新操作符](#)时调用这个函数。如果对象有析构函数，执行[删除操作符](#)是使用。

因此动态安置的对象只有用[新操作符](#)创建，而用[删除操作符](#)删除或者执行MQL5卸载程序时删除。动态创建的对象声明的指针不会影响其初始化命令。初始化和无法初始化由程序员控制。

变量简要特征

创建，删除，调用构造函数和析构函数命令的信息如下表。

	全局自动变量	局部自动变量	动态创建对象
--	--------	--------	--------

初始化	加载mql5后	执行中公开代码线到达时	执行新操作符
初始化命令	初始化命令	命令声明	不考虑命令声明
无法初始化	mql5程序卸载前	推出公示分程序	执行删除操作符或者 mql5卸载前
无法初始化命令	与初始化命令相反的命令	与初始化命令相反的命令	不考虑初始化命令
调用构造函数	加载mql5程序	初始化	执行新操作符
调用析构函数	卸载mql5程序	退出变量初始化分程序	执行删除操作符
错误日志	“专家”日志中尝试删除 自动创建的对象	“专家”日志中尝试删除 自动创建的对象	“专家”日志，卸载 mql5程序时不删除动态 穿件的对象。

另见

[数据类型](#), [类型密封和扩展](#), [变量初始化](#), [可见范围和变量使用期](#)

预处理程序

预处理程序是一个特殊MQL4的子程序，在程序执行之前预先准备的程序源代码。

预处理程序会尽可能地读取源代码。代码的结构可能包括MQL4 程序源代码的特殊文件。对于读取的代码尽可能地按照具体常数分配储存。

预处理程序允许mql5 程序参量指定。

- [常量声明](#)
- [设置程序属性](#)
- [程序文本文件](#)
- [输入函数](#)

如果# 标志被使用在程序的第一线, 这条线是预处理程序方针。预处理程序方向末端以换行字符结尾。

常量声明 (#define)

#define 直接用来指定常量助记名。 :

```
#define identifier expression // беспараметрическая форма
#define identifier(par1,... par8) expression // параметрическая форма
```

#define 直接代替 expression 为在源文件中找出更多的标识符目录。标识符仅代表自己时可被替换。如果标识符是注解一部分，字符串一部分或者其他标识符一部分，则不能代替。

常量标识符与变量名同样规则。值可以是多种类型：

```
#define ABC 100
#define PI 3.14
#define COMPANY_NAME "MetaQuotes Software Corp."
...
void ShowCopyright()
{
    Print("Copyright 2001-2009, ",COMPANY_NAME);
    Print("http://www.metaquotes.net");
}
```

expression 由几种标记组成，例如关键字，常量，有常量无量表达式。expression在直线末端结束，不可转到下一行。

示例：

```
#define TWO 2
#define THREE 3
#define INCOMPLETE TWO+THREE
#define COMPLETE (TWO+THREE)
void OnStart()
{
    Print("2 + 3*2 = ",INCOMPLETE*2);
    Print("(2 + 3)*2 = ",COMPLETE*2);
}
// 结果
// 2 + 3*2 = 8
// (2 + 3)*2 = 10
```

#define

identifier

expression

```
// пример с двумя параметрами a и b
#define A 2+3
#define B 5-1
#define MUL(a, b) ((a)*(b))
```



```
double c=MUL(A,B);
Print("c=",c);
/*
выражение double c=MUL(A,B);
равносильно double c=((2+3)*(5-1));
*/
// Результат
// c=20
```

expression,

,

:

```
// пример с двумя параметрами a и b
#define A 2+3
#define B 5-1
#define MUL(a, b) a*b

double c=MUL(A,B);
Print("c=",c);
/*
выражение double c=MUL(A,B);
равносильно double c=2+3*5-1;
*/
// Результат
// c=16
```

8

```
// правильная параметрическая форма
#define LOG(text) Print(__FILE__, "(", __LINE__, ") :", text) // один параметр - 'text'

// неправильная параметрическая форма
#define WRONG_DEF(p1, p2, p3, p4, p5, p6, p7, p8, p9) p1+p2+p3+p4 // более 8 параме
```

另见

[标识符](#) - [字符常量](#)

程序属性 (#property)

每个mql5-程序都允许指定额外的名为#property的特殊参数，有助于客户端不用启动程序就能够进行贴切的互联网服务。首先，它包括指标外部设置。包括文件中描述的属性完全忽略。属性必须在mql5-主文件中指定。

```
#property identifier value
```

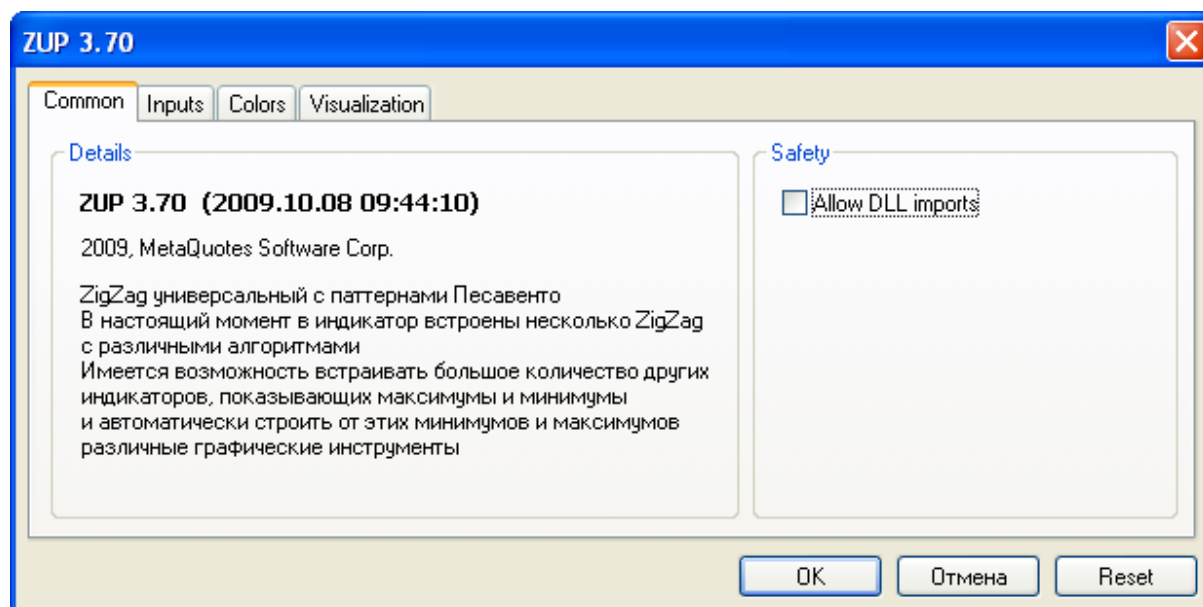
编译器在执行模块配置中编写声明值。

常数	类型	描述
连接	字符型	链接到公司网站
版权	字符型	公司名
版本	字符型	程序版本，最大31位字符
描述	字符型	mql5-程序的简单文本描述。现存几种描述，每个描述文本中的一行。包括换行所有描述总长度不能超过511个字符。
堆栈大小	整型	循环调用堆栈大小
程序库		程序库；指定非启动函数，带有输出修饰符的函数也可以引入到其他mql5-程序中
indicator_ applied _ price	整型	指定默认值到“应用”字段。如果属性没有指定，您可以指定一个ENUM_APPLIED_PRICE值，默认值为PRICE_CLOSE
indicator_ chart_ window		显示图表窗口指标
indicator_ separat e_ window		显示独立窗口指标
indicator_ height	int	(INDICATOR_HEIGHT)
indicator_ buffers	整型	指标计算缓冲区
indicator_ plots	整型	指标图解系列数目
indicator_ minimu m	双精度 型	独立指标窗口底部比例限制
indicator_ maxim um	双精度 型	独立指标窗口顶部比例限制
indicator_ labelN	字符型	为数据窗口显示的N-th 图解系列设置标签。对于需要多指标缓冲 (DRAW_CANDLES, DRAW_FILLING and others) 的图解系列，通过分号“;”规定标签名。
indicator_ colorN	颜色	N行颜色，N为图解系列的数量；从1开始计数
indicator_ widthN	整型	图解系列粗线条，N为图解系列数量，从1开始计数
indicator_ styleN	整型	图解系列的线型，由ENUM_LINE_STYLE 值指定。N为图解系列数量，从1开始计数

indicator__typeN	整型	绘图种类，由 ENUM_DRAW_TYPE 值指定。N为图解系列数量，从1开始计数
indicator__levelN	双精度型	独立指标窗口中N水平线
indicator__levelcolor	颜色	指标水平线颜色
indicator__levelwidth	整型	指标粗水平线
indicator__levelstyle	整型	指标水平线类型
script_show_confirm		运行脚本前显示确认窗口
script_show_inputs		运行脚本前显示属性窗口，禁止确认窗口
tester__indicator	字符型	以“indicator_name.ex5”形式命名自定义指标。如果相关参数通过常量字符串设置，自动定义调用 iCustom() 函数需要测试的指标。对于其他情况下（使用函数 IndicatorCreate() 或者使用无常量字符串的参数设置指标名）属性是必须的
tester__file	字符型	双引号中（作为常数字符串）扩展指示测试文件名。指定的文件传送到测试。若需要的话，测试的输入文件必须被指定。
tester__library	字符型	双引号中扩展程序库名。程序库有dll或者ex5扩展名。需要测试的程序库自动定义。然而，如果任何程序库用于 自定义 指标，属性是必须的。

描述和版本号的任务样本

```
#property version      "3.70"          // EA交易的当前版本
#property description  "ZigZag universal with Pesavento Patterns"
#property description  "At the moment in the indicator several ZigZags with different
#property description  "It is possible to embed a large number of other indicators sho
#property description  "lows and automatically build from these highs and lows various
```



指定指标缓存独立标签示例（“C开盘价；C最高价；C最低价；C收盘价”）

```
#property indicator_chart_window
#property indicator_buffers 4
#property indicator_plots 1
#property indicator_type1 DRAW_CANDLES
#property indicator_width1 3
#property indicator_label1 "C open;C high;C low;C close"
```



包括文件 (#include)

#include 命令可以放置到程序的任意部分，但是通常所有文件的源代码被统一放置。调用格式；

```
#include <file_name>
#include "file_name"
```

示例：

```
#include <WinUser32.mqh>
#include "mylib.mqh"
```

预处理程序用 WinUser32.mqh. 文件内容代替线 #include <file_name>。三角括号表示WinUser32.mqh文件将会从标准目录调用(通常目录为terminal_installation_directory\MQL5\Include) 。当前目录不可见。

如果文件名锁在引号中，搜索将在当前目录中执行(加载的源代码主文件) 。 标准目录不可见。

另见

[标准数据库](#) - [输入函数](#)

输入函数 (#import)

函数从MQL5编译模板(*. ex5 文件) 和执行系统文件模板(文件 *. dll) 通过。模板名称被指定在#import指令中。能够正确形成输入函数的编译器调用和组织适当的[参数传送](#)，需要带有完整的[函数](#)描述部分。函数描述会立即按照#import “模板名称” 执行。新的#import命令完成引入输入函数描述部分。

```
#import "file_name"
    func1 define;
    func2 define;
    ...
    funcN define;
#import
```

输入函数可以有几个名称。相同名称不同的模块的函数可以同时输入。输入函数名与嵌入函数名一致。[范围解析](#)操作决定需要调用哪个函数。

因为引入函数是在模块外面被编写，编译器无法检查通过参量的正确性。因此，为避免运行错误，它必须精确地描述传送到输入函数的参数的组成和命令。传到输入函数的参数（从EX5，和从DLL-模块）可以有默认值。

以下在输入函数中不能用作参数：

- [指针](#) (*) ；
- 连接[动态数组](#)或者指针的对象。

类，字符串数组或者包括字符串或者动态数组的复合对象不能作为参数传送到DLL输入函数。

示例：

```
#import "user32.dll"
int    MessageBoxW(uint hWnd,string lpText,string lpCaption,uint uType);
#import "stdlib.ex5"
string ErrorDescription(int error_code);
int    RGB(int red_value,int green_value,int blue_value);
bool   CompareDoubles(double number1,double number2);
string DoubleToStrMorePrecision(double number,int precision);
string IntegerToHexString(int integer_number);
#import "ExpertSample.dll"
int    GetIntValue(int);
double GetDoubleValue(double);
string GetStringValue(string);
double GetArrayItemValue(double &arr[],int,int);
bool   SetArrayItemValue(double &arr[],int,int,double);
double GetRatesItemValue(double &rates[][6],int,int,int);
#import
```

在mq5 程序执行期间输入函数，需要使用前期绑定。 这就意味着ex5程序加载时启动资料库。

不建议使用Drive:\Directory\FileName.Ext加载模块的权限定名。MQL5资料库会从 terminal_dir\MQL5\Libraries文件夹中加载。

另见

[包括文件](#)

面向对象的程序设计

面向对象的程序设计(OOP) 主要是针对数据或者与数据密不可分行文的程序设计。数据和行为合起来称为类，对象就是类的实例。

面向对象处理组件：

- [类型封装和扩展性](#)
- [继承机制](#)
- [多态性](#)
- [重载](#)
- [虚拟函数](#)

OOP把计算当成行为建模。模型项是抽象计算代表的对象。假设我们想编写一个著名的游戏"Tetris"。我们必须学习如何用四个正方形随机组成的图形来建模。我们还需要调节图形下降的速度，定义图形旋转变换的操作。屏幕上图形移动受文档对象模型边框限制，这也需要建模。此外，消除填满的行，然后获得相应的得分。

因此，这个简单易懂的游戏需要创建几个模型-图形模型，文档对象模型，移动模型等等。所有这些模型都是抽象的，通过电脑运算表示。描述这些模型，抽象数据类型ADT（或者[复杂数据类型](#)）概念被使用。严格说来，DOM中“图形”运动模型并不是数据类型，但是它是使用“DOM”数据类型限制条件，“图形”数据类型全部操作。

对象是[类](#)变量。面向对象的程序设计允许您轻松创建使用ADT。面向对象程序设计使用继承机制。其优势是允许从用户已经定义的数据类型中获得衍生类型。

例如，创建俄罗斯方块图形，首先创建一个基本类图形很方便；代表所有可能图形的其他七个类可以从基本图形衍生出来。图形行为在基本类中就确定了，而执行每个独立图形的行为在衍生类中定义。

在OOP中对象对其行为负责。ADT开发人员包括描述相关对象期望的行为代码。实际上，对象对其行为负责，显著简化了对象用户程序设计的任务。

如果我们想要在屏幕上画一个图，我们需要知道中心点在哪里，以及如何画它们。如果独立的图形知道如何画自己，当使用这个图形时程序员可以发送“draw”消息。

MQL5语言类似于C++，也有ADT的[封装](#)机制。一方面，封装与内部特殊类型结合，另一方面，也影响这类型对象的可接入外部函数连结。执行细节对于使用这个类型的程序很难达到。

OOP概念有一系列相关概念，包括以下内容：

- 模拟真实世界操作
- 用户定义数据类型的有效性
- 隐藏执行细节
- 通过继承机制重新使用代码的可能性
- 执行期解释调用函数

一些概念非常不明确，有一些很抽象，另一些又很大众化。

类型的密封和扩展Encapsulation and Extensibility of Types

OOP是编写软件的均衡办法。数据和行为放在一起。密封创建了定义用户的数据类型，扩展语言数据类型并相互作用。类型扩展是一个向语言添加用户定义数据类型的机会，该类型像[基本类型](#)一样，很容易使用。

抽象数据类型，例如，字符串，是著名行为类型理想的描述。

字符串用户知道字符串操作，例如连结或者打印，都有一定的行文。连结和打印操作称为类函数。

某种ADT执行有一定的限制，例如，字符串长度有限制。这些限制影响了行为全部开放。同时，内部或者私人执行细节不直接影响用户看对象的方法。例如，字符串尝以数组来执行，而这个数组的内部基本地址及其名称对于用户则不是必须的。

密封就是当向用户定义类型提供端口时，隐藏执行细节的能力。在MQL5中，及在C++中，类和结构定义（[类](#)和[结构](#)）用于接入关键字 `private`、`protected`和`public`连结的封存规定中。

关键字`public` 接入其后面的成员，无任何限制就可以打开。无关键字，类成员默认封锁。个人成员仅可以接入这个类里成员函数。

Protected类函数不仅可以在其类中得到，也可以在其继承类中得到。Public类函数可以在类声明范围任一函数中得到。保护可以隐藏类的部分执行，因此阻止了数据结构中意外改变。接入限制和数据隐藏都是面向对象程序设计的一个特点。

通常情况下，类函数由`protected`修饰符保护和声明，读取和编写值通过使用`publics`接入修饰符定义的所谓的特别的set-and get-类函数来执行。

示例：

```
class CPerson
{
protected:
    string      m_name;           // 名称
public:
    void        SetName(string n){m_name=n;} // 设置 名称
    string      GetName(){return (m_name);} // 返回 名称
};
```

这个办法提供了几种优势。首先，通过函数名我们知道它是做什么的-设置或者获得类成员值。其次，也许在将来我们将需要改变CPerson类中或者任一衍生类中的m_name CPerson变量类型。

在这种情况下，我们只需要改变执行函数SetName() 和GetName()，而CPerson类的对象可用于使用无代码改变的程序因为用户甚至不知道m_name数据类型已经改变。

示例：

```
struct Name
{
    string      first_name;       // 名称
    string      last_name;       // 上一个 名称
};
```



```
class CPerson
{
protected:
    Name                m_name;                // 名称
public:
    void                SetName(string n);
    string              GetName() {return(m_name.first_name+" "+m_name.last_name);}
private:
    string              GetFirstName(string full_name);
    string              GetLastName(string full_name);
};

void CPerson::SetName(string n)
{
    m_name.first_name=GetFirstName(n);
    m_name.last_name=GetLastName(n);
}

string CPerson::GetFirstName(string full_name)
{
    int pos=StringFind(full_name," ");
    if(pos>0) StringSetCharacter(full_name,pos,0);
    return(full_name);
}

string CPerson::GetLastName(string full_name)
{
    string ret_string;
    int pos=StringFind(full_name," ");
    if(pos>0) ret_string=StringSubstr(full_name,pos+1);
    else     ret_string=full_name;
    return(ret_string);
}
```

另见

[数据类型](#)

继承算法

OOP属性特点是通过继承算法鼓励代码重复使用。新类从现存的，基本类中生成。衍生类使用基本类成员，但是也做以更改和补充。

许多类型是现存类型的变异。为每个类开发新代码非常乏味。此外，新代码意味着新的错误。衍生类继承了基本类的描述，因此不必重复开发和重复测试代码。继承关系是层级体系。

层级是一个允许复制所有多样性和复杂性元素的类函数。它介绍了对象等级。例如，元素周期表有气体。它们对所有的周期元素控制固有的属性。

惰性气体是下一重要子集的构成。层级就是惰性气体，例如氩是气体，是系统的一部分。这样的层级可以很轻松的解释惰性气体。我们知道其原子包括质子和电子，对于所有其他元素也是如此。

我们知道如其他气体一样它们在常温也是气体状态。我们知道惰性气体子集中，无气体能与其他化学元素发生化学反应，它是所有惰性气体的特性。

考虑到几何图形的继承例子。描述各种简单图形（圆形，三角形，矩形，正方形等等），最好的方法就是创建基本类（[ADI](#)），它是所有衍生类的先祖。

让我们创建一个基本类 CShape，它包括描述图形最常用的构件。这些构件描述任何图形所特有的属性-图像类型和定位点主坐标。

示例：

```
//--- 基本类的形状
class CShape{
{
protected:
    int      m_type;           // 形状类型
    int      m_xpos;           // 基本点的 X - 坐标
    int      m_ypos;           // 基本点的 Y - 坐标
public:
    CShape() {m_type=0; m_xpos=0; m_ypos=0;} // constructor
    void      SetXPos(int x) {m_xpos=x;} // 设置 X
    void      SetYPos(int y) {m_ypos=y;} // 设置 Y
};
```

下一步，创建基本类衍生出的新类，这里我们可以添加说明类的必要的字段。对于圆形添加包括半径值构件是必须的。正方形以边值为特点。因此，由继承基本类CShape而衍生的类如下声明：

```
//--- 派生类 圆形
class CCircle : public CShape // 冒号后定义基本类
{                               // 从继承算法开始
private:
    int      m_radius;          // 圆弧半径
public:
    CCircle() {m_type=1;} // 构造函数，类型 1
};
```

正方形类声明类似：

```
//--- 派生类 方形
class CSquare : public CShape      // 冒号后定义基本类
{                                  // 从继承算法开始
private:
    int          m_square_side;    // 方形的边

public:
    CSquare() {m_type=2;} // 构造函数，类型2
};
```

注意对象创建时首先调用基本类构造函数，然后调用衍生类的[构造函数](#)。当对象毁坏时首先调用衍生类的[析构函数](#)，然后调用基本类析构函数。

因此，通过声明基本类中常用构件，我们可以在衍生类中添加额外构件，指定特殊类。继承算法允许创建多次重复使用的强大的代码函数库。

从现存类创建衍生类的句法如下：

```
class class_name :
    (public | protected | private) opt base_class_name
{
    class members declaration
};
```

衍生类一方面就是其构件的可见性（公开），继承人（继承）。关键字public，protected和private用于指定范围，该范围中基本类构件也对衍生类有效。衍生类表头中冒号后的Public关键字表明基本类CShape的protected和public构件应该继承为衍生类CCircle的protected和public构件。

基本类的private类构件对衍生类无效。public继承也意味着衍生类（CCircle and CSquare）就是 CShapes。也就是，正方形（CSquare）是一个图形（CShape），但是图形却不一定就是正方形。

衍生类是基本类的变体，它继承了基本类的protected和public构件。基本类的构造函数和析构函数不能继承。除了基本类的构件，新构件也会添加进衍生类中。

不同于基本类，衍生类包括执行构件函数。与[重载](#)无共同点，同名函数的意思会因签名不同而不同。

在protected继承中，基本类的public和protected构件成为衍生类的protected构件。在private继承中，基本类的public和protected构件成为衍生类的private构件。

在protected和private继承中，“衍生类对象就是基本类对象”的关系不是真的。protected和private继承类型也很少见，每一个都需要小心使用。

MQL5没有多继承算法。

另见

[架构和类](#)

多态性

多态性为不同分类对象提供了一个机会，通过继承，当调用相同函数元素时以不同的方式回应，通过帮助创造普遍原理表述行为不仅是基础类别，更是继承类别。

继续研发基础分类CShape，定义成员函数GetArea()，目的在于计算阴影区域，在所有继承分类里，通过从基础分类继承下来，我们重新定义这种函数以计算特殊阴影面积的规则相同

在一个区域内（CSquare分类），通过面积计算一圈（CCircle分类），区域通过半径等表示。创建数组存储CShape类型交易对象，基础类别中的量个交易品种和继承的交易分类都能存储，然后就可以调用数组中每个元素的任意相同函数。

示例：

```
//--- 基本类
class CShape
{
protected:
    int          m_type;           // 形状类型
    int          m_xpos;           // 基本点的X - 坐标
    int          m_ypos;           // 基本点的Y - 坐标
public:
    void          CShape() {m_type=0;}; // 构造函数，类型=0
    int          GetType() {return(m_type);}; // 返回 图形类型
    virtual
    double        GetArea() {return (0); } // 返回 图形区域
};
```

现在，所有的派生类都有成员函数 getArea()，返回0值，该函数以每个后代变化转变为基础实现。

```
//--- 派生类 圆形
class CCircle : public CShape // 冒号后定义基本类
{                               // 从继承算法开始
private:
    int          m_radius;       // 圆弧半径

public:
    void          CCircle() {m_type=1;}; // 构造函数，类型=1
    void          SetRadius(double r) {m_radius=r;};
    virtual double GetArea() {return (3.14*m_radius*m_radius);} // 圆形 区域
};
```

对于Square分类来说，申报是相同的：

```
//--- 派生类 方形
class CSquare : public CShape // 冒号后定义基本类
{                               // 从继承算法开始
private:
    int          m_square_side;  // 方形的边

public:
```



```
//--- тут мы намеренно "забыли" задать значение для shapes[2]
//circle=new CCircle();
//circle.SetRadius(10);
//shapes[2]=circle;

//--- для неиспользуемого элемента установим значение NULL
shapes[2]=NULL;

//--- создаем объект CSquare и запишем его указатель в shapes[3]
CSquare *square=new CSquare();
square.SetSide(5);
shapes[3]=square;

//--- создаем объект CSquare и запишем его указатель в shapes[4]
square=new CSquare();
square.SetSide(10);
shapes[4]=square;

//--- массив указателей есть, получим его размер
int total=ArraySize(shapes);
//--- пройдем в цикле по всем указателям в массиве
for(int i=0; i<5;i++)
{
    //--- если по указанному индексу указатель является валидным
    if(CheckPointer(shapes[i])!=POINTER_INVALID)
    {
        //--- выведем в лог тип и площадь фигуры
        PrintFormat("Объект типа %d имеет площадь %G",
            shapes[i].GetType(),
            shapes[i].GetArea());
    }
    //--- если указатель имеет тип POINTER_INVALID
    else
    {
        //--- сообщим об ошибке
        PrintFormat("Объект shapes[%d] не инициализирован! Его указатель %s",
            i,EnumToString(CheckPointer(shapes[i])));
    }
}

//--- мы должны самостоятельно уничтожить все созданные динамические объекты
for(int i=0;i<total;i++)
{
    //--- удалять можно только объекты, чей указатель имеет тип POINTER_DYNAMIC
    if(CheckPointer(shapes[i])==POINTER_DYNAMIC)
    {
        //--- сообщим об удалении
        PrintFormat("Удаляем shapes[%d]",i);
    }
}
```

```
    ///--- уничтожим объект по его указателю  
    delete shapes[i];  
}  
}  
}
```

[delete](#)
delete
[POINTER_DYNAMIC](#),

但此外，通过继承和多态性重新定义函数，包括一个工具和一个分离中建立的不同函数的不同参量，这表示该分类拥有几个相同名称的函数，但是不同类型的参量，因此，通过[function overload](#)实施多态性。

另见

[标准程序库](#)

重载

在一个类中定义两个或者以上同名类函数是可能的，但是会有不同数量的参量。当这个发生时，类函数就称为*重载*而这样的过程被称为*类函数重载*。

类函数重载是一种实现[多态](#)的方法。如[函数重载](#)一样类函数重载按照以下规则执行。

如果调用的函数无准确匹配，编译器就会在三个连续阶中搜索适合的函数：

1. 类函数中搜索；
2. 基本类函数中搜索，按照最近到最开始顺序。
3. 其他函数中搜索。

如果任何阶中都没有确切匹配的，但是在不同阶中找到几个适合的函数，使用最近阶找到的函数。一个阶中，只有一个适合的函数。

MQL5没有操作符重载。

另见

[重载函数](#)

虚拟函数

虚拟关键字是函数说明符，会根据原理选择动态的运行时间，并在基本或派生类别中找出恰当函数。架构中不能有虚拟函数，它只能用于改变函数[声明](#)。

虚拟函数，与一般函数一样，都需要 [可执行体](#)。当调用时，语义与其他函数一样

虚拟函数在派生类里可能会遭到拒绝，选择[函数定义](#)的种类需要虚拟函数的动态调用（在运行时间里），典型情况是，当基本类别包含虚拟函数时，派生类都含有函数里特有的版本。

指针指向基本类别里可以标明基本分类对象或者派生类对象，函数成员选择的调用会在运行时间内进行，会依靠类型对象执行，并不是指标的类型，如果没有派生类成员，数据分类的虚拟函数就是默认值。

无论是否用虚拟关键字描述，[析构函数](#)永远是[虚拟](#)的。

在MT5_ Tetris.mqh例子中虚拟函数的使用，虚拟函数 Draw 的基本类别CTetrisShape，定义在包含文件MT5_ TetisShape.mqh.中。

```
//+-----+
class CTetrisShape
{
protected:
    int          m_type;
    int          m_xpos;
    int          m_ypos;
    int          m_xsize;
    int          m_ysize;
    int          m_prev_turn;
    int          m_turn;
    int          m_right_border;
public:
    void          CTetrisShape();
    void          SetRightBorder(int border) { m_right_border=border; }
    void          SetYPos(int ypos)         { m_ypos=ypos;           }
    void          SetXPos(int xpos)          { m_xpos=xpos;           }
    int           GetYPos()                  { return(m_ypos);         }
    int           GetXPos()                  { return(m_xpos);         }
    int           GetYSize()                 { return(m_ysize);        }
    int           GetXSize()                 { return(m_xsize);        }
    int           GetType()                  { return(m_type);         }
    void          Left()                    { m_xpos-=SHAPE_SIZE;      }
    void          Right()                   { m_xpos+=SHAPE_SIZE;      }
    void          Rotate()                   { m_prev_turn=m_turn; if(++m_turn>3) }
    virtual void  Draw()                    { return;                  }
    virtual bool  CheckDown(int& pad_array[]);
    virtual bool  CheckLeft(int& side_row[]);
    virtual bool  CheckRight(int& side_row[]);
};
```

进一步说，对于每一个分类，该函数都会依照下一代分类的特性来执行，例如，第一个模型CTetrisShape1有自己

的执行函数Draw() :

```
class CTetrisShapel : public CTetrisShape
{
public:
    ///--- 绘画形状
    virtual void Draw()
    {
        int i;
        string name;
        ///---
        if(m_turn==0 || m_turn==2)
        {
            ///--- 水平线
            for(i=0; i<4; i++)
            {
                name=SHAPE_NAME+(string)i;
                ObjectSetInteger(0,name,OBJPROP_XDISTANCE,m_xpos+i*SHAPE_SIZE);
                ObjectSetInteger(0,name,OBJPROP_YDISTANCE,m_ypos);
            }
        }
        else
        {
            ///--- 垂直线
            for(i=0; i<4; i++)
            {
                name=SHAPE_NAME+(string)i;
                ObjectSetInteger(0,name,OBJPROP_XDISTANCE,m_xpos);
                ObjectSetInteger(0,name,OBJPROP_YDISTANCE,m_ypos+i*SHAPE_SIZE);
            }
        }
    }
}
```

方形台由CTetrisShape6 分类描述, 并实现Draw() 语法:

```
class CTetrisShape6 : public CTetrisShape
{
public:
    ///--- 绘画形状
    virtual void Draw()
    {
        int i;
        string name;
        ///---
        for(i=0; i<2; i++)
        {
            name=SHAPE_NAME+(string)i;
            ObjectSetInteger(0,name,OBJPROP_XDISTANCE,m_xpos+i*SHAPE_SIZE);
        }
    }
}
```

```
        ObjectSetInteger(0,name,OBJPROP_YDISTANCE,m_ypos);
    }
    for(i=2; i<4; i++)
    {
        name=SHAPE_NAME+(string)i;
        ObjectSetInteger(0,name,OBJPROP_XDISTANCE,m_xpos+(i-2)*SHAPE_SIZE);
        ObjectSetInteger(0,name,OBJPROP_YDISTANCE,m_ypos+SHAPE_SIZE);
    }
}
};
```

根据分类，建立对象归属，在派生类里调用虚拟函数。

```
void CTetrisField::NewShape()
{
//--- 创建7个随机形状中的一个
    int nshape=rand()%7;
    switch(nshape)
    {
        case 0: m_shape=new CTetrisShape1; break;
        case 1: m_shape=new CTetrisShape2; break;
        case 2: m_shape=new CTetrisShape3; break;
        case 3: m_shape=new CTetrisShape4; break;
        case 4: m_shape=new CTetrisShape5; break;
        case 5: m_shape=new CTetrisShape6; break;
        case 6: m_shape=new CTetrisShape7; break;
    }
//--- 绘画
    m_shape.Draw();
//---
}
```

另见

[标准程序库](#)

标准常量，枚举和架构

为了简化编写程序并使其程序文本更加方便，在MQL5语言中预定义的标准常量和表达式，此外，[结构](#)服务也用于存储信息。

[整型](#)的标准变量与宏命令类似。

此变量是按照用途分组的：

- [图表常量](#) 与价格图表共同使用：开盘，导航，设置参量；
- [对象常量](#) 是用来加工图表中添加和显示出来的图像对象；
- [指标常量](#) 用于标准和自定义指标；
- [环境状态](#) 描述MQL5程序的属性，显示客户端信息，安全性和活期存款账户；
- [交易常量](#) 在交易过程中允许说明各类型信息；
- [命名常量](#) 是MQL5语言中的常量；
- [数据结构](#) 描述数据存储格式；
- [错误和警告代码](#) 描述编译器信息和交易服务器发出的交易请求；
- [输入/输出常量](#) 是用来与[文件函数](#)一起工作，并在[MessageBox\(\)](#)函数显示器上显示信息。

图表常量

该常量描述了图表中个各种属性，并分成如下组：

- [事件类型](#) – 与图表工作时时间发生;
- [图表时间表](#) – 标准内置周期;
- [图表属性](#) – 用作[图表函数](#)参量的标识符;
- [定位常量](#) - [ChartNavigate\(\)](#) 函数的参量值；
- [展示图表](#) - 设置图表外观.

图表事件类型

使用预定函数OnChartEvent() 能实现9种事件类型，自定义事件65536标识符在CHARTEVENT_ CUSTOM 到 CHARTEVENT_ CUSTOM_ LAST 范围内浮动，为生成自定义事件，可以使用EventChartCustom() 函数。

ENUM_ CHART_ EVENT

ID	说明
CHARTEVENT_ KEYDOWN	按键次数
CHARTEVENT_ OBJECT_ CREATE	图解对象 创建 (CHART_ EVENT_ OBJECT_ CREATE=true)
CHARTEVENT_ OBJECT_ CHANGE	图解对象 通过属性对话改变性质
CHARTEVENT_ OBJECT_ DELETE	图解对象 删除 (CHART_ EVENT_ OBJECT_ DELETE=true)
CHARTEVENT_ CLICK	点击一个图表
CHARTEVENT_ OBJECT_ CLICK	点击 图解对象
CHARTEVENT_ OBJECT_ DRAG	拖放 图解对象
CHARTEVENT_ OBJECT_ ENDEDIT	在图表对象编辑里进行文本末尾编辑
CHARTEVENT_ CHART_ CHANGE	
CHARTEVENT_ CUSTOM	事件的首写字符在自定义事件中
CHARTEVENT_ CUSTOM_ LAST	事件的最后一个字符在自定义事件中

对于每个事件类型，OnChartEvent() 函数的输入参数都有固定值，这在此事件处理过程中是必须的，事件和值通过参量传递并列举在如下表格中

事件	ID参量值	参数常量值	Value of the dparam parameter	Value of the sparam parameter
按键事件	CHARTEVENT_ KEYDOWN	按键代码	—	—
图解对象创建事件 (CHARTEVENT_ OBJECT_ CREA TE	—	—	创建的图表对象的名称

CHART_EVENT_OBJECT_CREATE=true)				
通过性质对话转变对象性质事件	CHARTEVENT_OBJECT_CHANGE	—	—	修改的图表对象的名称
图解对象删除事件 (CHART_EVENT_OBJECT_DELETE=true)	CHARTEVENT_OBJECT_DELETE	-	-	删除图表对象的名称
鼠标点击图表事件	CHARTEVENT_CLICK	X坐标	Y坐标	—
鼠标点击属于图表的对象事件	CHARTEVENT_OBJECT_CLICK	X坐标	Y坐标	事件发生时图表对象的名称
使用鼠标拖动图解对象事件	CHARTEVENT_OBJECT_DRAG	—	—	移动图表对象名称
在编辑标签图表对象的进入访问完成文本编辑事件	CHARTEVENT_OBJECT_EDIT	—	—	在文本编辑完成后编辑标签图表对象名称
在N数字下用户使用ID事件	CHARTEVENT_CUSTOM+N	EventChartCustom() 函数的值	EventChartCustom() 函数的值	EventChartCustom() 函数的值

示例：

```

#define KEY_NUMPAD_5      12
#define KEY_LEFT         37
#define KEY_UP           38
#define KEY_RIGHT        39
#define KEY_DOWN         40
#define KEY_NUMLOCK_DOWN 98
#define KEY_NUMLOCK_LEFT 100
#define KEY_NUMLOCK_5    101
#define KEY_NUMLOCK_RIGHT 102
#define KEY_NUMLOCK_UP   104
//+-----+
//|  专家初始化函数          |
//+-----+
int OnInit()
{
//---
    Print("The expert with name ",MQL5InfoString(MQL5_PROGRAM_NAME)," is running");
//---
    return(0);
}
//+-----+
//|                          |
//+-----+
void OnChartEvent(const int id,          // 事件标识符
                  const long& lparam,    // 事件长整型参量
                  const double& dparam,  // 事件双精度型参量
                  const string& sparam   // 事件字符串型参量
                  )
{
//--- 鼠标左键点击图表
    if(id==CHARTEVENT_CLICK)
    {
        Print("The coordinates of the mouse click on the chart are: x = ",lparam," y = ");
    }
//--- 鼠标点击图形物件
    if(id==CHARTEVENT_OBJECT_CLICK)
    {
        Print("The mouse has been clicked on the object with name '"+sparam+"'");
    }
//--- 按下密钥
    if(id==CHARTEVENT_KEYDOWN)
    {
        switch(lparam)
        {
            case KEY_NUMLOCK_LEFT: Print("The KEY_NUMLOCK_LEFT has been pressed"); br
            case KEY_LEFT:         Print("The KEY_LEFT has been pressed");       br
            case KEY_NUMLOCK_UP:   Print("The KEY_NUMLOCK_UP has been pressed");  br
            case KEY_UP:           Print("The KEY_UP has been pressed");          br
            case KEY_NUMLOCK_RIGHT: Print("The KEY_NUMLOCK_RIGHT has been pressed"); br
            case KEY_RIGHT:        Print("The KEY_RIGHT has been pressed");       br
            case KEY_NUMLOCK_DOWN: Print("The KEY_NUMLOCK_DOWN has been pressed"); br
            case KEY_DOWN:         Print("The KEY_DOWN has been pressed");        br
            case KEY_NUMPAD_5:     Print("The KEY_NUMPAD_5 has been pressed");     br
            case KEY_NUMLOCK_5:    Print("The KEY_NUMLOCK_5 has been pressed");    br
            default:               Print("Some not listed key has been pressed");
        }
        ChartRedraw();
    }
//--- 物件被删除了
    if(id==CHARTEVENT_OBJECT_DELETE)
    {

```



```
        Print("The object with name ",sparam," has been deleted");
    }
//--- 物件被创建了
    if(id==CHARTEVENT_OBJECT_CREATE)
    {
        Print("The object with name ",sparam," has been created");
    }
//--- 移动物件或者更改定位点坐标
    if(id==CHARTEVENT_OBJECT_DRAG)
    {
        Print("The anchor point coordinates of the object with name ",sparam," has been
    }
//--- 物件编辑文本被更改
    if(id==CHARTEVENT_OBJECT_ENDEDIT)
    {
        Print("The text in the Edit field of the object with name ",sparam," has been c
    }
}
```

另见

[事件处理函数](#) - [工作事件](#)

图表时间表

所有预定义的图表时间表都有唯一的标识符，当MQL5程序运行时，PERIOD_ CURRENT 标识符代表图表本期。

ENUM_ TIMEFRAMES

ID	描述
PERIOD_ CURRENT	当前时间表
PERIOD_ M1	1 分钟
PERIOD_ M2	2 分钟
PERIOD_ M3	3 分钟
PERIOD_ M4	4 分钟
PERIOD_ M5	5 分钟
PERIOD_ M6	6 分钟
PERIOD_ M10	10 分钟
PERIOD_ M12	12 分钟
PERIOD_ M15	15 分钟
PERIOD_ M20	20 分钟
PERIOD_ M30	30 分钟
PERIOD_ H1	1小时
PERIOD_ H2	2 小时
PERIOD_ H3	3 小时
PERIOD_ H4	4 小时
PERIOD_ H6	6小时
PERIOD_ H8	8小时
PERIOD_ H12	12 小时
PERIOD_ D1	1 天
PERIOD_ W1	1 周
PERIOD_ MN1	1月

示例：

```
string chart_name="test_Object_Chart";
Print("Let's try to create a Chart object with the name ",chart_name);
//--- 如果没有这个物件- 创建它
if(ObjectFind(0,chart_name)<0)ObjectCreate(0,chart_name,OBJ_CHART,0,0,0,0,0);
```

```
//--- 定义交易品种
ObjectSetString(0,chart_name,OBJPROP_SYMBOL,"EURUSD");
//--- 设置定位点X坐标
ObjectSetInteger(0,chart_name,OBJPROP_XDISTANCE,100);
//--- 设置定位点Y坐标
ObjectSetInteger(0,chart_name,OBJPROP_YDISTANCE,100);
//--- 设置图表宽度
ObjectSetInteger(0,chart_name,OBJPROP_XSIZE,400);
//--- 设置高度
ObjectSetInteger(0,chart_name,OBJPROP_YSIZE,300);
//--- 设置时间表
ObjectSetInteger(0,chart_name,OBJPROP_PERIOD,PERIOD_D1);
//--- 设置规模(从 0 到 5)
ObjectSetInteger(0,chart_name,OBJPROP_SCALE,4);
//--- 禁止鼠标选择
ObjectSetInteger(0,chart_name,OBJPROP_SELECTABLE,false);
```

另见

[秒周期](#) [~周期](#) [~日期和时间](#) [~对象可见性](#)

图表属性

ENUM_CHART_PROPERTY 计数标识符与使用[工作图表函数](#)参量相同。在类型属性列里的缩写r/o表示只读，不能更改，当访问某一功能时，指定额外参数修饰语是有必要的，该修饰语提供了图表窗口的数量，0是主窗口。

函数ChartSetInteger()和ChartGetInteger()。

ENUM_CHART_PROPERTY_INTEGER

ID	描述	性质类型
CHART_BRING_TO_TOP		bool
CHART_EVENT_OBJECT_CREATE	mql5- (CHARTEVENT_OBJECT_CREATE)	bool
CHART_EVENT_OBJECT_DELETE	mql5-	bool

	(CHARTEVENT_OBJECT_DELETE)	
CHART_MODE	图表类型（蜡烛台、字节或线）	enum ENUM_CHART_MODE
CHART_FOREGROUND	背景的价格图表	布尔
CHART_SHIFT	左边缩进价格图表	布尔
CHART_AUTOSCROLL	自动移向图表的右边框	布尔
CHART_SCALE	测量	整型 从0到5
CHART_SCALEFIX	固定标盘模式	布尔
CHART_SCALEFIX_11	测量方式1:1	布尔
CHART_SCALE_PT_PER_BAR	每字节指定相关测量	布尔
CHART_SHOW_OHLC	在左上角显示OHLC值	布尔
CHART_SHOW_BID_LINE	在图表水平线上显示出价值	布尔
CHART_SHOW_ASK_LINE	在图表水平线上显示要价值	布尔
CHART_SHOW_LAST_LINE	在图表水平线上显示最终值	布尔
CHART_SHOW_PERIOD_SEP	在相邻周期显示垂直分离器	布尔
CHART_SHOW_GRID	在图表中显示网格	布尔
CHART_SHOW_VOLUMES	在图表中显示成交量	enum ENUM_CHART_VOLUME_MODE

CHART_SHOW_OBJECT_DESCR	弹出图解对象摘要	布尔
CHART_VISIBLE_BARS	图表上显示的字节数量	int r/o
CHART_WINDOWS_TOTAL	图表窗口总数，包括指标预览窗口	int r/o
CHART_WINDOW_IS_VISIBLE	预览窗口可见性	bool r/o 修饰符 - 子窗口号
CHART_WINDOW_HANDLE	处理图表窗口 (HWND)	int r/o
CHART_FIRST_VISIBLE_BAR	图表中第一可见字节字符。字节索引同于 时序列 。	int r/o
CHART_WIDTH_IN_BARS	以字节转发图表	int r/o
CHART_WIDTH_IN_PIXELS	像素转发图表	int r/o
CHART_HEIGHT_IN_PIXELS	图表像素高度	int r/o 修饰符 - 子窗口号
CHART_COLOR_BACKGROUND	图表背景颜色	颜色
CHART_COLOR_FOREGROUND	轴线、缩放和OHLC线的颜色	颜色
CHART_COLOR_GRID	网格颜色	颜色
CHART_COLOR_VOLUME	成交量颜色和开仓水平	颜色
CHART_COLOR_CHART_UP	上升字节、阴影和大型烛台整体边界的颜色	颜色
CHART_COLOR_CHART_DOWN	下降字节、阴影和支撑烛台的颜色	颜色
CHART_COLOR_CHART_LINE	折线图颜色和日语烛台躲闪颜色	颜色
CHART_COLOR_CANDLE_BULL	大型烛台主体颜色	颜色
CHART_COLOR_CANDLE_BEAR	承受烛台主体颜色	颜色

CHART_COLOR_BID	出价水平颜色	颜色
CHART_COLOR_ASK	要价水平颜色	颜色
CHART_COLOR_LAST	最后执行交易 价格水平线颜色 (Last)	颜色
CHART_COLOR_STOP_LEVEL	停止订购水平 颜色 (斩仓和获利)	颜色
CHART_SHOW_TRADE_LEVELS	在图表中显示 交易水平 (开仓水平、斩仓、获利和代办订单)	布尔
CHART_SHOW_DATE_SCALE		bool
CHART_SHOW_PRICE_SCALE		bool

函数 [ChartSetDouble\(\)](#) 和 [ChartGetDouble\(\)](#)

ENUM_CHART_PROPERTY_DOUBLE

ID	描述	性质类型
CHART_SHIFT_SIZE	大小在右边百分比 边界从零字节缩进	双精度 (10%-50%)
CHART_FIXED_MAX	固定图表最大值	双精度
CHART_FIXED_MIN	固定图表最小值	双精度
CHART_POINTS_PER_BAR	测量相关的每个字节	双精度
CHART_PRICE_MIN	图表最大值	double r/o 修饰符 - 子窗口号
CHART_PRICE_MAX	图表最大值	double r/o 修饰符 - 子窗口号

[ChartSetString\(\)](#) 和 [ChartGetString\(\)](#) 函数

ENUM_CHART_PROPERTY_STRING

ID	描述	性质类型
CHART_COMMENT	图表中的评论文本	字符串

示例：

```
int chartMode=ChartGetInteger(0,CHART_MODE);
switch(chartMode)
{
    case(CHART_BARS):    Print("CHART_BARS");    break;
    case(CHART_CANDLES): Print("CHART_CANDLES");break;
    default:Print("CHART_LINE");
}
bool shifted=ChartGetInteger(0,CHART_SHIFT);
if(shifted) Print("CHART_SHIFT = true");
else Print("CHART_SHIFT = false");
bool autoscroll=ChartGetInteger(0,CHART_AUTOSCROLL);
if(autoscroll) Print("CHART_AUTOSCROLL = true");
else Print("CHART_AUTOSCROLL = false");
int chartHandle=ChartGetInteger(0,CHART_WINDOW_HANDLE);
Print("CHART_WINDOW_HANDLE = ",chartHandle);
int windows=ChartGetInteger(0,CHART_WINDOWS_TOTAL);
Print("CHART_WINDOWS_TOTAL = ",windows);
if(windows>1)
{
    for(int i=0;i<windows;i++)
    {
        int height=ChartGetInteger(0,CHART_HEIGHT_IN_PIXELS,i);
        double priceMin=ChartGetDouble(0,CHART_PRICE_MIN,i);
        double priceMax=ChartGetDouble(0,CHART_PRICE_MAX,i);
        Print(i+": CHART_HEIGHT_IN_PIXELS = ",height," pixels");
        Print(i+": CHART_PRICE_MIN = ",priceMin);
        Print(i+": CHART_PRICE_MAX = ",priceMax);
    }
}
```


定位常量

ENUM_ CHART_ POSITION 列表中的三个标识符有可能是[ChartNavigate\(\)](#)函数的方位参量的值。

ENUM_ CHART_ POSITION

ID	描述
CHART_ BEGIN	图表开始（最原始价位）
CHART_ CURRENT_ POS	当前位置
CHART_ END	图表末尾（当前价格）

示例：

```
long handle=ChartOpen("EURUSD",PERIOD_H12);
if(handle!=0)
{
    ChartSetInteger(handle,CHART_AUTOSCROLL,false);
    ChartSetInteger(handle,CHART_SHIFT,true);
    ChartSetInteger(handle,CHART_MODE,CHART_LINE);
    ResetLastError();
    bool res=ChartNavigate(handle,CHART_END,150);
    if(!res) Print("Navigate failed. Error = ",GetLastError());
    ChartRedraw();
}
```

图表陈述

价格图表可以以下三种方式陈列：

- 字符；
- 蜡烛台；
- 反射线。

显示价格图表的特殊方式是通过[ChartSetInteger](#)(chart__handle,[CHART__MODE](#),chart__ mode) ,函数建立 , chart__ mode是ENUM_ CHART_ MODE计算式中的一个值。

ENUM_ CHART_ MODE

ID	描述
CHART_ BARS	以序列字符陈列
CHART_ CANDLES	以日语蜡烛台陈列
CHART_ LINE	收盘价格线型显示

在价格表格上显示成交量的特殊方式是通过使用[ChartSetInteger](#)(chart__handle,[CHART_ SHOW_ VOLUMES](#),volume__ mode) 函数 , volume__ mode是计算式ENUM_ CHART_ VOLUME_ MODE中值的一个。

ENUM_ CHART_ VOLUME_ MODE

ID	描述
CHART_ VOLUME_ HIDE	成交量隐藏
CHART_ VOLUME_ TICK	最小价格成交量
CHART_ VOLUME_ REAL	交易成交量

示例：

```
//--- 获得当前图表处理权
long handle=ChartID();
if(handle>0) // 如果成功，加上自定义
{
    //--- 禁止自动滚动
    ChartSetInteger(handle,CHART_AUTOSCROLL,false);
    //--- 设置图表右缩进
    ChartSetInteger(handle,CHART_SHIFT,true);
    //--- 显示蜡烛图
    ChartSetInteger(handle,CHART_MODE,CHART_CANDLES);
    //--- 从历史记录起始位置按100柱为一页滚动
    ChartNavigate(handle,CHART_CURRENT_POS,100);
    //--- 设置订单交易量显示模式
    ChartSetInteger(handle,CHART_SHOW_VOLUMES,CHART_VOLUME_TICK);
}
```

另见：

[打开图表](#)， [图表ID](#)

对象常量

可以创建39个图解对象并在价格图表中显示出来，所有与对象有关的常量都被分成9组：

- [对象类型](#) – 图解对象的标识符；
- [对象属性](#) – 建立和获得图解对象的性质；
- [对象定位方法](#) – 在图表中定位对象常量；
- [定位角](#) – 指出时间表的角度，在对象上定位；
- [对象可见性](#) – 对象可见时建立时间表；
- [埃利奥特波动水平](#) – 波浪渐变标记；
- [江恩对象](#) – 江恩角度线盒网格的趋向常量；
- [网页颜色](#) – 预定义网络颜色的常量；
- [Wingdings](#) – Wingdings字形的字符代码

物件类型

当使用 `ObjectCreate()` 函数建立图表对象，对于增加的对象类型有必要说明一下，它有可能是 `ENUM_OBJECT` 计算式中的一个，对象的下一项规格属性有可能使用与 `图线对象` 一同使用

`ENUM_OBJECT`

ID		描述
<code>OBJ_VLINE</code>		垂直线
<code>OBJ_HLINE</code>	—	水平线
<code>OBJ_TREND</code>	/	趋势线
<code>OBJ_TRENDBYANGLE</code>	△	趋势角度
<code>OBJ_CHANNEL</code>	≡	等距离通道
<code>OBJ_STDDEVCHANNEL</code>	≡	标准偏离通道
<code>OBJ_REGRESSION</code>	↗	线形回归通道
<code>OBJ_PITCHFORK</code>	///	安德鲁分叉线
<code>OBJ_GANNLIN</code>	/G	江恩线
<code>OBJ_GANNFAN</code>	⦿G	江恩扇形线
<code>OBJ_GANNGRID</code>	⦿G	江恩网格线
<code>OBJ_FIBO</code>	⦿F	斐波纳契撤回
<code>OBJ_FIBOTIMES</code>	⦿F	斐波纳契时间周期线
<code>OBJ_FIBOFAN</code>	⦿F	斐波纳契扇形线
<code>OBJ_FIBOARC</code>	⦿F	斐波纳契弧线
<code>OBJ_FIBOCHANNEL</code>	⦿F	斐波纳契通道
<code>OBJ_EXPANSION</code>	⦿F	斐波纳契扩展
<code>OBJ_ELLIOTWAVE5</code>	⦿E	埃利奥特动机波动
<code>OBJ_ELLIOTWAVE3</code>	⦿F	埃利奥特修正波动
<code>OBJ_RECTANGLE</code>	■	巨型
<code>OBJ_TRIANGLE</code>	▲	三角形
<code>OBJ_ELLIPSE</code>	●	椭圆形
<code>OBJ_CYCLES</code>	⦿	周期线
<code>OBJ_ARROW_THUMB_UP</code>	👍	大拇指向上
<code>OBJ_ARROW_THUMB_DOWN</code>	👎	大拇指向下
<code>OBJ_ARROW_UP</code>	↑	向上箭头
<code>OBJ_ARROW_DOWN</code>	↓	向下箭头

OBJ_ARROW_STOP		停止标志
OBJ_ARROW_CHECK		检测标志
OBJ_ARROW_LEFT_PRICE		左侧定价标签
OBJ_ARROW_RIGHT_PRICE		右侧定价标签
OBJ_ARROW_BUY		买入标志
OBJ_ARROW_SELL		卖出标志
OBJ_ARROW		箭头
OBJ_TEXT		文本
OBJ_LABEL		标签
OBJ_BUTTON		按钮
OBJ_CHART		图表
OBJ_BITMAP		位图
OBJ_BITMAP_LABEL		位图标签
OBJ_EDIT		编辑
OBJ_ARROWED_LINE		箭头线路
OBJ_EVENT		“事件”对象在经济日历对应一个事件
OBJ_RECTANGLE_LABEL		“ ”

对象属性

在价格图表中的每个图解型对象都有确定的建立属性，对象属性的值是通过 [图解对象工作函数](#) 传输的。每个 [对象类型](#) 对象类型都有自己的属性，ENUM_ OBJECT_ PROPERTY 计算式中的可用值都罗列出来。一些属性需要说明，诸如斐波纳契延长对象。在 [ObjectSet...\(\)](#) 和 [ObjectGet...\(\)](#) 函数中很有必要说明 modifier 参量值。

[ObjectSetInteger\(\)](#) 和 [ObjectGetInteger\(\)](#) 函数

ENUM_ OBJECT_ PROPERTY_ INTEGER

标识符	描述	性质类型
OBJPROP_ COLOR	颜色	颜色
OBJPROP_ STYLE	类型	ENUM_ LINE_ STYLE
OBJPROP_ WIDTH	路线厚度	整型
OBJPROP_ BACK	显示背景上的对象	布尔
OBJPROP_ FILL	(OBJ _ RE CTA NGL E, OBJ _ TRI ANG LE, OBJ _ EL LIPS E, OBJ	bool

	__CHANNEL, OBJECT_STATE_DEVEL, OBJECT_REGISTER_SECTION)	
OBJPROP_SELECTED	所选对象	布尔
OBJPROP_READONLY	在编辑对象上编辑文本	布尔
OBJPROP_TYPE	对象类型	ENUM_OBJECT
OBJPROP_TIME	时间坐标	日期时间修饰符=定位点数字
OBJPROP_SELECTABLE	对象可用性	布尔
OBJPROP_CREATETIME	创建对象时间	datetime r/o
OBJPROP_LEVELS	水平数量	整型
OBJPROP_LEVELCOLOR	线路水平颜色	颜色修饰符=水平数字
OBJPROP_LEVELSTYLE	线路水平风格	ENUM_LINE_STYLE 修饰符=水平数字
OBJPROP_LEVELWIDTH	线路水平厚度	int 修饰符=水平数字
OBJPROP_FONTSIZE	字体大小	整型
OBJPROP_RAY_LEFT	向左	布尔

	发出射线	
OBJPROP_RAY_RIGHT	向右发出射线	布尔
OBJPROP_RAY		bool
OBJPROP_ELLIPSE		布尔

	"	
	" (OBJ_FIB_OAR_C)	
OBJPROP_ARROWCODE	箭头对象代码	图表
OBJPROP_TIMEFRAMES	时间表上的对象可见性	set of flags flags
OBJPROP_ANCHOR	图解对象的定位点	ENUM_ARROW_ANCHOR (for OBJ_ARROW) , ENUM_ANCHORPOINT (for OBJ_LABEL and OBJ_TEXT)
OBJPROP_XDISTANCE	X轴定位点像素距离	整型
OBJPROP_YDISTANCE	Y轴定位点像素距离	整型
OBJPROP_DIRECTION	江恩物件趋势	ENUM_GANN_DIRECTION
OBJPROP_DEGREE	埃利奥特波动标记水平	ENUM_ELLIOT_WAVE_DEGREE
OBJPROP_DRAWLINES	埃利奥特波动标记排成一行	布尔
OBJPROP_STATE	按钮	布尔

	状态 (按 压/未 按 压)	
OBJPROP_ CHART_ ID	<div>" " (<u>OBJ</u> <u>_CH</u> <u>ART</u>) .</div>	long

	<div></div>	
OBJPROP_XSIZE	X轴的 图解 对象 大小 (像 素宽 度)	整型

	<pre>" " " (OBJ _BIT MAP _LABEL BEL OBJ _BIT MAP) .</pre>	
--	---	--

	.	
OBJPROP_YOFFSET	X-	int
	<div></div>	

	<div>"</div> <div>"</div> <div>"</div> <div>" (</div> <div>OBJ</div> <div>_BIT</div> <div>MAP</div> <div>_LA</div> <div>BEL</div> <div>OBJ</div> <div>_BIT</div> <div>MAP</div> <div>) .</div>	
--	--	--

	.	
OBJPROP_PERIOD	图解 对象 时间 表	ENUM_TIMEFRAMES
OBJPROP_DATE_SCALE	显示 图解 对象 时间 比例	布尔
OBJPROP_PRICE_SCALE	显示 图解 对象 价格 比例	布尔
OBJPROP_CHART_SCALE	图解 对象 测量	0-5的整数值
OBJPROP_BGCOLOR	编辑 对象 背景 颜色 OBJ _ EDI T, OBJ _ BU TTO N, OBJ _ RE CTA NGL E_ L ABE L	颜色

```

• ChartClose\( \);
• /
ChartSetSymbolPeriod\( \);
• CHART_SCALE, CHART_BRING_TO_TOP,
  CHART_SHOW_DATE_SCALE CHART_SHOW_PRICE_SCALE (
ENUM\_CHART\_PROPERTY\_INTEGER) .

  OBJ\_BITMAP\_LABEL OBJ\_BITMAP

  ,
  ,
  .

  OBJPROP_XSIZE OBJPROP_YSIZE.
  " "
  OBJPROP_XOFFSET
  OBJPROP_YOFFSET.

```

[ObjectSetDouble\(\)](#) 和 [ObjectGetDouble\(\)](#) 函数

ENUM_OBJECT_PROPERTY_DOUBLE

标识符	描述	属性类型
OBJPROP_PRICE	价格坐标	双精度 修饰符=定位点号
OBJPROP_LEVELVALUE	水平值	双精度 修饰符=水平号
OBJPROP_SCALE	比例（江恩对象和斐波纳契弧线）	双精度
OBJPROP_ANGLE	角度	双精度
OBJPROP_DEVIATION	标准偏差通道误差	双精度

[ObjectSetString\(\)](#) 和 [ObjectGetString\(\)](#) 函数

ENUM_OBJECT_PROPERTY_STRING

标识符	描述	属性类型
OBJPROP_NAME	物件名称	字符串
OBJPROP_TEXT	物件描述（物件中包含的文本）	字符串
OBJPROP_TOOLTIP		string

	"\n" ()	
OBJPROP_LEVELTEXT	水平描述	字符串 修饰符=水平号
OBJPROP_FONT	字体	字符串
OBJPROP_BITMAPFILE	文件的位图标签名称	字符串 修饰符: 0-state ON, 1-state OFF
OBJPROP_SYMBOL	图表对象的交易品种	字符串

OBJ_RECTANGLE_LABEL (" ")
,
ENUM_BORDER_TYPE.

ENUM_BORDER_TYPE

BORDER_FLAT	
BORDER_RAISED	
BORDER_SUNKEN	

示例：

```
#define UP          "\x0431"

//+-----+
//|  脚本程序启动函数          |
//+-----+
void OnStart()
{
//---
    string label_name="my_OBJ_LABEL_object";
    if(ObjectFind(0,label_name)<0)
    {
        Print("Object ",label_name," not found. Error code = ",GetLastError());
        //--- 创建标签物件
        ObjectCreate(0,label_name,OBJ_LABEL,0,0,0);
        //--- 设置X坐标
        ObjectSetInteger(0,label_name,OBJPROP_XDISTANCE,200);
        //--- 设置Y坐标
        ObjectSetInteger(0,label_name,OBJPROP_YDISTANCE,300);
        //--- 定义文本颜色
        ObjectSetInteger(0,label_name,OBJPROP_COLOR,clrWhite);
        //--- 定义标签物件文本
```

```
ObjectSetString(0,label_name,OBJPROP_TEXT,UP);  
//--- 定义字体  
ObjectSetString(0,label_name,OBJPROP_FONT,"Wingdings");  
//--- 定义字体大小  
ObjectSetInteger(0,label_name,OBJPROP_FONTSIZE,10);  
//--- 顺时针45度  
ObjectSetDouble(0,label_name,OBJPROP_ANGLE,-45);  
//--- 禁止鼠标选择  
ObjectSetInteger(0,label_name,OBJPROP_SELECTABLE,false);  
//--- 图表上绘画  
ChartRedraw(0);  
}  
}
```

Methods of Object Binding

图解对象文本和标签(OBJ_ TEXT 和OBJ_ LABEL) 有9种不同的方法串联定位，使用[ObjectSetInteger](#)函数规定必要变体(chart_ handle, object_ name, OBJPROP_ ANCHOR, anchor_ point_ mode) , anchor_ point_ mode 是ENUM_ ANCHORPOINT值中的一个。

ENUM_ ANCHOR_ POINT

ID	描述
ANCHOR_ LEFT_ UPPER	左上角定位点
ANCHOR_ LEFT	左侧定位点
ANCHOR_ LEFT_ LOWER	左下角定位点
ANCHOR_ LOWER	下方定位点
ANCHOR_ RIGHT_ LOWER	右下方定位点
ANCHOR_ RIGHT	右侧定位点
ANCHOR_ RIGHT_ UPPER	右上角定位点
ANCHOR_ UPPER	上方定位点
ANCHOR_ CENTER	对象中间严格定位点

示例：

```

string text_name="my_OBJ_TEXT_object";
if (ObjectFind(0,text_name)<0)
{
    Print("Object ",text_name," not found. Error code = ",GetLastError());
    //-- 获得图表最大价格
    double chart_max_price=ChartGetDouble(0,CHART_PRICE_MAX,0);
    //-- 创建物件标签
    ObjectCreate(0,text_name,OBJ_TEXT,0,TimeCurrent(),chart_max_price);
    //-- 设置文本颜色
    ObjectSetInteger(0,text_name,OBJPROP_COLOR,clrWhite);
    //-- 设置背景色
    ObjectSetInteger(0,text_name,OBJPROP_BGCOLOR,clrGreen);
    //-- 设置标签物件文本
    ObjectSetString(0,text_name,OBJPROP_TEXT,TimeToString(TimeCurrent()));
    //-- 设置文本字体
    ObjectSetString(0,text_name,OBJPROP_FONT,"Trebuchet MS");
    //-- 设置字体大小
    ObjectSetInteger(0,text_name,OBJPROP_FONTSIZE,10);
    //-- 定位右上角
    ObjectSetInteger(0,text_name,OBJPROP_ANCHOR,ANCHOR_RIGHT_UPPER);
    //-- 顺时针方向旋转90度
    ObjectSetDouble(0,text_name,OBJPROP_ANGLE,90);
}

```

```
//--- 禁止鼠标选择物件
ObjectSetInteger(0,text_name,OBJPROP_SELECTABLE,false);
//--- 重画物件
ChartRedraw(0);
}
```

图解对象箭头(OBJ_ ARROW) 连接到坐标有两种方式，标识符连接到ENUM_ ARROW_ ANCHOR。

ENUM_ ARROW_ ANCHOR

ID	描述
ANCHOR_ TOP	上部定位点
ANCHOR_ BOTTOM	下部定位点

示例：

```
void OnStart()
{
//--- 辅助数组
double Ups[],Downs[];
datetime Time[];
//--- 设置数组为时间序列
ArraySetAsSeries(Ups,true);
ArraySetAsSeries(Downs,true);
ArraySetAsSeries(Time,true);
//--- 创建指标iFractals句柄
int FractalsHandle=iFractals(NULL,0);
Print("FractalsHandle = ",FractalsHandle);
//--- 设置最近的错误值到零
ResetLastError();
//--- 复制指标值
int copied=CopyBuffer(FractalsHandle,0,0,1000,Ups);
if(copied<=0)
{
Print("Unable to copy the upper fractals. Error = ",GetLastError());
return;
}

ResetLastError();
//--- 复制指标值
copied=CopyBuffer(FractalsHandle,1,0,1000,Downs);
if(copied<=0)
{
Print("Unable to copy the bottom fractals. Error = ",GetLastError());
return;
}

ResetLastError();
//--- 复制包括最近1000开盘柱的时间序列
```

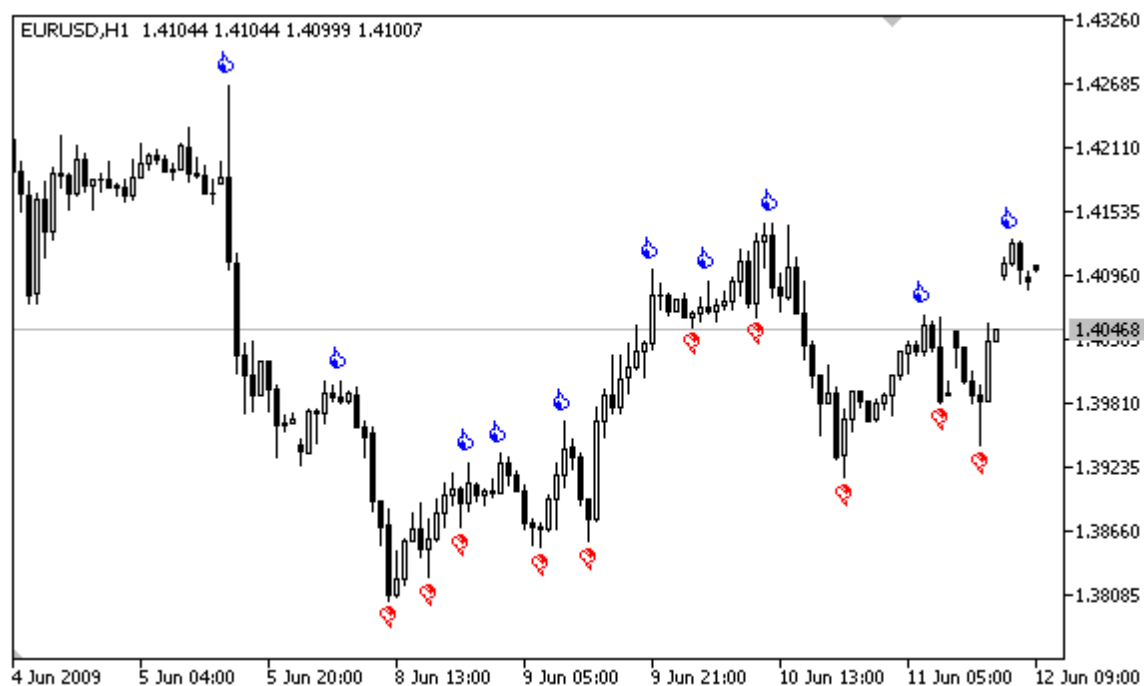
```

copied=CopyTime(NULL,0,0,1000,Time);
if(copied<=0)
{
    Print("Unable to copy the Opening Time of the last 1000 bars");
    return;
}

int upcounter=0,downcounter=0; // 数箭头数
bool created;// 接收尝试结果去创建物件
for(int i=2;i<copied;i++)// 运行通过指标i Fractal s值
{
    if(Ups[i]!=EMPTY_VALUE)// 找到上面的分数据
    {
        if(upcounter<10)// 创建少于10个 "向上" 箭头
        {
            //--- 创建 "向上" 物件
            created=ObjectCreate(0,string(Time[i]),OBJ_ARROW_THUMB_UP,0,Time[i],Ups[i]
            if(created)// 若设置 - 调节它
            {
                //--- 定位点在下面为了不覆盖柱
                ObjectSetInteger(0,string(Time[i]),OBJPROP_ANCHOR,ANCHOR_BOTTOM);
                //--- 最后接触-绘画
                ObjectSetInteger(0,string(Time[i]),OBJPROP_COLOR,clrBlue);
                upcounter++;
            }
        }
    }
    if(Downs[i]!=EMPTY_VALUE)// 发现较低的分数据
    {
        if(downcounter<10)// 创建少于10个 "向下" 箭头
        {
            //--- 创建 "向下" 物件
            created=ObjectCreate(0,string(Time[i]),OBJ_ARROW_THUMB_DOWN,0,Time[i],Dow
            if(created)// 若设置 - 调节它
            {
                //--- 定位点在上面为了不覆盖柱
                ObjectSetInteger(0,string(Time[i]),OBJPROP_ANCHOR,ANCHOR_TOP);
                //--- 最后接触-绘画
                ObjectSetInteger(0,string(Time[i]),OBJPROP_COLOR,clrRed);
                downcounter++;
            }
        }
    }
}
}

```

在脚本执行完毕后，图表看起来像这样。



附加物件的图表角落

存在图解说对象数量，可以设置图表内角，相对于坐标指定像素，有如下物件类型（括号内的对象标识符是规定的）：

- Label (OBJ__LABEL) ；
- Button (OBJ__BUTTON) ；
- Chart (OBJ__CHART) ；
- Bitmap Label (OBJ__BITMAP__LABEL) ；
- Rectangle Label (OBJ__RECTANGLE__LABEL) ；
- Edit (OBJ__EDIT) .

为了指定图表内角，X和Y坐标都要测量像素，使用ObjectSetInteger(chartID,name,OBJPROP_ CORNER, chart__corner)：

- 图表ID- 图表标识符；
- 名字 – 图解说对象名称；
- OBJPROP_ CORNER – 为捆绑指定一角的性质ID ；
- chart__corner – 目标图表，可以是 ENUM_ BASECORNER 计算式中的一个。

ENUM_ BASE_ CORNER

ID	描述
CORNER_ LEFT_ UPPER	中央坐标在图表左上角
CORNER_ LEFT_ LOWER	中央左边在图表左下角
CORNER_ RIGHT_ LOWER	中央坐标在图表右下角
CORNER_ RIGHT_ UPPER	中央左边在图表右上角

示例：

```
void CreateLabel(long chart_id,
                string name,
                int chart_corner,
                string text_label,
                int x_ord,
                int y_ord)
{
    //---
    ObjectCreate(chart_id,name,OBJ_LABEL,0,0,0);
    ResetLastError();
    if(!ObjectSetInteger(chart_id,name,OBJPROP_CORNER,chart_corner))
        Print("Unable to set the angle to bind the object ",
              name, ", error code ",GetLastError());
    ObjectSetInteger(chart_id,name,OBJPROP_XDISTANCE,x_ord);
    ObjectSetInteger(chart_id,name,OBJPROP_YDISTANCE,y_ord);
    ObjectSetString(chart_id,name,OBJPROP_TEXT,text_label);
}
```

```
//+-----+
//| 脚本程序启动函数 |
//+-----+
void OnStart()
{
//---
    int height=ChartGetInteger(0,CHART_HEIGHT_IN_PIXELS,0);
    int width=ChartGetInteger(0,CHART_WIDTH_IN_PIXELS,0);
    string arrows[4]={"LEFT_UPPER","RIGHT_UPPER","RIGHT_LOWER","LEFT_LOWER"};
    CreateLabel(0,arrows[0],CORNER_LEFT_UPPER,"0",50,50);
    CreateLabel(0,arrows[1],CORNER_RIGHT_UPPER,"1",50,50);
    CreateLabel(0,arrows[2],CORNER_RIGHT_LOWER,"2",50,50);
    CreateLabel(0,arrows[3],CORNER_LEFT_LOWER,"3",50,50);
}
```

对象可见性

对象组合可见标记决定图表时间表，该对象是明显的，为了建立或者得到OBJPROP_ TIMEFRAMES类型值，可以使用 [ObjectSetInteger\(\)](#) / [ObjectGetInteger\(\)](#) 函数。

ID	值	描述
OBJ_ NO_ PERIODS	0	时间表中不引用对象
OBJ_ PERIOD_ M1	0x00000001	1分钟图表显示
OBJ_ PERIOD_ M2	0x00000002	2分钟图表显示
OBJ_ PERIOD_ M3	0x00000004	3分钟图表显示
OBJ_ PERIOD_ M4	0x00000008	4分钟图表显示
OBJ_ PERIOD_ M5	0x00000010	5分钟图表显示
OBJ_ PERIOD_ M6	0x00000020	6分钟图表显示
OBJ_ PERIOD_ M10	0x00000040	10分钟图表显示
OBJ_ PERIOD_ M12	0x00000080	12分钟图表显示
OBJ_ PERIOD_ M15	0x00000100	15分钟图表显示
OBJ_ PERIOD_ M20	0x00000200	20分钟图表显示
OBJ_ PERIOD_ M30	0x00000400	30分钟图表显示
OBJ_ PERIOD_ H1	0x00000800	1小时图表显示
OBJ_ PERIOD_ H2	0x00001000	2小时图表显示
OBJ_ PERIOD_ H3	0x00002000	3小时图表显示
OBJ_ PERIOD_ H4	0x00004000	4小时图表显示
OBJ_ PERIOD_ H6	0x00008000	6小时图表显示
OBJ_ PERIOD_ H8	0x00010000	8小时图表显示
OBJ_ PERIOD_ H12	0x00020000	12小时图表显示
OBJ_ PERIOD_ D1	0x00040000	以天为单位的图表显示
OBJ_ PERIOD_ W1	0x00080000	以周为单位的图表显示
OBJ_ PERIOD_ MN1	0x00100000	以月为单位的图表显示
OBJ_ ALL_ PERIODS	0x001fffff	所有时间表对象显示

可见性标记可以通过使用符号"|"来建立，例如，标记组合OBJ_ PERIOD_ M10|OBJ_ PERIOD_ H4 表示该对象在10分钟到4小时这个时间段内可见

示例：

```
void OnStart ()
{
```

```
//---
string highlevel="PreviousDayHigh";
string lowlevel="PreviousDayLow";
double prevHigh;           // 前一天最高价
double prevLow;            // 前一天最低价
double highs[],lows[];     // 最高和最低价数组

//--- 重置上一个错误
ResetLastError();
//--- 获得日常时间表最近的2个最高值
int highsgot=CopyHigh(Symbol(),PERIOD_D1,0,2,highs);
if(highsgot>0) // 如果复制成功
{
    Print("High prices for the last 2 days were obtained successfully");
    prevHigh=highs[0]; // 前一天最高价
    Print("prevHigh = ",prevHigh);
    if(ObjectFind(0,highlevel)<0) // 没有找到名为highlevel物件
    {
        ObjectCreate(0,highlevel,OBJ_HLINE,0,0,0); // 创建水平线物件
    }
    //--- 为高位线价格平面设置值
    ObjectSetDouble(0,highlevel,OBJPROP_PRICE,0,prevHigh);
    //--- 仅设置PERIOD_M10和PERIOD_H4的可见性
    ObjectSetInteger(0,highlevel,OBJPROP_TIMEFRAMES,OBJ_PERIOD_M10|OBJ_PERIOD_H4);
}
else
{
    Print("Could not get High prices over the past 2 days, Error = ",GetLastError())
}

//--- 重置上一个错误
ResetLastError();
//--- 获得日常时间表最近的2个最低值
int lowsgot=CopyLow(Symbol(),PERIOD_D1,0,2,lows);
if(lowsgot>0) // 如果复制成功
{
    Print("Low prices for the last 2 days were obtained successfully");
    prevLow=lows[0]; // 前一天最低价
    Print("prevLow = ",prevLow);
    if(ObjectFind(0,lowlevel)<0) // 没有找到名为lowlevel物件
    {
        ObjectCreate(0,lowlevel,OBJ_HLINE,0,0,0); // 创建水平线物件
    }
    //--- 为低位线价格平面设置值
    ObjectSetDouble(0,lowlevel,OBJPROP_PRICE,0,prevLow);
    //--- 仅设置PERIOD_M10和PERIOD_H4的可见性
    ObjectSetInteger(0,lowlevel,OBJPROP_TIMEFRAMES,OBJ_PERIOD_M10|OBJ_PERIOD_H4);
}
```

```
else Print("Could not get Low prices for the last 2 days, Error = ", GetLastError())

ChartRedraw(0); // 强制重画图表
}
```

另见

[秒周期](#)、[周期](#)、[图表时间表](#)，[日期和时间](#)

埃利奥特波动水平

埃利奥特波动是由两个图解对象类型OBJ_ ELLIOTWAVE5和OBJ_ ELLIOTWAVE3表示，为创建波动大小（波动标签方法），使用OBJPROP_ DEGREE 属性，计算式ENUM_ ELLIOT_ WAVE_ DEGREE中的值可以表示出来。

ENUM_ ELLIOT_ WAVE_ DEGREE

ID	描述
ELLIOTT_ GRAND_ SUPERCYCLE	庞大的超周期
ELLIOTT_ SUPERCYCLE	超周期
ELLIOTT_ CYCLE	循环
ELLIOTT_ PRIMARY	原色
ELLIOTT_ INTERMEDIATE	媒介
ELLIOTT_ MINOR	次要的
ELLIOTT_ MINUTE	分钟
ELLIOTT_ MINUETTE	波浪
ELLIOTT_ SUBMINUETTE	潜波浪

示例：

```

for(int i=0;i<ObjectsTotal(0);i++)
{
    string currobj=ObjectName(0,i);
    if((ObjectGetInteger(0,currobj,OBJPROP_TYPE)==OBJ_ELLIOTWAVE3) ||
        ((ObjectGetInteger(0,currobj,OBJPROP_TYPE)==OBJ_ELLIOTWAVE5)))
    {
        //--- 在INTERMEDIATE设置标记水平
        ObjectSetInteger(0,currobj,OBJPROP_DEGREE,ELLIOTT_INTERMEDIATE);
        //--- 显示波峰间的线
        ObjectSetInteger(0,currobj,OBJPROP_DRAWLINES,true);
        //--- 设置线颜色
        ObjectSetInteger(0,currobj,OBJPROP_COLOR,clrBlue);
        //--- 设置线宽
        ObjectSetInteger(0,currobj,OBJPROP_WIDTH,5);
        //--- 设置描述
        ObjectSetString(0,currobj,OBJPROP_TEXT,"test script");
    }
}

```

江恩物件

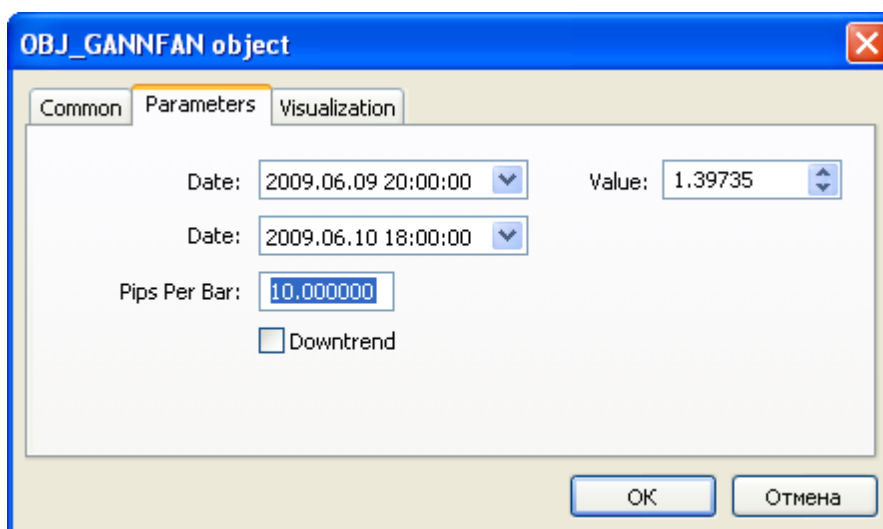
对于 江恩角度线 (OBJ_ GANNFAN) 和 江恩网格 (OBJ_ GANNGRID) 来说，ENUM_ GANN_ DIRECTION 中的两种值可以指定出来，会在开头建立。

ENUM_ GANN_ DIRECTION

ID	描述
GANN_ UP_ TREND	与上升趋势一致的线
GANN_ DOWN_ TREND	与下降趋势一致的线

建立主干线1x1规模，使用 [ObjectSetDouble](#) 函数(chart_ handle, gann_ object_ name, OBJPROP_ SCALE, scale)：

- chart_ handle – 对象加载的图表窗口；
- gann_ object_ name – 对象名称；
- OBJPROP_ SCALE – “Scale” 属性标识符；
- 比例 – 需要以Pips/Bar为单位。



创建江恩角度线示例：

```
void OnStart ()
{
//---
string my_gann="OBJ_ GANNFAN object";
if (ObjectFind (0,my_gann)<0)// 物件没找到
{
//--- 通知失败
Print ("Object ",my_gann," not found. Error code = ",GetLastError());
//--- 获得图表最大价格
double chart_max_price=ChartGetDouble (0,CHART_PRICE_MAX,0);
//--- 获得图表最小价格
double chart_min_price=ChartGetDouble (0,CHART_PRICE_MIN,0);
//--- 多少柱显示在图表上？
```



```

int bars_on_chart=ChartGetInteger(0,CHART_VISIBLE_BARS);
//--- 创建一个数组，写下每个柱的开盘时间
datetime Time[];
//--- 像时间序列一样访问数组
ArraySetAsSeries(Time,true);
//--- 现在复制图表中可见的柱数据到数组中
int times=CopyTime(NULL,0,0,bars_on_chart,Time);
if(times<=0)
{
    Print("Could not copy the array with the open time!");
    return;
}
//--- 完成预备

//--- 图表中心柱指数
int center_bar=bars_on_chart/2;
//--- 图表赤道 - 最大和最小值之间
double mean=(chart_max_price+chart_min_price)/2.0;
//--- 设置第一定位点坐标到中心
ObjectCreate(0,my_gann,OBJ_GANNFAN,0,Time[center_bar],mean,
    //--- 第二定位点到右边
    Time[center_bar/2],(mean+chart_min_price)/2.0);
Print("Time[center_bar] = "+(string)Time[center_bar]+"   Time[center_bar/2] = "+
//Print("Time[center_bar]/="+Time[center_bar]+"   Time[center_bar/2]="+Time[cent
//--- 设置 Pips / Bar 单元比例
ObjectSetDouble(0,my_gann,OBJPROP_SCALE,10);
//--- 设置趋向线
ObjectSetInteger(0,my_gann,OBJPROP_DIRECTION,GANN_UP_TREND);
//--- 设置线宽
ObjectSetInteger(0,my_gann,OBJPROP_WIDTH,1);
//--- 定义线型
ObjectSetInteger(0,my_gann,OBJPROP_STYLE,STYLE_DASHDOT);
//--- 设置线的颜色
ObjectSetInteger(0,my_gann,OBJPROP_COLOR,clrYellowGreen);
//--- 允许用户选择物件
ObjectSetInteger(0,my_gann,OBJPROP_SELECTABLE,true);
//--- 自动选择
ObjectSetInteger(0,my_gann,OBJPROP_SELECTED,true);
//--- 图表上绘画
ChartRedraw(0);
}
}

```

网页颜色

以下颜色参量是由颜色类型定义的：

clrBlack	clrDarkGreen	clrDarkSlateGray	clrOlive	clrGreen	clrTeal	clrNavy	clrPurple
clrMaroon	clrIndigo	clrMidnightBlue	clrDarkBlue	clrDarkOliveGreen	clrSaddleBrown	clrForestGreen	clrOliveDrab
clrSeaGreen	clrDarkGoldenrod	clrDarkSlateBlue	clrSienna	clrMediumBlue	clrBrown	clrDarkTurquoise	clrDimGray
clrLightSeaGreen	clrDarkViolet	clrFireBrick	clrMediumVioletRed	clrMediumSeaGreen	clrChocolate	clrCrimson	clrSteelBlue
clrGoldenrod	clrMediumSpringGreen	clrLawnGreen	clrCadetBlue	clrDarkOrchid	clrYellowGreen	clrLimeGreen	clrOrangeRed
clrDarkOrange	clrOrange	clrGold	clrYellow	clrChartreuse	clrLime	clrSpringGreen	clrAqua
clrDeepSkyBlue	clrBlue	clrMagenta	clrRed	clrGray	clrSlateGray	clrPeru	clrBlueViolet
clrLightSlateGray	clrDeepPink	clrMediumTurquoise	clrDodgerBlue	clrTurquoise	clrRoyalBlue	clrSlateBlue	clrDarkKhaki
clrIndianRed	clrMediumOrchid	clrGreenYellow	clrMediumAquamarine	clrDarkSeaGreen	clrTomato	clrRosyBrown	clrOrchid
clrMediumPurple	clrPaleVioletRed	clrCoral	clrCornflowerBlue	clrDarkGray	clrSandyBrown	clrMediumSlateBlue	clrTan
clrDarkSalmon	clrBurlyWood	clrHotPink	clrSalmon	clrViolet	clrLightCoral	clrSkyBlue	clrLightSalmon
clrPlum	clrKhaki	clrLightGreen	clrAquamarine	clrSilver	clrLightSkyBlue	clrLightSteelBlue	clrLightBlue
clrPaleGreen	clrThistle	clrPowderBlue	clrPaleGoldenrod	clrPaleTurquoise	clrLightGray	clrWheat	clrNavajoWhite
clrMoccasin	clrLightPink	clrGainsboro	clrPeachPuff	clrPink	clrBisque	clrLightGoldenrod	clrBlanchedAlmond
clrLemonChiffon	clrBeige	clrAntiqueWhite	clrPapayaWhip	clrCornsilk	clrLightYellow	clrLightCyan	clrLinen
clrLavender	clrMistyRose	clrOldLace	clrWhiteSmoke	clrSeashell	clrIvory	clrHoneydew	clrAliceBlue
clrLavenderBlush	clrMintCream	clrSnow	clrWhite				

通过使用 [ObjectSetInteger\(\)](#) 函数，颜色可以建立一个对象，使用 [PlotIndexSetInteger\(\)](#) 函数，建立颜色到常规指标中。为了得到颜色值，还可以使用简单函数 [ObjectGetInteger\(\)](#) 和 [PlotIndexGetInteger\(\)](#)。

示例：

```
//---- 指标设置
#property indicator_chart_window
#property indicator_buffers 3
```

```
#property indicator_plots 3
#property indicator_type1 DRAW_LINE
#property indicator_type2 DRAW_LINE
#property indicator_type3 DRAW_LINE
#property indicator_color1 clrBlue
#property indicator_color2 clrRed
#property indicator_color3 clrLime
```

Wingdings

该字符使用 `OBJ_ARROW` 对象：

32	33	34	35	36	37	38	39	40	41	42	43	44	45	46	47
48	49	50	51	52	53	54	55	56	57	58	59	60	61	62	63
64	65	66	67	68	69	70	71	72	73	74	75	76	77	78	79
80	81	82	83	84	85	86	87	88	89	90	91	92	93	94	95
96	97	98	99	100	101	102	103	104	105	106	107	108	109	110	111
112	113	114	115	116	117	118	119	120	121	122	123	124	125	126	127
128	129	130	131	132	133	134	135	136	137	138	139	140	141	142	143
144	145	146	147	148	149	150	151	152	153	154	155	156	157	158	159
160	161	162	163	164	165	166	167	168	169	170	171	172	173	174	175
176	177	178	179	180	181	182	183	184	185	186	187	188	189	190	191
192	193	194	195	196	197	198	199	200	201	202	203	204	205	206	207
208	209	210	211	212	213	214	215	216	217	218	219	220	221	222	223
224	225	226	227	228	229	230	231	232	233	234	235	236	237	238	239
240	241	242	243	244	245	246	247	248	249	250	251	252	253	254	255

通过使用 `ObjectSetInteger()` 函数，可以建立必要字符。

示例：

```
void OnStart()
{
    //---
    string up_arrow="up_arrow";
    datetime time=TimeCurrent();
    double lastClose[1];
    int close=CopyClose(Symbol(),Period(),0,1,lastClose);    // 获得收盘价
    //--- 如果获得价格
    if(close>0)
    {
        ObjectCreate(0,up_arrow,OBJ_ARROW,0,0,0,0,0);          // 创建一个箭头
        ObjectSetInteger(0,up_arrow,OBJPROP_ARROWCODE,241);    // 设置箭头代码
        ObjectSetInteger(0,up_arrow,OBJPROP_TIME,time);        // 设置时间
        ObjectSetDouble(0,up_arrow,OBJPROP_PRICE,lastClose[0]); // 预定价格
        ChartRedraw(0);                                         // 现在绘制箭头
    }
    else
        Print("Unable to get the latest Close price!");
}
```

指标常量

预定义37个 [技术指标](#)，可以使用在MQL5语言程序中，此外，它为使用 [iCustom\(\)](#) 函数创建自定义指标提供了机会，所有常量都被分成5组：

- [价格常量](#) – 选择价格和成交量类型，指标是可以评估的；
- [平滑方式](#) – 指标中使用的内置滤波方式；
- [指标线](#) – 当使用 [CopyBuffer\(\)](#) 访问指标值时的指标缓冲区标识符；
- [绘画样式](#) – 建立并书写18字节的线条设计；
- [自定义指标属性](#) 与 [自定义](#) 指标工作时使用在函数中；
- [指标类型](#) 用来当使用 [IndicatorCreate\(\)](#) 建立处理器时指明技术指标；
- [数据类型标识符](#) 用来当把 [MqlParam](#) 数组传到 [IndicatorCreate\(\)](#) 函数中时指明数据类型。

价格常数

技术指标的计算要求价格价值和成交量价值，该计算将执行，有7个预先的标识符在ENUM_APPLIED_PRICE项目中，以指定计算的期望价值基础。

ENUM_APPLIED_PRICE

ID	描述
PRICE_CLOSE	收盘价格
PRICE_OPEN	开盘价格
PRICE_HIGH	一个时期的最高价格
PRICE_LOW	一个时期的最低价格
PRICE_MEDIAN	中间值 (高+低) / 2
PRICE_TYPICAL	典型价格 (高+低+收盘价) / 3
PRICE_WEIGHTED	平均价格 (高+低+收盘价格+开盘价格) / 4

如果成交量用于计算，必要在 ENUM_APPLIED_VOLUME 中列举出一两个值

ENUM_APPLIED_VOLUME

ID	描述
VOLUME_TICK	赊欠成交量
VOLUME_REAL	交易成交量

[iStochastic\(\)](#) 技术指标以两种方式计算使用：

- 或者只以收盘价；
- 或者以最高或最低价；

为计算选择必须变量，指定ENUM_STO_PRICE 计算中的一种值

ENUM_STO_PRICE

ID	描述
STO_LOWHIGH	基于最低价/最高价的计算
STO_CLOSECLOSE	基于开盘价/收盘价的计算

如果技术指标使用价格数据的计算，建立ENUM_APPLIED_PRICE类型，然后任何指标处理（内置在终端里或者使用者输入）都能用来输入价格序列。在此情况下，指标中零缓冲器的值会用来计算，这样就很容易使用另一指标建造该指标的值，自定义指标的处理需要调用 [iCustom\(\)](#) 函数处理。

示例：

```
#property indicator_separate_window
#property indicator_buffers 2
#property indicator_plots 2
```

```

//--- 输入函数
input int      RSIPeriod=14;          // 计算 RSI 的周期
input int      Smooth=8;              // 平滑 RSI 周期
input ENUM_MA_METHOD meth=MODE_SMA;  // 修匀法
//---- 图 RSI
#property indicator_label1  "RSI"
#property indicator_type1   DRAW_LINE
#property indicator_color1  clrRed
#property indicator_style1  STYLE_SOLID
#property indicator_width1  1
//---- 图 RSI_Smoothed
#property indicator_label2  "RSI_Smoothed"
#property indicator_type2   DRAW_LINE
#property indicator_color2  clrNavy
#property indicator_style2  STYLE_SOLID
#property indicator_width2  1
//--- 指标缓冲区
double         RSIBuffer[];           // 这里存储 RSI 值
double         RSI_SmoothedBuffer[]; // RSI 平滑值
int            RSIhandle;              // 处理 RSI 指标
//+-----+
//| 自定义指标初始化函数 |
//+-----+
void OnInit()
{
//--- 指标缓冲区绘图 indicator buffers mapping
    SetIndexBuffer(0,RSIBuffer,INDICATOR_DATA);
    SetIndexBuffer(1,RSI_SmoothedBuffer,INDICATOR_DATA);
    IndicatorSetString(INDICATOR_SHORTNAME,"iRSI");
    IndicatorSetInteger(INDICATOR_DIGITS,2);
//---
    RSIhandle=iRSI(NULL,0,RSIPeriod,PRICE_CLOSE);
//---
}
//+-----+
//| 自定义指标重复函数 |
//+-----+
int OnCalculate(const int rates_total,
               const int prev_calculated,
               const int begin,
               const double &price[])
{
//--- 重置上一个错误值
    ResetLastError();
//--- 数组 RSI Buffer [] 中获得 RSI 指标数据
    int copied=CopyBuffer(RSIhandle,0,0,rates_total,RSIBuffer);

```

```
if(copied<=0)
{
    Print("Unable to copy the values of the indicator RSI. Error = ",
        GetLastError()," ", copied ="",copied);
    return(0);
}
//--- 创建使用RSI值的平均值的指标
int RSI_MA_handle=iMA(NULL,0,Smooth,0,method,RSIhandle);
copied=CopyBuffer(RSI_MA_handle,0,0,0,rates_total,RSI_SmoothedBuffer);
if(copied<=0)
{
    Print("Unable to copy the smoothed indicator of RSI. Error = ",
        GetLastError()," ", copied ="",copied);
    return(0);
}
//--- 为下次调用返回prev_calculated值
return(rates_total);
}
```


平滑方式

许多技术指标都是以各种价位序列平滑方式为基础的，许多标准技术指标要求输入平滑型的规格作为输入参数，对于平滑型描述类型的说明来说，可使用ENUM_ MA_ METHOD列表中的标识符。

ENUM_ MA_ METHOD

ID	描述
MODE_ SMA	简单平均数
MODE_ EMA	指数平均数
MODE_ SMMA	平滑平均数
MODE_ LWMA	线性平均数

示例：

```
double ExtJaws[];
double ExtTeeth[];
double ExtLips[];
//---- 移动平均数句柄
int ExtJawsHandle;
int ExtTeethHandle;
int ExtLipsHandle;
//--- 获得 MA's 句柄
ExtJawsHandle=iMA(NULL,0,JawsPeriod,0,MODE_SMMA,PRICE_MEDIAN);
ExtTeethHandle=iMA(NULL,0,TeethPeriod,0,MODE_SMMA,PRICE_MEDIAN);
ExtLipsHandle=iMA(NULL,0,LipsPeriod,0,MODE_SMMA,PRICE_MEDIAN);
```

指标线

在图表中许多[技术指标](#)会显示几个缓冲区，指标缓冲区起始数字是0，当使用[CopyBuffer\(\)](#)函数复制指标值到双精度类型数组时，对于一些指标来说，它可以代替数字表示复制标识符。

当复制 [iMACD\(\)](#)、[iRVI\(\)](#) 和 [iStochastic\(\)](#) 值时可用的指标线标识符

Constant	值	描述
MAIN_ LINE	0	干线
SIGNAL_ LINE	1	信号线

当复制 [ADX\(\)](#) 和 [ADXW\(\)](#) 值时可用的指标线标识符。

常量	值	描述
MAIN_ LINE	0	干线
PLUSDI_ LINE	1	线 +DI
MINUSDI_ LINE	2	线 -DI

当复制[iBands\(\)](#)值时可用的指标线标识符

常量	值	描述
BASE_ LINE	0	干线
UPPER_ BAND	1	最高限制
LOWER_ BAND	2	最低限制

当复制 [iEnvelopes\(\)](#) 和 [iFractals\(\)](#) 值时可用的指标线标识符

常量	值	描述
UPPER_ LINE	0	上边线
LOWER_ LINE	1	下边线

当复制[iGator\(\)](#)值时可用的指标线标识符。

常量	值	描述
UPPER_ HISTOGRAM	0	上直方图
LOWER_ HISTOGRAM	2	底部直方图

当复制 [iAlligator\(\)](#) 值时可用的指标线标识符。

常量	值	描述
GATORJAW_ LINE	0	爪线
GATORTEETH_ LINE	1	牙线
GATORLIPS_ LINE	2	唇线

当复制 `Ichimoku()` 值时可用的指标线标识符。

常量	值	描述
TENKANSEN_LINE	0	转换线
KIJUNSEN_LINE	1	基准线
SENKOUSPANA_LINE	2	闪光跨度A线
SENKOUSPANB_LINE	3	闪光跨度B线
CHINKOSPAN_LINE	4	延迟线

绘画风格

当创建一个自定义指标，可以指定18字节的图解测绘中任何一种（显示在基本图表窗口或者图表子窗口），其价格指定在ENUM_DRAW_TYPE 计算式中。

在一个自定义指标中，可以使用任意指标新建/绘画风格，为存储必要数据，每个建筑风格要求5种全局数组数组中的一个。通过使用SetIndexBuffer() 函数可使这些数据数组与指标缓冲区绑定在一起，ENUM_INDEXBUFFER_TYPE 数据类型是每个缓冲区指定的。

依据绘图风格，需要1-4个缓冲区（显示为INDICATOR_DATA）。如果该风格承认兼容的动态交替颜色（所有包含在颜色中的名称），就需要更多颜色的缓冲区（指示类型INDICATOR_COLOR_INDEX）。在价值缓冲区与风格相一致时，颜色缓冲区会受到限制。

ENUM_DRAW_TYPE

ID	描述	数据缓冲	颜色缓冲
DRAW_NONE	没有画线	1	0
DRAW_LINE	画线	1	0
DRAW_SECTION	线条	1	0
DRAW_HISTOGRAM	从0位线 画柱状图	1	0
DRAW_HISTOGRAM2	两个指标 命令的柱 状图	2	0
DRAW_ARROW	画箭头	1	0
DRAW_ZIGZAG	之字设计 允许垂直 界面	2	0
DRAW_FILLING	两层间的 颜色	2	0
DRAW_BARS	以字符序 列展示	4	0
DRAW_CANDLES	以蜡烛台 序列展示	4	0
DRAW_COLOR_LINE	多色线	1	1
DRAW_COLOR_SECTION	多色分段	1	1
DRAW_COLOR_HISTOGRAM	从0线开 始的多色 直方图	1	1
DRAW_COLOR_HISTOGRAM2	两个指标 缓冲区的 多色直方 图	2	1

DRAW_COLOR_ARROW	描绘多色箭头	1	1
DRAW_COLOR_ZIGZAG	多色之字形	2	1
DRAW_COLOR_BARS	多色字节	4	1
DRAW_COLOR_CANDLES	多色蜡烛台	4	1

提取所选描绘类型标识符，ENUM_PLOT_PROPERTY列表列出函数 [PlotIndexSetInteger\(\)](#) 和 [PlotIndexGetInteger\(\)](#)

ENUM_PLOT_PROPERTY_INTEGER

ID	描述	属性类型
PLOT_ARROW	DRAW_ARROW 类型箭头代码	无符字符型
PLOT_ARROW_SHIFT	DRAW_ARROW 类型垂直箭头转换	整型
PLOT_DRAW_BEGIN	在DataWindow中 没有描绘和值的原始字节数量	整型
PLOT_DRAW_TYPE	图解建筑类型	ENUM_DRAW_TYPE
PLOT_SHOW_DATA	在DataWindow中 显示建筑值标志	布尔型
PLOT_SHIFT	在字节中与时间轴 一起描绘的指标转换	整型
PLOT_LINE_STYLE	画线类型	ENUM_LINE_STYLE
PLOT_LINE_WIDTH	画线层次	整型
PLOT_COLOR_INDEXES	颜色数量	整型
PLOT_LINE_COLOR	包含绘画颜色的缓冲区指标	颜色 修饰符 = 颜色数量指标

函数 [PlotIndexSetDouble\(\)](#)

ENUM_PLOT_PROPERTY_DOUBLE

ID	描述	属性类型
PLOT_EMPTY_VALUE	测绘空值，没有绘图	双精度型

函数 [PlotIndexSetString\(\)](#)

ENUM_PLOT_PROPERTY_STRING

ID	描述	属性类型
PLOT_LABEL	显示在 DataWindow中的 指标图解系列的名称	字符串型

在自定义指标中，有5种类型可以使用绘画线条，他们只在或者厚度时是有效的

ENUM_LINE_STYLE

ID	描述
STYLE_SOLID	实线
STYLE_DASH	折线
STYLE_DOT	虚线
STYLE_DASHDOT	折点线
STYLE_DASHDOTDOT	双折点线

为了建立线条画类型和描绘类型，可以使用 [PlotIndexSetInteger\(\)](#) 函数，使用函数 [ObjectSetInteger\(\)](#) 可以标明斐波纳契引申厚度和描绘水平类型。

示例：

```
#property indicator_chart_window
#property indicator_buffers 1
#property indicator_plots 1
//--- 指标缓冲区
double      MABuffer[];
//+-----+
//| 自定义指标初始化函数          |
//+-----+
void OnInit()
{
//--- 绑定数组和指数为0的指标缓冲区
    SetIndexBuffer(0,MABuffer,INDICATOR_DATA);
//--- 设置画线
    PlotIndexSetInteger(0,PLOT_DRAW_TYPE,DRAW_LINE);
//--- 设置线型
    PlotIndexSetInteger(0,PLOT_LINE_STYLE,STYLE_DOT);
//--- 设置线颜色
    PlotIndexSetInteger(0,PLOT_LINE_COLOR,clrRed);
//--- 设置线粗细
    PlotIndexSetInteger(0,PLOT_LINE_WIDTH,1);
//--- 为线设置标签
    PlotIndexSetString(0,PLOT_LABEL,"Moving Average");
```

```
//---
}
//+-----+
//| 自定义指标重复函数 |
//+-----+
int OnCalculate(const int rates_total,
               const int prev_calculated,
               const datetime &time[],
               const double &open[],
               const double &high[],
               const double &low[],
               const double &close[],
               const long &tick_volume[],
               const long &volume[],
               const int &spread[])
{
//---
    for(int i=prev_calculated;i<rates_total;i++)
    {
        MABuffer[i]=close[i];
    }
//--- 为下次调用返回prev_calculated值
    return(rates_total);
}
```

Custom Indicators Properties

用在自定义指标中的指标缓冲区的数量是受限的，使用[SetIndexBuffer\(\)](#)函数可以设计成自定义缓冲区，因为要存储，所有指定数据类型是有必要的，也可能是ENUM_INDEXBUFFER_TYPE其中一个值。

ENUM_INDEXBUFFER_TYPE

ID	描述
INDICATOR_DATA	描绘数据
INDICATOR_COLOR_INDEX	颜色
INDICATOR_CALCULATIONS	媒介运算的辅助缓冲区

自定义指标提供许多设置提供方便演绎，这些设置通过使用[IndicatorSetDouble\(\)](#),[IndicatorSetInteger\(\)](#) 和 [IndicatorSetString\(\)](#) 函数来分别相应指标性质，指标属性的标识符列表在ENUM_CUSTOMIND_PROPERTY中。

ENUM_CUSTOMIND_PROPERTY_INTEGER

ID	描述	属性类型
INDICATOR_DIGITS	描绘指标值的精确度	整型
INDICATOR_HEIGHT		int

	(#property indicator _height)	
INDICATOR_LEVELS	指标窗口中的水平数量	整型
INDICATOR_LEVELCOLOR	水平线颜色	颜色 修饰符 = 层号
INDICATOR_LEVELSTYLE	水平线类型	<u>ENUM_LINE_STYLE</u> 修饰符 = 层号
INDICATOR_LEVELWIDTH	水平线厚度	整型 修饰符 = 层号

ENUM_CUSTOMIND_PROPERTY_DOUBLE

ID	描述	属性类型
INDICATOR__MINIMUM	指标窗口最小化	双精度
INDICATOR__MAXIMUM	指标窗口最大化	双精度
INDICATOR__LEVELVALUE	水平值	双精度 修饰符 = 层号

ENUM_CUSTOMIND_PROPERTY_STRING

ID	描述	属性类型
INDICATOR_SHORTNAME	短指标名称	字符串
INDICATOR_LEVELTEXT	水平描述	字符串 修饰符=层号

示例：

```
//--- 指标设置
#property indicator_separate_window
#property indicator_buffers 4
#property indicator_plots 2
#property indicator_type1 DRAW_LINE
#property indicator_type2 DRAW_LINE
#property indicator_color1 clrLightSeaGreen
#property indicator_color2 clrRed
//--- 输入参数
extern int KPeriod=5;
extern int DPeriod=3;
extern int Slowing=3;
//--- 指标缓冲区
double MainBuffer[];
double SignalBuffer[];
double HighesBuffer[];
double LowesBuffer[];
//+-----+
//| 自定义指标初始化函数 |
//+-----+
void OnInit()
{
//--- 指标缓冲区绘图
SetIndexBuffer(0,MainBuffer,INDICATOR_DATA);
SetIndexBuffer(1,SignalBuffer,INDICATOR_DATA);
SetIndexBuffer(2,HighesBuffer,INDICATOR_CALCULATIONS);
SetIndexBuffer(3,LowesBuffer,INDICATOR_CALCULATIONS);
//--- 设置精确性
IndicatorSetInteger(INDICATOR_DIGITS,2);
//--- 设置水平
IndicatorSetInteger(INDICATOR_LEVELS,2);
IndicatorSetDouble(INDICATOR_LEVELVALUE,0,20);
IndicatorSetDouble(INDICATOR_LEVELVALUE,1,80);
//--- 为子窗口设置最大最小值
IndicatorSetDouble(INDICATOR_MINIMUM,0);
IndicatorSetDouble(INDICATOR_MAXIMUM,100);
```

```
//--- 设置第一条画指数的柱
PlotIndexSetInteger(0,PLOT_DRAW_BEGIN,KPeriod+Slowing-2);
PlotIndexSetInteger(1,PLOT_DRAW_BEGIN,KPeriod+Slowing+DPeriod);
//--- 为第二行设置STYLE_DOT类型
PlotIndexSetInteger(1,PLOT_LINE_STYLE,STYLE_DOT);
//--- 为DataWindow和指标子窗口标签命名
IndicatorSetString(INDICATOR_SHORTNAME,"Stoch("+KPeriod+", "+DPeriod+", "+Slowing+")
PlotIndexSetString(0,PLOT_LABEL,"Main");
PlotIndexSetString(1,PLOT_LABEL,"Signal");
//--- 设置画线空值
PlotIndexSetDouble(0,PLOT_EMPTY_VALUE,0.0);
PlotIndexSetDouble(1,PLOT_EMPTY_VALUE,0.0);
//--- 完成初始化
}
```

技术指标类型

为使用 [接入值](#)，有两种方法建立指标处理器，第一种方法是从[技术指标](#) 列表中直接指定函数名称，第二种方法是使用[IndicatorCreate\(\)](#)，通过分配ENUM_INDICATOR中的标识符共同增添任意指标处理。两种处理办法的创建结果是相同的，当使用MQL5编辑程序时，在特定情况下使用最简便的。

当创建指标IND_CUSTOM类型时，[输入参数 MqlParam](#) 数组中第一个元素的类型领域一定有 [ENUM_DATATYPE](#)，表达式中的 TYPE_STRING 值，而第一个元素中的string_value一定包括自定义指标中的名称。

ENUM_INDICATOR

标识符	指标
IND_AC	加速振荡器
IND_AD	聚集/分散
IND_ADX	均定向指标
IND_ADXW	亚当理论的ADX
IND_ALLIGATOR	鳄鱼指标
IND_AMA	相应的移动平均数
IND_AO	动量振荡指标-AO指标
IND_ATR	真实波动幅度均值
IND_BANDS	布林线指标（AO指标）
IND_BEARS	熊市
IND_BULLS	牛市
IND_BWMFI	市场便利指标
IND_CCI	顺势指标
IND_CHAIKIN	佳庆指标
IND_CUSTOM	自定义指标
IND_DEMA	双精度移动平均线
IND_DEMARKER	DEM指标
IND_ENVELOPES	轨道线指标
IND_FORCE	强力指数
IND_FRACTALS	拼图
IND_FRAMA	自适应拼图移动平均数
IND_GATOR	振荡器
IND_ICHIMOKU	一目均衡图

IND_ MA	平均移动
IND_ MACD	MACD
IND_ MFI	货币流量索引
IND_ MOMENTUM	动量
IND_ OBV	平衡成交量
IND_ OSMA	OsMA
IND_ RSI	相对强势索引
IND_ RVI	相对活力索引
IND_ SAR	抛物线SAR
IND_ STDDEV	标准偏差
IND_ STOCHASTIC	随机振荡器
IND_ TEMA	三倍指数移动平均值
IND_ TRIX	三倍指数移动平均值振荡器
IND_ VIDYA	索引变量平均值
IND_ VOLUMES	成交量
IND_ WPR	威廉姆斯百分比幅度

数据类型标识符

当使用IndicatorCreate() 函数创建自定义处理器时，MqlParam 类型数组会以最后一个参量规定，因此，MqlParam结构，描述的指标，包括一种特殊的字段类型。该领域包括的关于数据类型（实数、整数 或者 字符串类型）的信息是通过数组中的特殊元素传递的，MqlParam结构中的值或许是ENUM_DATATYPE值中的一个。

ENUM_DATATYPE

标识符	数据类型
TYPE_BOOL	布尔
TYPE_CHAR	字符型
TYPE_UCHAR	无符字符型
TYPE_SHORT	短整型
TYPE_USHORT	无符号短整型
TYPE_COLOR	颜色
TYPE_INT	整型
TYPE_UINT	无符号整型
TYPE_DATETIME	日期时间
TYPE_LONG	长整型
TYPE_ULONG	无符号长整型
TYPE_FLOAT	浮点型
TYPE_DOUBLE	双精度型
TYPE_STRING	字符串

数组中的每一个元素都能描述创建的技术指标录入函数，因此数组中元素的类型和命令一定与描述保持严格一致。

环境状态

MQL5程序中描述当前运行环境的常量分为如下组：

- [客户端属性](#) – 客户端信息；
- [运行MQL5程序属性](#) – MQL5程序特征，它帮助控制执行命令；
- [交易品种属性](#) – 有关交易品种的信息；
- [账户属性](#) – 有关现金账户的信息；

• _____ -

客户端属性

有关客户端信息包含在两种函数中，[TerminalInfoInteger\(\)](#) 和 [TerminalInfoString\(\)](#)。对于参量来说，这些函数分别从ENUM_TERMINAL_INFO_INTEGER 和 ENUM_TERMINAL_INFO_STRING中接收值。

ENUM_TERMINAL_INFO_INTEGER

标识符	描述	类型
TERMINAL_BUILD	客户端构造编号	整型
TERMINAL_CONNECTED	连接的交易服务器	布尔型
TERMINAL_DLLS_ALLOWED	使用DLL许可	布尔型
TERMINAL_TRADE_ALLOWED	允许交易	布尔型
TERMINAL_EMAIL_ENABLED	在制定终端允许使用SMTP-server 和 login发送邮件	布尔型
TERMINAL_FTP_ENABLED	在制定终端允许使用FTP-server 和 login发送报告	布尔型
TERMINAL_MAXBARS	图表中的最大字节	整型
TERMINAL_LANGUAGE	在客户端建立的 语言代码页 数字	整型

只能在两种目录下执行[文件操作](#)；使用TERMINAL_DATA_PATH 和 TERMINAL_COMMONDATA_PATH 要求特征包括相似路径。

ENUM_TERMINAL_INFO_STRING

标识符	描述	类型
TERMINAL_COMPANY	公司名称	字符串
TERMINAL_NAME	程序端名称	字符串
TERMINAL_PATH	程序端文件夹启动	字符串
TERMINAL_DATA_PATH	程序端数据文件夹存储	字符串
TERMINAL_COMMONDATA_PATH	电脑中所有程序端的普通路径	字符串

为更好的了解路径，存储在TERMINAL_PATH, TERMINAL_DATA_PATH 和 TERMINAL_COMMONDATA_PATH参量中的属性，推荐使用脚本，它可以通过当前复制的客户端的来返回这些值。

示例：脚本返回的有关客户端路径的信息

```
//+-----+
//|                                     Check_TerminalPaths.mq5 |
//|                                     Copyright 2009, MetaQuotes Software Corp. |
//|                                     http://www.mql5.com |
//+-----+
#property copyright "2009, MetaQuotes Software Corp."
```



```
#property link      "http://www.mql5.com"
#property version   "1.00"

//+-----+
//| 脚本程序启动函数 |
//+-----+

void OnStart()
{
//---
    Print("TERMINAL_PATH = ",TerminalInfoString(TERMINAL_PATH));
    Print("TERMINAL_DATA_PATH = ",TerminalInfoString(TERMINAL_DATA_PATH));
    Print("TERMINAL_COMMONDATA_PATH = ",TerminalInfoString(TERMINAL_COMMONDATA_PATH));
}
```

脚本执行的结果在Experts Journal中可以看到相关信息，如下表示：

Message
TERMINAL_COMMONDATA_PATH = C:\Documents and Settings\All Users\Application Data\MetaQuotes\Terminal...
TERMINAL_DATA_PATH = C:\Program Files\MetaTrader 5
TERMINAL_PATH = C:\Program Files\MetaTrader 5

运行MQL程序属性

为了获取当前运行的MQL5程序的信息，可以使用ENUM_MQL5_INFO_INTEGER 和 ENUM_MQL5_INFO_STRING常量。

函数 [MQL5InfoInteger](#)

ENUM_MQL5_INFO_INTEGER

标识符	描述	类型
MQL5_PROGRAM_TYPE	MQL5程序类型	ENUM_PROGRAM_TYPE
MQL5_DLLS_ALLOWED	允许为已生效的程序使用DLL	布尔型
MQL5_TRADE_ALLOWED	允许为已生效的程序交易	布尔型
MQL5_DEBUGGING	标志表示调试方式	布尔型
MQL5_TESTING	标志表示测试过程	布尔型
MQL5_OPTIMIZATION	标志表示最佳化过程	布尔型
MQL5_VISUAL_MODE	标志表示视觉测试器过程	布尔型

函数 [MQL5InfoString](#)

ENUM_MQL5_INFO_STRING

标识符	描述	类型
MQL5_PROGRAM_NAME	mql5已执行程序名称	字符串
MQL5_PROGRAM_PATH	以执行系统路径	字符串

运行程序类型信息，使用ENUM_PROGRAM_TYPE 值

ENUM_PROGRAM_TYPE

标识符	描述
PROGRAM_SCRIPT	脚本
PROGRAM_EXPERT	专家
PROGRAM_INDICATOR	指标

示例：

```
ENUM_PROGRAM_TYPE mql_program=(ENUM_PROGRAM_TYPE)MQL5InfoInteger(MQL5_PROGRAM_TYPE
switch(mql_program)
{
```

```
case PROGRAM_SCRIPT:
{
    Print(__FILE__+" is script");
    break;
}
case PROGRAM_EXPERT:
{
    Print(__FILE__+" is Expert Advisor");
    break;
}
case PROGRAM_INDICATOR:
{
    Print(__FILE__+" is custom indicator");
    break;
}
default:Print("MQL5 type value is ",mql_program);
}
```

交易品种属性

可以通过 [SymbolInfoInteger\(\)](#)，[SymbolInfoDouble\(\)](#) 和 [SymbolInfoString\(\)](#)。函数获得当前市场信息。第一个参量是交易品种名称，第二个函数参量的值是ENUM_SYMBOL_INFO_INTEGER, ENUM_SYMBOL_INFO_DOUBLE 和 ENUM_SYMBOL_INFO_STRING标识符中的一个。

函数 [SymbolInfoInteger\(\)](#)

ENUM_SYMBOL_INFO_INTEGER

标识符	描述	类型
SYMBOL_SELECT	在市场价格中选择交易品种	布尔型
SYMBOL_VOLUME	最后订单成交量	长整型
SYMBOL_VOLUMEHIGH	当天最大订单	长整型
SYMBOL_VOLUMELOW	当天最小订单	长整型
SYMBOL_VOLUMEBID	当前买入成交量	长整型
SYMBOL_VOLUMEASK	当前卖出成交量	长整型
SYMBOL_TIME	最后报价时间	日期时间
SYMBOL_DIGITS	小数点后数字	整型
SYMBOL_SPREAD_FLOAT	浮点传播迹象	布尔型

SYMBOL_SPREAD	相关传播值	整型
SYMBOL_TICKS_BOOKDEPTH	显示在 Depth of Market 要求中的最大数量，交易品种无队列要求，值是0	整型
SYMBOL_TRADE_CALC_MODE	合约价格计算方式	ENUM_SYMBOL_CALC_MODE
SYMBOL_TRADE_MODE	订单执行类型	ENUM_SYMBOL_TRADE_MODE
SYMBOL_START_TIME	交易品种交易开始日期（通常用于期货）	日期时间
SYMBOL_EXPIRATION_TIME	交易品种交易结束日期（通常用于期货）	日期时间
SYMBOL_TRADE_STOPS_LEVEL	止蚀盘当前收	整型

	盘价格的 最小 空间	
SYMBOL_TRADE_FREEZE_LEVEL	凝结 交易 操作 的距 离	整型
SYMBOL_TRADE_EXEMODE	订单 执行 方式	ENUM_SYMBOL_TRADE_EXECUTION
SYMBOL_SWAP_MODE	交易 计算 模式	ENUM_SYMBOL_SWAP_MODE
SYMBOL_SWAP_ROLLOVER3DAYS	日翻 滚收 费	ENUM_DAY_OF_WEEK
SYMBOL_EXPIRATION_MODE	到期 模式 允许 命令 标志	整型
SYMBOL_FILLING_MODE	填充 模式 允许 命令 标志	整型

函数 [SymbolInfoDouble\(\)](#)

ENUM_SYMBOL_INFO_DOUBLE

标识符	描述	类型
SYMBOL_BID	买价—最佳卖出信息	双精度
SYMBOL_BIDHIGH	一天中最高买价	双精度
SYMBOL_BIDLOW	一天中最低买价	双精度
SYMBOL_ASK	卖价—最佳买入信息	双精度
SYMBOL_ASKHIGH	一天中最高买价	双精度
SYMBOL_ASKLOW	一天中最低买价	双精度
SYMBOL_LAST	最后订单价格	双精度
SYMBOL_LASTHIGH	一天中最高	双精度
SYMBOL_LASTLOW	一天中最低	双精度

SYMBOL_POINT	交易品种点值	双精度
SYMBOL_TRADE_TICK_VALUE	Value of SYMBOL_TRADE_TICK_VALUE_PROFIT	双精度
SYMBOL_TRADE_TICK_VALUE_PROFIT	为利润空间计算最小价位	双精度
SYMBOL_TRADE_TICK_VALUE_LOSS	为亏损空间计算最小价位	双精度
SYMBOL_TRADE_TICK_SIZE	最小价格改变	双精度
SYMBOL_TRADE_CONTRACT_SIZE	交易贸易合同	双精度
SYMBOL_VOLUME_MIN	一笔订单中的最小成交量	双精度
SYMBOL_VOLUME_MAX	一笔订单中的最大成交量	双精度
SYMBOL_VOLUME_STEP	交易执行缓步的最小成交量	双精度
SYMBOL_VOLUME_LIMIT	每个交易品种的平仓（忽视方位）和代办订单允许的最大总成交量	双精度
SYMBOL_SWAP_LONG	最低交易值	双精度
SYMBOL_SWAP_SHORT	最小交易值	双精度
SYMBOL_MARGIN_INITIAL	原始保证金表示每一笔保证金开仓成交量的数量	双精度
SYMBOL_MARGIN_MAINTENANCE	维持保证金，在交易品种中建立保证金数量，从一笔订单中获得保证金利润，当客户账户改变时，使用客户资产检测系统，如果维持保证金等于，使用原始保证金	双精度
SYMBOL_MARGIN_LONG	看涨行情中保证金利润率	双精度
SYMBOL_MARGIN_SHORT	看跌行情中保证金利润率	双精度
SYMBOL_MARGIN_LIMIT	限价订单中保证金利润率	双精度
SYMBOL_MARGIN_STOP	停止订单中保证金利润率	双精度
SYMBOL_MARGIN STOPLIMIT	限制停止订单中保证金利润率	双精度

函数 [SymbolInfoString\(\)](#)

ENUM_SYMBOL_INFO_STRING

Identifier	描述	类型
SYMBOL_CURRENCY_BASE	交易	字符串

	品种 基础 货币	
SYMBOL_ CURRENCY_ PROFIT	利润 货币	字符串
SYMBOL_ CURRENCY_ MARGIN	保证 金货 币	字符串
SYMBOL_ BANK	当前 报价 支线	字符串
SYMBOL_ DESCRIPTION	交易 品种 描述	字符串
SYMBOL_ PATH	交易 品种 树形 通路	字符串

待办订单中每个交易品种的几处终结点是能够制定的，一个标志匹配一种方式，使用 OR (|) 逻辑操作符能够连接标志，例如，SYMBOL_ EXPIRATION_ GTC|SYMBOL_ EXPIRATION_ SRECIFIED，为了检测交易品种的某一方式是否被允许操作，结果可以使用AND (&) 逻辑操作符来与方式标志相对比。

如果交易品种的SYMBOL_ EXPIRATION_ SRECIFIED标志可以指定，发送待办订单，可以为其指定某一时刻交易。

标识符	值	描述
SYMBOL_ EXPIRATION_ GTC	1	该命令在无限时间周期中是有效的，直到它被明确删除
SYMBOL_ EXPIRATION_ DAY	2	当天结束前，该命令是有效的
SYMBOL_ EXPIRATION_ SRECIFIED	4	在命令中指定期限时间

示例：

```
//+-----+
//| 检测是否允许指定终结模式          |
//+-----+
bool IsExpirationTypeAllowed(string symbol,int exp_type)
{
//--- 获得描述允许终结模式的属性值
    int expiration=(int)SymbolInfoInteger(symbol,SYMBOL_EXPIRATION_MODE);
//--- 返回真值，如果exp_type模式允许的话
    return ((expiration&exp_type)==exp_type);
}
```


当发出命令时，可以为建立的命令指定成交量的交易政策。在图表中每个交易品种都是可以指定的，你可以通过连接标志为一个交易品种建立多种方式。这些标志通过逻辑操作符OR(|) 连接，例如，
 SYMBOL_FILLING_ALL_OR_NONE|SYMBOL_CANCEL_REMAIND，如果坚持交易品种的确切模式是否允许，结果可以使用AND(&) 逻辑操作符与方式标志进行比较。

标识符	值	描述
SYMBOL_FILLING_ALL_OR_NONE	1	“全部或者全无”。如果订单中的成交量是指定的，规定价格不能全部填满，命令就会被取消并无法管理订单
SYMBOL_CANCEL_REMAIND	2	如果只有一部分指定成交量被填满，，可行成交量订单是可以执行的，命令取消余数，新订单无处安置
SYMBOL_RETURN_REMAIND	4	订单已经被可行成交量执行，会为剩余部分建立相同价格的新订单

示例：

```
//+-----+
//|   检测是否允许指定终结模式   |
//+-----+
bool IsFillingTypeAllowed(string symbol,int fill_type)
{
//--- 获得描述允许终结模式的属性值
    int filling=(int)SymbolInfoInteger(symbol,SYMBOL_FILLING_MODE);
//--- 返回真值，如果fill_type模式允许的话
    return((filling & fill_type)==fill_type);
}
```

为计算交易品种的保证金规定额，使用The ENUM_SYMBOL_CALC_MODE项目获取信息。

ENUM_SYMBOL_CALC_MODE

标识符	描述	公式
SYMBOL_CALC_MODE_FOREX	外汇方式 —计算利润和保证金	保证金：Lots*Contract_Size/Leverage 利润：(close_price-open_price) *Contract_Size*Lots
SYMBOL_CALC_MODE_FUTURES	期货方式 —计算保证金和利	保证金：Lots *InitialMargin*Percentage/100 利润：(close_price-open_price) *TickPrice/ TickSize*Lots

	润	
SYMBOL_CALC_MODE_CFD	CFD 方式 —计 算保 证金 和利 润	保证金：Lots *ContractSize*MarketPrice*Percentage/100 利润：(close_ price-open_ price) *Contract_ Size*Lots
SYMBOL_CALC_MODE_CFDINDEX	CFD 指数 模式 —通 过指 数计 算保 证金 和利 润	保证金：(Lots*ContractSize*MarketPrice) *TickPrice/TickSize 利润：(close_ price-open_ price) *Contract_ Size*Lots
SYMBOL_CALC_MODE_CFDLEVERAGE	CFD 杠杆 模式 —通 过杠 杆贸 易计 算保 证金 和利 润	保证金： (Lots*ContractSize*MarketPrice*Percentage) / Leverage 利润：(close_ price-open_ price) *Contract_ Size*Lots

在交易品种中有许多种交易方式，某一交易品种的交易方式信息反映在ENUM_SYMBOL_TRADE_MODE项目值中。

ENUM_SYMBOL_TRADE_MODE

标识符	描述
SYMBOL_TRADE_MODE_DISABLED	交易品种交易失去功效
SYMBOL_TRADE_MODE_LONGONLY	长仓允许
SYMBOL_TRADE_MODE_SHORTONLY	短仓允许
SYMBOL_TRADE_MODE_CLOSEONLY	平仓操作允许
SYMBOL_TRADE_MODE_FULL	没有交易限制

定义在项目ENUM_SYMBOL_TRADE_EXECUTION中的每一个交易品种的可能性订单执行方式

ENUM_SYMBOL_TRADE_EXECUTION

标识符	描述
SYMBOL_TRADE_EXECUTION_REQUEST	应邀执行
SYMBOL_TRADE_EXECUTION_INSTANT	立即执行
SYMBOL_TRADE_EXECUTION_MARKET	交易执行

在枚举ENUM_SYMBOL_SWAP_MODE中指定的持仓位转移交换计算方式。

ENUM_SYMBOL_SWAP_MODE

标识符	描述
SYMBOL_SWAP_MODE_DISABLED	禁止交换模式（不交换）
SYMBOL_SWAP_MODE_BY_POINTS	恰当点交换
SYMBOL_SWAP_MODE_BY_MONEY	货币交换，入金货币交易品种属性指定值。
SYMBOL_SWAP_MODE_BY_INTEREST	年交换百分比（银行模式-一年360天）
SYMBOL_SWAP_MODE_BY_MARGIN_CURRENCY	货币交换，保证金货币交易品种属性指定值。

为说明工作日使用ENUM_DAY_OF_WEEK项目值

ENUM_DAY_OF_WEEK

标识符	描述
SUNDAY	星期日
MONDAY	星期一
TUESDAY	星期二
WEDNESDAY	星期三
THURSDAY	星期四
FRIDAY	星期五
SATURDAY	星期六

账户属性

使用 [AccountInfoInteger\(\)](#)、[AccountInfoDouble\(\)](#) 和 [AccountInfoString\(\)](#) 函数获得活期账户信息，该函数参数值可用接收类似于ENUM_ACCOUNT_INFO项目的值。

使用 [AccountInfoInteger\(\)](#) 函数

ENUM_ACCOUNT_INFO_INTEGER

标识符	描述	类型
ACCOUNT_LOGIN	账户数量	长整型
ACCOUNT_TRADE_MODE	账户交易方式	ENUM_ACCOUNT_TRADE_MODE
ACCOUNT_LEVERAGE	账户杠杆	长整型
ACCOUNT_LIMIT_ORDERS	活跃待办订单最大允许量	整型
ACCOUNT_MARGIN_SO_MODE	建立最少使用保证金方式	ENUM_ACCOUNT_MARGIN_SO_MODE
ACCOUNT_TRADE_ALLOWED	允许活期账户交易	布尔型
ACCOUNT_TRADE_EXPERT	允许EA交易	布尔型

函数 [AccountInfoDouble\(\)](#)

ENUM_ACCOUNT_INFO_DOUBLE

标识符	描述	类型
ACCOUNT_BALANCE	存入货币时账户结余	双精度
ACCOUNT_CREDIT	存入货币时账户亏空	双精度
ACCOUNT_PROFIT	存入货币时账户当前利润	双精度
ACCOUNT_EQUITY	存入货币时账户权益	双精度
ACCOUNT_MARGIN	存入货币时账户保证金使用	双精度
ACCOUNT_FREEMARGIN	存入货币时账户可用保证金	双精度
ACCOUNT_MARGIN_LEVEL	账户保证金水平百分比形式制定	双精度
ACCOUNT_MARGIN_SO_CALL	保证金调用水平，依据建立的ACCOUNT_MARGIN_SO_MODE，以百分比形式或存入货币时期表	双精度

	示	
ACCOUNT_ MARGIN_ SO_ SO	保证金停用水平，依据建立的 ACCOUNT_ MARGIN_ SO_ MODE，以百分比形式或存入货币时期表示	双精度

函数 [AccountInfoString\(\)](#)

ENUM_ ACCOUNT_ INFO_ STRING

标识符	描述	类型
ACCOUNT_ NAME	用户名	字符串
ACCOUNT_ SERVER	交易服务器名称	字符串
ACCOUNT_ CURRENCY	账户货币	字符串
ACCOUNT_ COMPANY	提供账户的公司名称	字符串

ENUM_ ACCOUNT_ TRADE_ MODE

标识符	描述
ACCOUNT_ TRADE_ MODE_ DEMO	样本账户
ACCOUNT_ TRADE_ MODE_ CONTEST	竞争账户
ACCOUNT_ TRADE_ MODE_ REAL	真实账户

ENUM_ ACCOUNT_ STOPOUT_ MODE

标识符	描述
ACCOUNT_ STOPOUT_ MODE_ PERCENT	账户以半百分比方式截止
ACCOUNT_ STOPOUT_ MODE_ MONEY	账户以货币方式截止

TesterStatistics(),
ENUM_STATISTICS.
- int double -
double.
double,

ENUM_STATISTICS

STAT_INITIAL_DEPOSIT		double
STAT_WITHDRAWAL		double
STAT_PROFIT	STAT_GROSS_PROFIT STAT_GROSS_LOSS (STAT_GROSS_LOSS)	double
STAT_GROSS_PROFIT	()	double

STAT_GROSS_LOSS	<pre> (,) </pre>	double
STAT_MAX_PROFITTRADE	<pre> - </pre>	double
STAT_MAX_LOSSTRADE	<pre> - </pre>	double
STAT_CONPROFITMAX	<pre> . </pre>	double
STAT_CONPROFITMAX_TRADES	<pre> , STAT_CONPROFITMAX AX (</pre>	int

)	
STAT__MAX__CONWINS		double
STAT__MAX__CONPROFIT__TRADES	STAT__MAX__CONWINS	int
STAT__CONLOSSMAX	.	double
STAT__CONLOSSMAX__TRADES	STAT__CONLOSSMAX (int
STAT__MAX__CONLOSSES)	double

STAT__MAX__CONLOSS__TRADES	STAT__MAX__CONLOSSES	int
STAT__BALANCEMIN		double
STAT__BALANCE__DD		double
STAT__BALANCEDD__PERCENT	(STAT__BALANCE__DD) .	double
STAT__BALANCE__DDREL__PERCENT		double

STAT__BALANCE__DD__RELATIVE	(STAT__BALANCE__D DREL__PERCENT)	double
STAT__EQUITYMIN		double
STAT__EQUITY__DD		double
STAT__EQUITYDD__PERCENT		double

	(STAT_EQUITY_DD) .	
STAT_EQUITY_DDREL_PERCENT	.	double
STAT_EQUITY_DD_RELATIVE	, (STAT_EQUITY_DDREL_PERCENT)) .	double
STAT_EXPECTED_PAYOFF		double
STAT_PROFIT_FACTOR		double

	<p>-</p> <p>STAT_GROSS_PROFIT/ STAT_GROSS_LOSS. STAT_GROSS_LOSS=0,</p> <p>DBL_MAX</p>	
STAT_RECOVERY_FACTOR	<p>-</p> <p>STAT_PROFIT/ STAT_BALANCE_DD</p>	double
STAT_SHARPE_RATIO		double
STAT_MAX_MARGINLEVEL		double
STAT_CUSTOM_ONTESTER	<p>,</p> <p>OnTester()</p>	double
STAT_DEALS		int
STAT_TRADES		int
STAT_PROFIT_TRADES		int
STAT_LOSS_TRADES		int

STAT_SHORT_TRADES		int
STAT_LONG_TRADES		int
STAT_PROFIT_SHORTTRADES		int
STAT_PROFIT_LONGTRADES		int
STAT_PROFITTRADES_AVGCON		int
STAT_LOSSTRADES_AVGCON		int

交易常数

用于程序买卖策略的各种常数分为如下组：

- [历史数据库属性](#) – 接收一般信息；
- [订单属性](#) – 获得交易订单信息；
- [仓位属性](#) – 获得当前位置信息；
- [交易属性](#) – 获得交易信息；
- [交易操作类型](#) – 交易可行性描述；
- [DOM交易订单](#) – 根据要求操作切割订单。

历史数据库属性

当访问函数 [timeseries](#) 时，[SeriesInfoInteger\(\)](#) 是用来获得额外 [交易品种信息](#)。必要属性标识符以函数参量传递，标识符可以是values of ENUM_ SERIES_ INFO_ INTEGER值中的一个。

ENUM_ SERIES_ INFO_ INTEGER

标识符	描述	类型
SERIES_ BARS_ COUNT	当前时刻交易品种周期字节价值	长整型
SERIES_ FIRSTDATE	当前时刻交易品种周期第一数据	日期时间
SERIES_ LASTBAR_ DATE	交易品种周期最后字节的开仓时间	日期时间
SERIES_ SERVER_ FIRSTDATE	忽略时间表的服务器上交易品种历史中第一个日期	日期时间
SERIES_ TERMINAL_ FIRSTDATE	忽略时间表的客户端上交易品种历史中第一个日期	日期时间
SERIES_ SYNCRONIZED	当前时刻交易品种/周期的同步数据	布尔型

订单属性

请求执行交易操作与命令一样正式化，每个订单都有多种属性准备，他们的信息可以使用[OrderGet...\(\)](#) 和 [HistoryOrderGet...\(\)](#) 函数获得。

函数 [OrderGetInteger\(\)](#) 和 [HistoryOrderGetInteger\(\)](#)

ENUM_ORDER_PROPERTY_INTEGER

标识符	描述	类型
ORDER_TIME_SETUP	订单设置时间	日期时间
ORDER_TYPE	订单类型	ENUM_ORDER_TYPE
ORDER_STATE	订单陈述	ENUM_ORDER_STATE
ORDER_TIME_EXPIRATION	订单终结时间	日期时间
ORDER_TIME_DONE	订单执行或取消时间	日期时间
ORDER_TYPE_FILLING	订单填满类型	ENUM_ORDER_TYPE_FILLING
ORDER_TYPE_TIME	订单使用期	ENUM_ORDER_TYPE_TIME
ORDER_MAGIC	EA交易的ID安置在订单之中（为确保每个EA交易安置都有独特的代码）	长整型
ORDER_POSITION_ID	一旦执行，位置指标马上建立一个订单。每个执行订单结果都使订单公开或修改已经存在的位置，明确指出指标的位置同时执行订单	长整型

函数 [OrderGetDouble\(\)](#) 和 [HistoryOrderGetDouble\(\)](#)

ENUM_ORDER_PROPERTY_DOUBLE

标识符	描述	类型
ORDER_VOLUME_INITIAL	订单最初交易量	双精度型
ORDER_VOLUME_CURRENT	订单当前交易量	双精度型
ORDER_PRICE_OPEN	订单中的规定价格	双精度型
ORDER_SL	斩仓值	双精度型
ORDER_TP	盈利值	双精度型
ORDER_PRICE_CURRENT	交易品种订单的当	双精度型

	前价格	
ORDER_ PRICE_ STOPLIMIT	限制停止订单的限制价格订单	双精度型

函数 [OrderGetString\(\)](#) 和 [HistoryOrderGetString\(\)](#)

ENUM_ ORDER_ PROPERTY_ STRING

标识符	描述	类型
ORDER_ SYMBOL	交易品种订单	字符串
ORDER_ COMMENT	评价订单	字符串

当使用[OrderSend\(\)](#)函数发送交易要求时，一些操作要求订单类型的指示，订单类型在类型列表中分类，尤其在[MqlTradeRequest](#)结构中，能接受到ENUM_ ORDER_ TYPE项目值。

ENUM_ ORDER_ TYPE

标识符	描述
ORDER_ TYPE_ BUY	市场购买订单
ORDER_ TYPE_ SELL	市场卖出订单
ORDER_ TYPE_ BUY_ LIMIT	限制买入待办订单
ORDER_ TYPE_ SELL_ LIMIT	限制卖出待办订单
ORDER_ TYPE_ BUY_ STOP	停止买入待办订单
ORDER_ TYPE_ SELL_ STOP	停止卖出待办订单
ORDER_ TYPE_ BUY_ STOP_ LIMIT	在到达订单价格之上，是限制买入订单安置在停止限制价格中
ORDER_ TYPE_ SELL_ STOP_ LIMIT	在到达订单价格之上，是限制卖出订单安置在停止限制价格中

每一个订单都有其作用的描述，为获得信息，与ORDER_ STATE修饰语一起使用[OrderGetInteger\(\)](#)或[HistoryOrderGetInteger\(\)](#)函数，允许值存储在 ENUM_ ORDER_ STATE 项目中。

ENUM_ ORDER_ STATE

标识符	描述
ORDER_ STATE_ STARTED	选中订单，并未被经纪人接收
ORDER_ STATE_ PLACED	接收订单
ORDER_ STATE_ CANCELED	客户撤销订单
ORDER_ STATE_ PARTIAL	履行部分订单
ORDER_ STATE_ FILLED	履行全部订单
ORDER_ STATE_ REJECTED	驳回订单

ORDER_STATE_EXPIRED	过期订单
---------------------	------

当使用[OrderSend\(\)](#) 函数发送交易请求时，在特殊架构[MqlTradeRequest](#)中的type_filling领域将会产生一个订单。ENUM_ORDER_TYPE_FILLING值是被允许的，为获得该性质的值，使用 [OrderGetInteger\(\)](#) 或 [HistoryOrderGetInteger\(\)](#) 函数和ORDER_TYPE_FILLING修饰语。

ENUM_ORDER_TYPE_FILLING

标识符	描述
ORDER_FILLING_AON	在等于或比规定价格更好的价格出现时，订单会专门执行一定成交量，如果没有足够的成交量满足订单上的对象，订单不会执行。
ORDER_FILLING_CANCEL	在等于或比规定价格更好的价格出现时，同意执行最大市场成交量，关于未填满的成交量的额外订单不会设置。
ORDER_FILLING_RETURN	在等于或比规定价格更好的价格出现时，同意执行最大市场成交量，关于未填满的成交量的额外订单会设置。

当使用[OrderSend\(\)](#) 函数发送交易请求时，订单有效期设定在[MqlTradeRequest](#) 特殊结构中的type_time领域，ENUM_ORDER_TYPE_TIME项目值是允许的，使用ORDER_TYPE_TIME修饰语和 [OrderGetInteger\(\)](#) 或 [HistoryOrderGetInteger\(\)](#) 函数可以获得该性质的值。

ENUM_ORDER_TYPE_TIME

标识符	描述
ORDER_TIME_GTC	取消订单
ORDER_TIME_DAY	前交易日订单
ORDER_TIME_SPECIFIED	过期订单

仓位属性

在开盘中执行[交易操作](#)结果，改变成交量和/或位置，或者消失。交易操作以[订单](#)为基础管理，在[交易请求](#)形式中通过 [OrderSend\(\)](#) 函数发送，对于每一个金融 [证券](#) (symbol) (交易品种) 来说，一个开仓位置是可行的，通过 [PositionGet...\(\)](#) 函数一个位置有一个属性准备设立。

函数 [PositionGetInteger\(\)](#)

ENUM_POSITION_PROPERTY_INTEGER

标识符	描述	类型
POSITION_TIME	开盘时间位置	日期时间
POSITION_TYPE	位置类型	ENUM_POSITION_TYPE
POSITION_MAGIC	位置幻数 (参看 ORDER_MAGIC)	长整型
POSITION_IDENTIFIER	位置标识符是唯一的数字，对于每个新的开仓位来说是制定的且在整个使用期内是不可更改的，翻转位置不能改变它的标识符。	长整型

函数 [PositionGetDouble\(\)](#)

ENUM_POSITION_PROPERTY_DOUBLE

标识符	描述	类型
POSITION_VOLUME	方位成交量	双精度
POSITION_PRICE_OPEN	方位开盘价格	双精度
POSITION_SL	开仓止损水平	双精度
POSITION_TP	开仓获利水平	双精度
POSITION_PRICE_CURRENT	开仓当前价位	双精度
POSITION_COMMISSION	佣金	双精度
POSITION_SWAP	积累交换	双精度
POSITION_PROFIT	当前利润	双精度

函数 [PositionGetString\(\)](#)

ENUM_POSITION_PROPERTY_STRING

标识符	描述	类型
POSITION_SYMBOL	交易品种位置	字符串

POSITION_COMMENT	方位注释	字符串
------------------	------	-----

开仓指向（买或卖）通过ENUM_POSITION_TYPE值来定义。为了获得开仓类型，使用[PositionGetInteger\(\)](#)函数POSITION_TYPE修饰语。

ENUM_POSITION_TYPE

标识符	描述
POSITION_TYPE_BUY	买入
POSITION_TYPE_SELL	卖出

交易属性

一个交易反映了以[订单](#)为基础的[交易操作](#) 执行，包括交易要求。每个交易都通过属性描述来获得有关信息，为了获取属性值，使用 [HistoryDealGet...\(\)](#) 函数类型，从类似项目中获得返回值。

函数 [HistoryDealGetInteger\(\)](#)

ENUM_DEAL_PROPERTY_INTEGER

标识符	描述	类型
DEAL__ORDER	交易 订单号	长整型
DEAL__TIME	交易时间	日期时间
DEAL__TYPE	交易类型	ENUM_DEAL_TYPE
DEAL__ENTRY	交易条目—允许进入、允许输出、颠倒	ENUM_DEAL_ENTRY
DEAL__MAGIC	订单幻数（参见 ORDER__MAGIC ）	长整型
DEAL__POSITION_ID	仓位标识符 在开仓、修正或者交易发生改变时，每个位置都有一个独特的标识符，为交易对象它被委派到所有执行订单，在全方位使用期	长整型

函数 [HistoryDealGetDouble\(\)](#)

ENUM_DEAL_PROPERTY_DOUBLE

标识符	描述	类型
DEAL__VOLUME	成交量	双精度
DEAL__PRICE	成交价格	双精度
DEAL__COMMISSION	订单佣金关	双精度
DEAL__SWAP	关闭的积累交换	双精度
DEAL__PROFIT	交易利润	双精度

函数 [HistoryDealGetString\(\)](#)

ENUM_DEAL_PROPERTY_STRING

标识符	描述	类型
-----	----	----

DEAL_SYMBOL	订单交易品种	字符串
DEAL_COMMENT	订单佣金	字符串

每笔订单都具有类型特征，允许在ENUM_DEAL_TYPE中列举价值。为了获得有关订单类型的信息，使用[HistoryDealGetInteger\(\)](#) 函数DEAL_TYPE 修饰语。

ENUM_DEAL_TYPE

标识符	描述
DEAL_TYPE_BUY	买入
DEAL_TYPE_SELL	卖出
DEAL_TYPE_BALANCE	结算
DEAL_TYPE_CREDIT	信贷
DEAL_TYPE_CHARGE	额外金额
DEAL_TYPE_CORRECTION	修正

订单不仅与建立在ENUM_DEAL_TYPE中的类型不同，也与改变方位的方式不同，这是一个简单的位置开仓，或者先前开仓的聚集（进入市场），如果相反方向订单覆盖了先前的开仓位置成交量，通过相似成交量（退出市场）的反向订单关闭位置。

所有情况在ENUM_DEAL_ENTRY值中都有描述。为了获得订单信息，使用[HistoryDealGetInteger\(\)](#) 函数DEAL_ENTRY 修饰语。

ENUM_DEAL_ENTRY

标识符	描述
DEAL_ENTRY_IN	进入
DEAL_ENTRY_OUT	出局
DEAL_ENTRY_INOUT	反面
DEAL_ENTRY_STATE	状态档案

交易操作类型

使用[OrderSend\(\)](#)函数通过向开仓位置发送命令进行交易，和放置、修改或删除待办订单命令一样。每个交易命令引用要求操作类型。交易操作在ENUM_TRADE_REQUEST_ACTIONS项目中描述。

ENUM_TRADE_REQUEST_ACTIONS

标识符	描述
TRADE_ACTION_DEAL	为规定参数的立即执行放置交易命令（市场命令）
TRADE_ACTION_PENDING	在制定环境下执行放置交易命令（待办订单）
TRADE_ACTION_SLTP	修改折仓并取走开仓利润值
TRADE_ACTION_MODIFY	修改先前放置的命令参量
TRADE_ACTION_REMOVE	删除先前放置的待办订单命令

Trade Orders in Depth Of Market

对股票来说，深度市场窗口是可行的，你能看到当前买入和卖出命令。交易操作的期望目标，要求数量和要求价格对每个命令都是制定的。

通过MQL5为获得现行状态的DOM信息，可以使用 [MarketBookGet\(\)](#) 函数，它在DOM 目录"screen shot"下的 [MqlBookInfo](#) 结构数组中。

ENUM__BOOK__TYPE

标识符	描述
BOOK__TYPE__BUY	买入命令（出价）
BOOK__TYPE__SELL	卖出命令（要价）

另见

[结构和类](#)， [DOM结构](#)， [交易操作类型](#)， [市场信息](#)

命名常量

所有使用在MQL5中的常量都能分成如下几组：

- [预定义大量代替值](#) – 在编辑过程中代替价值
- [数学常量](#) – 一些数学表达式的值
- [数值类型常量](#) – 一些简单类型限制
- [不能初始化原因代码](#) – 不能初始化原因描述
- [检测对象指针](#) - 通过 [CheckPointer\(\)](#) 函数返回计算类型指针
- [其他常量](#) – 所有其他常量

预定义大量代替值

为简化故障排除并获得MQL5程序运行的信息，有一种特殊的宏，是当时建立编辑的值。使用这些常量的最原始的方法是通过 `Print()` 函数输出值，如示例中所示。

常量	描述
<code>__LINE__</code>	源代码中的字符串数量，宏放置的位置
<code>__FILE__</code>	当前编辑文件的名称
<code>__PATH__</code>	
<code>__FUNCTION__</code>	函数名称，宏主体位置
<code>__MQ5BUILD__</code>	编译器构造数量

示例：

```
#property copyright "Copyright © 2009, MetaQuotes Software Corp."
#property link      "http://www.metaquotes.net"

//+-----+
//|  专家初始化函数          |
//+-----+
void OnInit()
{
//--- EA交易初始化输出信息示例
    Print(" __FUNCTION__ = ", __FUNCTION__, " __LINE__ = ", __LINE__ );
//--- 设置定时器事件间的间隔
    EventSetTimer(5);
//---
}

//+-----+
//|  专家无法初始化函数      |
//+-----+
void OnDeinit(const int reason)
{
//--- EA交易无法初始化输出信息示例
    Print(" __FUNCTION__ = ", __FUNCTION__, " __LINE__ = ", __LINE__ );
//---
}

//+-----+
//|  专家订单号函数          |
//+-----+
void OnTick()
{
//--- 订单收据输出信息
    Print(" __MQ5BUILD__ = ", __MQ5BUILD__, " __FILE__ = ", __FILE__ );
    Print(" __FUNCTION__ = ", __FUNCTION__, " __LINE__ = ", __LINE__ );
}
```

```

    test1(__FUNCTION__);
    test2();
//---
}
//+-----+
//|
//+-----+
void test1(string par)
{
//--- 函数内部信息输出
    Print(" __FUNCTION__ = ",__FUNCTION__," __LINE__ = ",__LINE__," par = ",par);
}
//+-----+
//|
//+-----+
void test2()
{
//--- 函数内部信息输出
    Print(" __FUNCTION__ = ",__FUNCTION__," __LINE__ = ",__LINE__);
}
//+-----+
//|
//+-----+
void OnTimer()
{
//---
    Print(" __FUNCTION__ = ",__FUNCTION__," __LINE__ = ",__LINE__);
    test1(__FUNCTION__);
}
//+-----+

```

数学常量

特殊常量包含价值为数学表达式保留，这些常量可以在程序中的任意位置使用，使用 [mathematical functions](#) 取代计算值。

常量	描述	值
M_E	e	2.71828182845904523536
M_LOG2E	log2(e)	1.44269504088896340736
M_LOG10E	log10(e)	0.434294481903251827651
M_LN2	ln(2)	0.693147180559945309417
M_LN10	ln(10)	2.30258509299404568402
M_PI	pi	3.14159265358979323846
M_PI_2	pi/2	1.57079632679489661923
M_PI_4	pi/4	0.785398163397448309616
M_1_PI	1/pi	0.318309886183790671538
M_2_PI	2/pi	0.636619772367581343076
M_2_SQRTPI	2/sqrt(pi)	1.12837916709551257390
M_SQRT2	sqrt(2)	1.41421356237309504880
M_SQRT1_2	1/sqrt(2)	0.707106781186547524401

示例：

```
//+-----+
//| 脚本程序启动函数 |
//+-----+
void OnStart()
{
//--- 打印常量值
Print("M_E = ",DoubleToString(M_E,16));
Print("M_LOG2E = ",DoubleToString(M_LOG2E,16));
Print("M_LOG10E = ",DoubleToString(M_LOG10E,16));
Print("M_LN2 = ",DoubleToString(M_LN2,16));
Print("M_LN10 = ",DoubleToString(M_LN10,16));
Print("M_PI = ",DoubleToString(M_PI,16));
Print("M_PI_2 = ",DoubleToString(M_PI_2,16));
Print("M_PI_4 = ",DoubleToString(M_PI_4,16));
Print("M_1_PI = ",DoubleToString(M_1_PI,16));
Print("M_2_PI = ",DoubleToString(M_2_PI,16));
Print("M_2_SQRTPI = ",DoubleToString(M_2_SQRTPI,16));
Print("M_SQRT2 = ",DoubleToString(M_SQRT2,16));
Print("M_SQRT1_2 = ",DoubleToString(M_SQRT1_2,16));
}
```

数值类型常量

每一简单数值类型都有确定的任务类型，并允许当MQL5程序操作正常使用时最佳化，对于较好的代码可读性和计算值的正确处理，有允许接收设置某一简单数据类型信息的常量。

常量	描述	值
CHAR_ MIN	最小值，可以代表图表类型	-128
CHAR_ MAX	最大值，可以代表图表类型	127
UCHAR_ MAX	最大值，可以代表无符字符型	255
SHORT_ MIN	最小值，可以代表短型	-32768
SHORT_ MAX	最大值，可以代表短型	32767
USHORT_ MAX	最大值，可以代表无符号短型	65535
INT_ MIN	最小值，可以代表整型	-2147483648
INT_ MAX	最大值，可以代表整型	2147483647
UINT_ MAX	最大值，可以代表无符号整型	4294967295
LONG_ MIN	最小值，可以代表长型	-9223372036854775808
LONG_ MAX	最大值，可以代表长型	9223372036854775807
ULONG_ MAX	最大值，可以代表无符号长型	18446744073709551615
DBL_ MIN	最小正值，可以代表双精度型	2.2250738585072014e-308
DBL_ MAX	最大值，可以代表双精度型	1.7976931348623158e+308
DBL_ EPSILON	最小值，满足条件 $1.0 + \text{DBL_ EPSILON} \neq 1.0$ （双精度类型）	2.2204460492503131e-016
DBL_ DIG	双精度型有效小数位数字	15
DBL_ MANT_ DIG	双精度型尾数算在内的二进制	53
DBL_ MAX_ 10_ EXP	双精度型指数角度最大小数	308
DBL_ MAX_ E XP	双精度型指数角度最大二进制值	1024
DBL_ MIN_ 10_ EXP	双精度型指数角度最小小数	(-307)
DBL_ MIN_ E XP	双精度型指数角度最小二进制值	(-1021)
FLT_ MIN	最小正值，能代表浮点类型	1.175494351e-38
FLT_ MAX	最大值，能代表浮点类型	3.402823466e+38
FLT_ EPSILON	最小值，满足因素： $1.0 + \text{DBL_ EPSILON} \neq$	1.192092896e - 07

	1.0（浮点类型）	
FLT_DIG	浮点型有效角度数位	6
FLT_MANT_DIG	浮点型二进制计算点数	24
FLT_MAX_10_EXP	浮点型指数角度最大小数值	38
FLT_MAX_EXP	浮点型最大二进制值	128
FLT_MIN_10_EXP	浮点型指数角度最小小数值	-37
FLT_MIN_EXP	浮点型指数角度最小二进制值	(-125)

示例：

```
void OnStart()
{
//--- 打印常量值
printf("CHAR_MIN = %d",CHAR_MIN);
printf("CHAR_MAX = %d",CHAR_MAX);
printf("UCHAR_MAX = %d",UCHAR_MAX);
printf("SHORT_MIN = %d",SHORT_MIN);
printf("SHORT_MAX = %d",SHORT_MAX);
printf("USHORT_MAX = %d",USHORT_MAX);
printf("INT_MIN = %d",INT_MIN);
printf("INT_MAX = %d",INT_MAX);
printf("UINT_MAX = %u",UINT_MAX);
printf("LONG_MIN = %I64d",LONG_MIN);
printf("LONG_MAX = %I64d",LONG_MAX);
printf("ULONG_MAX = %I64u",ULONG_MAX);
printf("EMPTY_VALUE = %.16e",EMPTY_VALUE);
printf("DBL_MIN = %.16e",DBL_MIN);
printf("DBL_MAX = %.16e",DBL_MAX);
printf("DBL_EPSILON = %.16e",DBL_EPSILON);
printf("DBL_DIG = %d",DBL_DIG);
printf("DBL_MANT_DIG = %d",DBL_MANT_DIG);
printf("DBL_MAX_10_EXP = %d",DBL_MAX_10_EXP);
printf("DBL_MAX_EXP = %d",DBL_MAX_EXP);
printf("DBL_MIN_10_EXP = %d",DBL_MIN_10_EXP);
printf("DBL_MIN_EXP = %d",DBL_MIN_EXP);
printf("FLT_MIN = %.8e",FLT_MIN);
printf("FLT_MAX = %.8e",FLT_MAX);
printf("FLT_EPSILON = %.8e",FLT_EPSILON);
}
```

不能初始化原因代码

使用 `UninitializeReason()` 函数返回初始化原因代码，可能值如下：

常量	值	描述
REASON_PROGRAM	0	通过调用 <code>ExpertRemove()</code> 函数EA交易终止操作
REASON_REMOVE	1	从图表中删除程序
REASON_RECOMPILE	2	程序重新编译
REASON_CHARTCHANGE	3	更改交易品种或图表周期
REASON_CHARTCLOSE	4	关闭图表
REASON_PARAMETERS	5	使用者改变输入参数
REASON_ACCOUNT	6	新账户激活
REASON_TEMPLATE	7	应用新模板
REASON_INITFAILED	8	值代表 <code>OnInit()</code> 处理器，返回非零值
REASON_CLOSE	9	

无法初始化原因代码通过预定函数 `OnDeinit`（整数常量原因）传递参数。

示例：

```
input int par=7;
//+-----+
//| 获得文本描述 |
//+-----+
string getUnitReasonText(int reasonCode)
{
    string text="";
    //---
    switch(reasonCode)
    {
        case REASON_ACCOUNT:
            text="Account was changed";break;
        case REASON_CHARTCHANGE:
            text="Symbol or timeframe was changed";break;
        case REASON_CHARTCLOSE:
            text="Chart was closed";break;
        case REASON_PARAMETERS:
            text="Input-parameter was changed";break;
        case REASON_RECOMPILE:
            text="Program "+__FILE__+" was recompiled";break;
        case REASON_REMOVE:
```

```
        text="Program "+__FILE__+" was removed from chart";break;
    case REASON_TEMPLATE:
        text="New template was applied to chart";break;
    default:text="Another reason";
}
//---
    return text;
}
//+-----+
//|  专家无法初始化函数          |
//+-----+
void OnDeinit(const int reason)
{
//--- 获得无法初始化原因代码的第一种方法
    Print(__FUNCTION__,"_Uninitialization reason code = ",reason);
//--- 获得无法初始化原因代码的第二种方法
    Print(__FUNCTION__,"_UninitReason = ",getUnitReasonText(_UninitReason));
}
```


检测对象指针

[CheckPointer\(\)](#) 函数是用来检测[对象指针](#)类型的，函数返回 ENUM_POINTER_TYPE 值中一个。

通过 [new\(\)](#) 操作创建POINTER_DYNAMIC型对象，[delete\(\) operator](#) 只能用于指针。

所有POINTER_AUTOMATIC类型其他指针，表示对象通过MQL5程序环境自动创建，该对象在使用后自动删除。

ENUM_POINTER_TYPE

常量	描述
POINTER_INVALID	错误指针
POINTER_DYNAMIC	new() 操作创建的对象指针
POINTER_AUTOMATIC	任意类型对象指针自动创建（不使用new() ）

另见

[运行错误](#)， [对象删除操作符删除](#)， [检测指针](#)

其他常量

CLR_ NONE 常量用来概述缺乏的颜色的，它表示指标不会划分 [图解对象](#) 或者 [图解系列](#)。这个常量不包括 [Web-color](#) 常量列表，当颜色参数必需时，可以应用到任何位置。

INVALID_ HANDLE 常数能用来检测处理文件（参见 [FileOpen\(\)](#) 和 [FileFindFirst\(\)](#)）。

常量	描述	值
CHARTS_ MAX	在终端与开仓图表类似的最大可能值	100
clrNONE	颜色缺乏	-1
EMPTY_ VALUE	指标缓冲区的空值	DBL_ MAX
INVALID_ HANDLE	错误处理	-1
IS_ DEBUG_ MODE	MQL5程序操作的调试方式标志	调试方式true,否则是false
NULL	任意类型零值	0
WHOLE_ ARRAY	代表很多条目剩下，直到数组末尾 如下，处理全部数组	-1
WRONG_ VALUE	常量可以隐藏 计算 传递到任何 枚举 类型数组	-1

EMPTY_ VALUE常量通常符合指标的值，但并不显示在图表中，例如，内置指标标准偏差是一个周期20，在历史中的前19个字节线并没有显示在图表中，如果使用 [iStdDev\(\)](#) 函数增添指标处理器并通过[CopyBuffer\(\)](#) 复制到指标数组里，然后这些值就与EMPTY_ VALUE相等。

可以为 [自定义指标](#) 选择指定说明自己指标的空值，当指标在图表中不显示时，使用 [PlotIndexSetDouble\(\)](#) 函数 [PLOT_ EMPTY_ VALUE](#) 修饰语。

[NULL](#) 常数可以分派到任意简单类型的变量中，或者分派到对象结构或者分类指针中。NULL为字符串变量分派表示该变量的全部初始化。

当有必要返回 [枚举](#) 值时，WRONG_ VALUE 常量为事件预定，而且还是一个错误值。例如，当我们需要报告返回值是项目中的值。列举一些函数CheckLineStyle() ,通过名称为对象返回线类型、指派。如果通过ObjectGetInteger() 类型检测结果是true，返回 [ENUM_ LINE_ STYLE](#) 中的值，否则返回WRONG_ VALUE。

```
void OnStart()
{
    if(CheckLineStyle("MyChartObject")==WRONG_VALUE)
        printf("Error line style getting.");
}
//+-----+
//| 为名称指定物件返回线型      |
//+-----+
ENUM_LINE_STYLE CheckLineStyle(string name)
{
```

```

long style;
//---
if(ObjectGetInteger(0,name,OBJPROP_STYLE,0,style))
    return((ENUM_LINE_STYLE)style);
else
    return(WRONG_VALUE);
}

```

WHOLE_ ARRAY常量为函数准备，在过程数组中需要制定元素数量：

- [ArrayCopy\(\)](#);
- [ArrayMinimum\(\)](#);
- [ArrayMaximum\(\)](#);
- [FileReadArray\(\)](#);
- [FileWriteArray\(\)](#)。

如果想要列举，指定位置中的所有数组值从头到尾都是需要处理的，你应该指定WHOLE_ ARRAY值。

当需要以调试方式对MQL5程序操作稍作改动，使用IS_ DEBUG_ MODE常量。例如，在终端记录或者增加额外图表对象中，调试方式需要额外的调试信息。

下面的例子创建了一个标签对象并在脚本运行方式中建立了自己的描述和颜色。为了运行MetaEditor中的脚本调试方式，按F5。如果能从浏览窗口中运行脚本，颜色和文本对象标签就会不同。

示例：

```

//+-----+
//|                                     Check_DEBUG_MODE.mq5 |
//|                                     Copyright © 2009, MetaQuotes Software Corp. |
//|                                     http://www.metaquotes.net |
//+-----+
#property copyright "Copyright © 2009, MetaQuotes Software Corp."
#property link      "http://www.metaquotes.net"
//+-----+
//| 脚本程序启动函数                                |
//+-----+
void OnStart()
{
//---
    string label_name="invisible_label";
    if(ObjectFind(0,label_name)<0)
    {
        Print("Object",label_name,"not found. Error code = ",GetLastError());
        //--- 创建标签
        ObjectCreate(0,label_name,OBJ_LABEL,0,0,0);
        //--- 设置 X 坐标
        ObjectSetInteger(0,label_name,OBJPROP_XDISTANCE,200);
        //--- 设置 Y 坐标
        ObjectSetInteger(0,label_name,OBJPROP_YDISTANCE,300);
        ResetLastError();
        if(IS_DEBUG_MODE) // 调试方式
    }
}

```

```
{
    //--- 显示脚本执行模式信息
    ObjectSetString(0,label_name,OBJPROP_TEXT,"DEBUG MODE");
    //--- 设置文本颜色成红色
    if(!ObjectSetInteger(0,label_name,OBJPROP_COLOR,clrRed))
        Print("Unable to set the color. Error",GetLastError());
}
else // 操作模式
{
    ObjectSetString(0,label_name,OBJPROP_TEXT,"RELEASE MODE");
    //--- 设置文本颜色为不可见
    if(!ObjectSetInteger(0,label_name,OBJPROP_COLOR,CLR_NONE))
        Print("Unable to set the color. Error ",GetLastError());
}
ChartRedraw();
DebugBreak(); // 如果处于调试模式，则会终止
}
```

另见

[DebugBreak](#), [执行MQL程序属性](#)

数据结构

MQL5语言提供5种预先 [结构](#) ..

- [MqlDateTime](#) 预定给 [日期和时间](#) ..
- [MqlParam](#) 当增加指标时使用 [IndicatorCreate\(\)](#) 函数发送输入参数；
- [MqlRates](#) 试图操纵 [历史数据](#)，包括价格，成交量和传播的数据信息；
- [MqlBookInfo](#) 为了获取[市场深度](#)信息；
- [MqlTradeRequest](#) 为了增加[交易操作](#)交易需求；
- [MqlTradeResult](#) 通过[OrderSend\(\)](#) 函数，获得交易服务器回应[交易请求](#)..
- [MqlTick](#) 为最多需求信息的当前价格设计快速检索。

MqlDateTime

数据类型结构包括int的8个字段：

```
struct MqlDateTime
{
    int year;           // 年
    int mon;            // 月
    int day;            // 天
    int hour;           // 小时
    int min;            // 分钟
    int sec;            // 秒
    int day_of_week;    // 一周中的每天 (0-周日, 1-周一, ..., 6-周六)
    int day_of_year;    // 一年中的日期号 (1月1日 赋予零值)
};
```

注释

闰年每年天数day_of_year，从三月开始，数字就与非闰年类似。

示例：

```
void OnStart()
{
    //---
    datetime date1=D'2008.03.01';
    datetime date2=D'2009.03.01';

    MqlDateTime str1,str2;
    TimeToStruct(date1,str1);
    TimeToStruct(date2,str2);
    printf("%02d.%02d.%4d, day of year = %d",str1.day,str1.mon,
        str1.year,str1.day_of_year);
    printf("%02d.%02d.%4d, day of year = %d",str2.day,str2.mon,
        str2.year,str2.day_of_year);
}
/* Result:
    01.03.2008, day of year = 60
    01.03.2009, day of year = 59
*/
```

另见

[TimeToStruct](#), [结构和类](#)

指标输入参量的结构 (MqlParam)

当使用 [IndicatorCreate\(\)](#) 函数增加[技术指标](#) 处理时，MqlParam参数结构被提供 [输入参量](#) 特殊设计。

```
struct MqlParam
{
    ENUM_DATATYPE    type;                // 输入参量类型, ENUM\_DATATYPE 值
    long             integer_value;        // 存储整数类型域
    double           double_value;        // 存储双精度类型域
    string           string_value;        // 存储字符串类型域
};
```

一个指标输入的所有参数都以MqlParam类型数组形式传输的，每一元素的类型领域都通过元素传递指定了数据类型。指标值的每一个元素必须在恰当的领域放在第一位(integer_value, double_value 或者 string_value) 在类型领域中依靠[ENUM_DATATYPE](#) 值指定。

如果IND_CUSTOM值以指标类型第三个传递到[IndicatorCreate\(\)](#) 函数中，输入参数数组中的第一个元素一定是 [ENUM_DATATYPE](#) 项目中TYPE_STRING 值，而string_value领域中一定包含[自定义指标](#) 名称。

Mql比率

结构存储有关价位、交易量和传递的信息。

```
struct MqlRates
{
    datetime time;           // 周期开始时间
    double open;             // 开盘价
    double high;             // 周期最高价
    double low;              // 周期最低价
    double close;            // 收盘价
    long tick_volume;        // 订单交易量
    int spread;              // 点差
    long real_volume;        // 交易量
};
```

示例：

```
void OnStart()
{
    MqlRates rates[];
    int copied=CopyRates(NULL,0,0,100,rates);
    if(copied<=0)
        Print("Error copying price data ",GetLastError());
    else Print("Copied ",ArraySize(rates)," bars");
}
```

另见

[复制比率](#)， [接入时间序列](#)

MqlBookInfo

提供市场深度信息。

```
struct MqlBookInfo
{
    ENUM_BOOK_TYPE   type;           // 从 ENUM\_BOOK\_TYPE 列举出的订单类型
    double            price;          // 价格
    long              volume;         // 交易量
};
```

注释

MqlBookInfo

DOM只有在一些交易品种中可以使用。

示例：

```
MqlBookInfo priceArray[];
bool getBook=MarketBookGet(NULL,priceArray);
if(getBook)
{
    int size=ArraySize(priceArray);
    Print("MarketBookInfo about ",Symbol());
}
else
{
    Print("Failed to receive DOM for the symbol ",Symbol());
}
```

另见

[MarketBookAdd](#) -[MarketBookRelease](#), [MarketBookGet](#), [DOM中交易订单](#), [数据类型](#)

交易请求结构 (MqlTradeRequest)

客户端与交易服务器执行其他安置操作相互作用，通过使用交易请求来执行。交易请求由特殊预定义 MqlTradeRequest类型的结构来体现，包含必要的执行交易订单。该请求执行结果由MqlTradeResult的类型结构来表示。

```
struct MqlTradeRequest
{
    ENUM_TRADE_REQUEST_ACTIONS    action;           // 交易操作类型
    ulong                          magic;             // EA交易 ID ( 魔幻数字 )
    ulong                          order;             // 订单号
    string                         symbol;            // 交易的交易品种
    double                         volume;            // 一手需求的交易量
    double                         price;             // 价格
    double                         stoplimit;         // 订单止损限价点位
    double                         sl;                // 订单止损价位点位
    double                         tp;                // 订单盈利价位点位
    ulong                          deviation;         // 需求价格最可能的偏差
    ENUM_ORDER_TYPE               type;              // 订单类型
    ENUM_ORDER_TYPE_FILLING       type_filling;      // 订单执行类型
    ENUM_ORDER_TYPE_TIME         type_time;         // 订单执行时间
    datetime                      expiration;        // 订单终止期 ( 为 ORDER_TIME_SPECIFIED 类型订单 )
    string                        comment;           // 订单注释
};
```

字段描述

字段	描述
action	交易操作类型，可以是 ENUM_TRADE_REQUEST_ACTIONS 项目值中的一个。
magic	EA交易ID，允许操作分析交易命令过程，当发送交易请求时，每个EA交易都能建立自己独特的ID
order	命令按钮，用来修改待办订单。
symbol	交易品种命令，不需要修改命令和关闭放置操作。
volume	要求命令成交手数，标识每笔订单真实成交量取决于 订单执行类型 。
price	价格，必须执行到达命令，交易品种的市场底单，它的执行类型是“市场执行”（ SYMBOL_TRADE_EXECUTION_MARKET ）， TRADE_ACTION_DEAL 类型，不要求价格说明。
stoplimit	价格价值，当价格达到price值时（必要条件），StopLimit安置待办订单，直到待办订单部能安置。
sl	在不利于价格活动的情况下的止损数值
tp	在优惠价格活动的情况下目标数值
deviation	最大价格偏差，指定在 points 中
type	命令类型，可以是 ENUM_ORDER_TYPE 项目值中的一个
type_filling	命令执行类型，可以是 ENUM_ORDER_TYPE_FILLING 项目值中的一个

type__time	命令终结类型，可以是 ENUM_ORDER_TYPE_TIME 项目值中的一个
expiration	命令终结期限（ ORDER_TIME_SPECIFIED 命令类型）
comment	命令注解文本

发送执行命令 [交易操作](#) 需要使用 [OrderSend\(\)](#) 函数。对于每个交易操作来说都需要指定强制域；也需要填满可选域。有7个可行情况发送交易命令：

请求执行

开仓交易命令在执行请求模式中（交易根据现行价格要求），需要指定如下9种域：

- 执行
- 交易品种
- 交易量
- 价格
- 止损水平
- 盈利水平
- 误差
- 类型
- _ filling类型

也有可能指定“magic”和“comment”域值

立即执行

开仓交易命令在立即执行模式中（流动价格交易），要求如下9种域的规格：

- 执行
- 交易品种
- 交易量
- 价格
- 止损水平
- 盈利水平
- 误差
- 类型
- _ filling类型

有可能指定“magic”和“comment”域值。

买卖执行

开仓交易命令在买卖执行方式中，要求指定如下5中域：

- 执行
- 交易品种
- 交易量
- 类型
- _ filling类型

也有可能指定“magic”和“comment”域值。

SL & TP修正

交易命令在StopLoss 和/或 TakeProfit 价格水平上修改，要求指定如下5种域：

- 执行
- 交易品种
- 止损水平
- 盈利水平

待办订单

交易命令安置在待办订单中，要求指定如下11种域：

- 行动
- 交易品种
- 交易量
- 价格
- 限制停止
- 止损水平
- 盈利水平
- 类型
- _ filling 类型
- _ time 类型
- 期满

也有可能指定“magic ”和“comment ”域值。

修改待办订单

交易命令在待办订单价格中修改，它要求指定如下7种域：

- 行动
- 命令
- 价格
- 止损水平
- 盈利水平
- _ time类型
- 期满

删除待办订单

交易命令删除一个待办订单，它要求指定如下2种域

- 行动
- 命令

另见

[结构和类](#), [交易函数](#) - [订单属性](#)

交易要求检查结构结果（MQL交易检测结果）

在向服务器发送 [交易操作](#) 要求时，推荐检测。该检测使用 [OrderCheck\(\)](#) 函数执行，检测要求和 MqlTradeCheckResult 结构变量类型已通过传递，检测结果会记录该变量。

```
struct MqlTradeCheckResult
{
    uint      retcode;           // 应答码
    double    balance;          // 交易执行后的余额
    double    equity;           // 交易执行后的净值
    double    profit;           // 浮动利润
    double    margin;           // 保证金规定额
    double    margin_free;      // 自由保证金
    double    margin_level;     // 保证金比例
    string    comment;          // 应答码注释（描述错误）
};
```

域描述

域	描述
retcode	返回代码
balance	在执行交易操作后账户余额
equity	在执行交易操作后净值
profit	在执行交易操作后的浮点利润值
margin	为执行操作要求的保证金
margin_ free	在执行交易操作后的可用保证金
margin_ level	在执行交易操作后建立保证金水平
comment	注解文本回应代码，错误描述

另见

[交易请求结构](#)，[当前价格结构](#)，[发送订单](#)，[订单检测](#)

交易请求结果结构（MqlTradeResult）

[交易请求](#)结果，交易服务器返回有关交易要求处理结果的相关数据来作为类型的MqlTradeResult 特殊预定义结构。

```
struct MqlTradeResult
{
    uint      retcode;           // 操作返回代码
    ulong     deal;             // 交易订单号，如果完成的话
    ulong     order;            // 订单号，如果下订单的话
    double    volume;           // 交易交易量，经纪人确认的
    double    price;            // 交易价格，经纪人确认的
    double    bid;              // 当前买入价
    double    ask;              // 当前卖出价
    string     comment;         // 经纪人操作注释（默认填充操作描述）
};
```

域描述

域	描述
retcode	交易服务器的 返回代码
deal	交易 标签，如果执行订单，对于 TRADE_ACTION_DEAL 交易操作类型来说是可行的
order	订单 标签，如果安置标签，对于 TRADE_ACTION_PENDING 交易操作类型来说是可行的
volume	交易成交量，由经纪人确认，取决于 订单填充类型
price	订单价格，由经纪人确认，取决于误差域 交易请求 和/或 交易操作
bid	当前市场买价（请求报价）
ask	当前市场卖价（请求报价）
comment	经纪人评论操作（通过操作描述默认填满）

交易操作结果返回到MqlTradeResult类型变量，以第二秒参量 [OrderSend\(\)](#) 去执行 [交易操作](#)。

示例：

```

//+-----+
//|  随着处理结果发送交易请求      |
//+-----+
bool MyOrderSend(MqlTradeRequest request,MqlTradeResult result)
{
//--- 重置最后一个错误代码到零
    ResetLastError();
//--- 发送请求
    bool success=OrderSend(request,result);
//--- 如果结果失败-找出为什么
    if(!success)
    {
        int answer=result.retcode;
        Print("TradeLog: Trade request failed. Error = ",GetLastError());
        switch(answer)
        {
            //--- 重新报价
            case 10004:
            {
                Print("TRADE_RETCODE_REQUOTE");
                Print("request.price = ",request.price,"    result.ask = ",
                    result.ask," result.bid = ",result.bid);
                break;
            }
            //--- 订单没有被服务器接受
            case 10006:
            {
                Print("TRADE_RETCODE_REJECT");
                Print("request.price = ",request.price,"    result.ask = ",
                    result.ask," result.bid = ",result.bid);
                break;
            }
            //--- 无效价格
            case 10015:
            {
                Print("TRADE_RETCODE_INVALID_PRICE");
                Print("request.price = ",request.price,"    result.ask = ",
                    result.ask," result.bid = ",result.bid);
                break;
            }
            //--- 无效 SL and/or TP
            case 10016:
            {
                Print("TRADE_RETCODE_INVALID_STOPS");
                Print("request.sl = ",request.sl," request.tp = ",request.tp);
                Print("result.ask = ",result.ask," result.bid = ",result.bid);
                break;
            }
            //--- 无效交易量
            case 10014:
            {
                Print("TRADE_RETCODE_INVALID_VOLUME");
                Print("request.volume = ",request.volume,"    result.volume = ",
                    result.volume);
                break;
            }
            //--- 无足够的钱进行交易操作
            case 10019:
            {
                Print("TRADE_RETCODE_NO_MONEY");
                Print("request.volume = ",request.volume,"    result.volume = ",

```

```
        result.volume,"    result.comment = ",result.comment);  
        break;  
    }  
    ///--- 一些其他的原因，输出服务器响应码  
    default:  
    {  
        Print("Other answer = ",answer);  
    }  
    }  
    ///--- 通过返回错误值通知服务器请求的不成功结果  
    return(false);  
    }  
    ///--- 发送命令 () 返回真值 - 复制答案  
    return(true);  
    }
```


返回当前价格结构 (MqlTick)

该构造是存储交易类型的最新价位，它能最快检索有关当前价格的最需要信息。

```
struct MqlTick
{
    datetime    time;           // 价格更新的最近时间
    double      bid;            // 当前买入价
    double      ask;            // 当前卖出价
    double      last;           // 上一次交易价格 (Last)
    ulong       volume;         // 上一次价格交易量
};
```

通过单一调用 [SymbolInfoTick\(\)](#) 函数，MqlTick type 变量允许获得要价，开价，最后价和成交量的值。

示例：

```
void OnTick()
{
    MqlTick last_tick;
    //---
    if(SymbolInfoTick(Symbol(),last_tick))
    {
        Print(last_tick.time,": Bid = ",last_tick.bid,
              " Ask = ",last_tick.ask," Volume = ",last_tick.volume);
    }
    else Print("SymbolInfoTick() failed, error = ",GetLastError());
    //---
}
```

另见

[结构和类](#)

错误和警告代码

该章节包括如下描述：

- [交易服务器返回代码](#) – 通过[OrderSend\(\)](#) 函数分析[交易请求](#) 发送结果；
- [编辑器警告](#) – 警告信息代码出现在编辑器中（不是错误）；
- [编译错误](#) – 尝试编译失败后的错误信息代码；
- [运行时间错误](#) – MQL5程序的属性错误代码可以通过使用 [GetLastError\(\)](#) 函数获得。

交易服务器返回代码

通过使用[OrderSend\(\)](#)函数，所有执行交易操作的请求都以[MqlTradeRequest](#)交易请求结构发送。函数执行结果代替[MqlTradeResult](#)结构，retcode域包括交易服务器返回代码，所有代码都在ENUM_TRADE_RETURN_CODES项目中描述。

ENUM_TRADE_RETURN_CODES

代码	常量	描述
10004	TRADE_RETCODE_REQUOTE	报价请求
10006	TRADE_RETCODE_REJECT	拒绝请求
10007	TRADE_RETCODE_CANCEL	交易者取消请求
10008	TRADE_RETCODE_PLACED	安置命令
10009	TRADE_RETCODE_DONE	完成要求
10010	TRADE_RETCODE_DONE_PARTIAL	请求部分完成
10011	TRADE_RETCODE_ERROR	请求处理错误
10012	TRADE_RETCODE_TIMEOUT	超时取消请求
10013	TRADE_RETCODE_INVALID	无效请求
10014	TRADE_RETCODE_INVALID_VOLUME	请求中无效成交量
10015	TRADE_RETCODE_INVALID_PRICE	请求中的无效价格
10016	TRADE_RETCODE_INVALID_STOPS	请求中的无效访问
10017	TRADE_RETCODE_TRADE_DISABLED	关闭交易
10018	TRADE_RETCODE_MARKET_CLOSED	收市
10019	TRADE_RETCODE_NO_MONEY	没有足够的钱实现请求
10020	TRADE_RETCODE_PRICE_CHANGED	改变价格
10021	TRADE_RETCODE_PRICE_OFF	没有报价处理请求
10022	TRADE_RETCODE_INVALID_EXPIRATION	请求中的无效命令截止日期
10023	TRADE_RETCODE_ORDER_CHANGED	改变命令状态
10024	TRADE_RETCODE_TOO_MANY_REQUESTS	太频繁的请求
10025	TRADE_RETCODE_NO_CHANGES	不改变请求
10026	TRADE_RETCODE_SERVER_DISABLES_AT	服务器无效自动交易
10027	TRADE_RETCODE_CLIENT_DISABLES_AT	客户端无效自动交易

10028	TRADE__RETCODE__LOCKED	处理锁住请求
10029	TRADE__RETCODE__FROZEN	命令或安置冻结
10030	TRADE__RETCODE__INVALID__FILL	无效命令填满字节
10031	TRADE__RETCODE__CONNECTION	与服务器无连接
10032	TRADE__RETCODE__ONLY__REAL	只在流水账允许操作
10033	TRADE__RETCODE__LIMIT__ORDERS	待办订单数量达到限制
10034	TRADE__RETCODE__LIMIT__VOLUME	订单成交量和交易品种位置达到限制

编译器警告

编译器警告信息只在情报目的和不出现错误信息的情况下显示。

代码	描述
21	在时间日期字符串中日期显示不完全记录
22	在时间日期中日期的错误代码，要求 年 $1970 \leq X \leq 3000$ 月 $0 < X \leq 12$ 日 $0 < X \leq 31/30/28 (29) \dots$
23	在时间日期字符串中时间的错误代码，要求 小时 $0 \leq X < 24$ 分钟 $0 \leq X < 60$
24	在RGB中的无效颜色，RGB部件中的元素在0到255之间
25	换码顺序中的位置字符 Known: <code>\n\r\t\\\\"X\x</code>
26	函数局部变量里太大的字符 (>512kb) 会减少数字
29	列举项已经定义 (副本) -把项目添加到第一个定义里
30	主要宏
32	构造函数一定是空型
33	解构函数一定要空型
34	常量不在整数范围内 ($X > _UI64_MAX$ $X < _I64_MIN$)，并且转变为双精度型函数
35	太长的HEX—超过16个重要字节 (开出高级半字节形式)
36	在HEX字符串"0x"中没有半字节
37	无函数—不执行任何命令
41	无主体函数，不被调用
43	在铸字过程中可能丢失的数据。示例： <code>int x = (double) z</code>
44	转变常量时精确度缺失 (数据)。示例： <code>int x = M_PI</code>
45	在比较操作中两个操作数标志之间的区别。示例： <code>(char) c1 > (uchar) c2</code>
46	输入函数遇到的问题—需声明#import或者输入隐蔽函数
47	太大的说明—在可执行文件中，不包括额外字节
48	指标缓冲区的数量比所要求的少
49	在指标中无颜色图解绘图
50	无图解系列描绘指标
51	在脚本中找不到'OnStart' 处理器函数
52	以错误参量定义'OnStart' 处理器函数

53	只在脚本中定义'OnStart' 函数
54	以错误参量定义'OnInit' 函数
55	在脚本中不使用'OnInit'函数
56	以错误参量定义'OnDeinit'函数
57	在脚本中不使用'OnDeinit'函数
58	定义两种'OnCalculate'函数，使用 固定价格数组 的OnCalculate()。
59	当计算复杂 整数 常量时检测过量装满
60	或许，变量 没有初始化
61	该声明会使不能参考特定行上声明的 局部变量
62	该声明会使不能参考特定行上声明的 全局变量
63	不能用于 静态分配数组
64	变量声明隐藏 预定义变量
65	true/false

编译错误

在编辑过程中通过内置编译器，MT5显示了关于系统错误删除的错误信息。这些错误列举在给定的下表中。为了编译可执行的源代码，按F7。程序包括的错误直到被淘汰的编辑器定义后才能编辑。

代码	描述
100	文件读数误差
101	*. EX5错误文件是用来书写记录的
103	没有足够内存完成编辑
104	编辑器不能识别的空语法
105	#include文件中的错误文件名
106	#include 包含文件中的访问文件错误（也许因为文件不存在）
108	与#define不匹配的名称
109	预处理器的位置命令（有效 #include，#define，#property，#import）
110	编辑器位置交易品种
111	无法实施的函数（现行描述，不是主体）
112	双引号(") 省略
113	尖角括号(<) 或双引号(") 省略
114	单引号(') 省略
115	回尖括号">"省略
116	声明中未标识的类型
117	无返回操作或者在所有分支中未发现返回
118	期望调用参量括号
119	错误书写EX5
120	无效访问数组
121	函数不是缺省型，返回操作会返回值
122	破坏者的特殊声明
123	冒号":"缺失
124	变量已声明
125	标识符变量已声明
126	变量名称太长（>250字节）
127	标识符变量已定义
128	结构未定义
129	相同名称构造成员已定义

130	没有相关构造成员
131	违背一对分支
132	开括号"("预期
133	不对称花括号（不是"} "）
134	难以编译（太多分支，满溢内部堆积水平）
135	（打开错误文件阅读）
136	没有足够内存下载源文件
137	预期变量
138	参考不能初始化
140	预期分配（抵达描述）
141	开括号"{ "预期
142	参数只执行 动态数组
143	使用"void"类型不能接受
144	没有成对的 ")" " 或者 "]"，例如 "(或" [" 缺席
145	没有成对的 "(or" ["，如下 ") "or"] "缺席
146	错误数组大小
147	太多参量(> 64)
149	该符号不是预期的
150	无效使用操作（无效操作数）
151	不允许空型表达式
152	期望操作
153	滥用终止
154	分号";"预期
155	逗号","预期
156	分类类型，而不是结构
157	预期表达式
158	HEX中的"non HEX character"或者太长数字（数字大于511）
159	字符串长度超过65534字节
160	函数定义无法接受
161	不是预期的程序结尾
162	禁止架构的前置声明
163	函数名称已定义并有另一个返回类型

164	函数名称已定义并建立不同参量
165	函数名称已定义并应用
166	调用时函数重载并未发现
167	缺省型返回值函数不能返回值
168	函数不能定义
170	预期价值
171	case 表达式中只有整数常量是有效的
172	switch中的case值已经使用
173	预期整数
174	预期#import表达式文件名称
175	在总体水平上，表达式不允许使用
176	在前";"省略插入语") "
177	预计左标记变量
178	不使用表达式结果
179	说明变量在case 中不允许使用
180	从字符串到数字的隐式变换
181	数字的隐式变换到字符串
182	重载函数的不明确调用（几种重载安置）
183	不包含if的非法符号else
184	无需使用switch的无效case或者default
185	省略号的不恰当使用
186	初始化序列比初始化常量元素多
187	预期case常数
188	要求表达式常数
189	不能改变的常数变量
190	预期回括号或逗号（说明数组成员）
191	
192	举例不能访问编辑器(const, extern, static)
193	使用不同值声明列举成员
194	相同名称定义变量
195	相同名称定义结构
196	期望列举成员名称

197	预期整数表达式
198	常数表达式除以0
199	函数参量的错误代码
200	通过引用的参量一定是变量
201	通过期望引用传递相同类型的变量
202	恒定变量不能通过不恒定变量传递
203	要求正整数常量
204	访问未保护分类成员失败
205	以另一种方式输入以定义
208	未生成可执行文件
209	指标未发现'OnCalculate' 入口点
210	只有在环形使用时能执行连续操作
211	错误访问私人分类成员（关闭）
213	未声明的结构方式或分类
214	错误访问私人分类方式（关闭）
216	不允许结构复制对象
218	数组转变指标
219	不允许数组在结构或分类中初始化
220	分类构造函数不能有参数
221	分类破坏者不能有参数
222	相同名称的分类方式或结构和参量已声明
223	预期操作数
224	以相同名称分类方式或结构，但使用不同参量（声明！=实现）
225	未声明定义函数
227	重载函数的歧义调用（几个多载集合参量的准确匹配）
228	预期变量名称
229	在此不能声明引用
230	以列举名称已经使用
232	预期的分类或结构
235	不能调用'delete'操作去删除数组
236	预期'while'操作
237	'delete'操作需要指针

238	已经对'switch'有'default'
239	句法错误
240	逃脱序列只能在字符串中发生 (以 '\开头)
241	需要数组一方括号[], 不适用于数组, 或者没有数组能参与数组参量传递
242	不能通过初始化序列初始化
243	输入不能定义
244	句法错误的优化程序错误
245	声明太多结构 (想要简化程序)
246	不允许参数转换
247	错误使用 “ 删除 ” 操作
248	不允许宣布参考指针
249	不允许宣布提及参考
250	不允许宣布指标
251	参量列表中不允许架构宣布
252	铸字过程中的无效操作
253	指标只在分类或架构中声明
256	未定义的标识符
257	可执行代码优化程序错误
258	可执行代码生成错误
260	“ 转换 ” 操作的无效表达式
261	字符串常数溢出, 简化程序
262	不能转换列举项目
263	不能使用数据 “ 虚拟 ” (分类或结构成员)
264	不能调用分类保护方法
265	拒绝虚拟函数返回不同类型
266	结构不能遗传分类
267	分类不能遗传结构
268	构造函数不能是虚拟的 (不允许使用virtual说明符)
269	结构不能有虚拟方式
270	函数必须有主体
271	禁止系统函数重载 (终端函数)
272	Const 标识符对函数来说没有分类或结构成员

273	不允许使用 const 数据（分类或结构成员）
274	在常数方法中不允许改变分类成员
275	不允许声明超过64个输入参量（输入变量）
276	不适合的初始化队列
277	参量丢失的缺省值（默认参量的特殊声明）
278	主要的缺省参量（声明和执行的同值）
279	不允许为恒定对象调用不恒定方式
280	对象有必要访问成员（建立无分类或结构指示）
281	已声明的结构名称不能在申报中使用
282	结构名称不能使用分类成员名称
283	错误类型或者'OnCalculate'接入点参量的组合
284	未受保护的转换（关闭继承）
285	构造和数组不能以输入变量形式使用
286	对于constructor/destructor来说Const无效
287	表示日期时间的错误字符串
288	未知属性（#property）
289	属性错误值
290	#property中的性质无效索引
291	调用省略参数- <func (x,) >
293	对象通过引用传递
294	数组通过引用传递
295	函数以可输出类型声明
296	函数不能以输出形式声明
297	禁止输出录入函数
298	导入函数不能有此参量（禁止传递指针，分类或者结构包括动态数据、指针，分类等）
299	需要类
300	#import 没有关闭
301	预期初始化序列
302	类型错配
303	外部变量 已经初始化
304	没有 输出 函数或者 接入点
305	不允许直接调用 构造函数

306	程序没有被声明为常量
307	程序没有被声明为常量
308	
309	
310	
311	
312	
313	
314	
315	
316	
317	
318	
319	(16)
320	
321	EnumToString()
322	
323	(32) BMP- 24
324	
325	
326	()
327	
328	,
329	
330	

331	_____
332	
333	_____
334	_____

运行时间错误

[GetLastError\(\)](#) 函数是用来返回储存在预定义变量 [_LastError](#) 中的上一错误代码。该值可以通过 [ResetLastError\(\)](#) 函数重置为0

常量	代码	描述
ERR_SUCCESS	0	操作成功完成
ERR_INTERNAL_ERROR	4001	意外内部错误
ERR_WRONG_INTERNAL_PARAMETER	4002	客户端函数内部调用的错误参数
ERR_INVALID_PARAMETER	4003	当调用系统函数时的错误参量
ERR_NOT_ENOUGH_MEMORY	4004	没有足够空间执行系统函数
ERR_STRUCT_WITHOBJECTS_ORCLASS	4005	结构包括字符串和/或动态数组和/或结构对象和/或分类
ERR_INVALID_ARRAY	4006	数组错误类型，错误大小，或者动态数组的损害对象
ERR_ARRAY_RESIZE_ERROR	4007	数组没有足够空间重置，或者没有能够改变静态数组的大小
ERR_STRING_RESIZE_ERROR	4008	没有足够内存重置字符串
ERR_NOTINITIALIZED_STRING	4009	没有初始化字符串
ERR_INVALID_DATETIME	4010	无效日期和/时间
ERR_ARRAY_BAD_SIZE	4011	要求数组大小超过2GB
ERR_INVALID_POINTER	4012	错误指针
ERR_INVALID_POINTER_TYPE	4013	错误指针类型
ERR_FUNCTION_NOT_ALLOWED	4014	系统函数不允许调用
图表		
ERR_CHART_WRONG_ID	4101	错误图表ID
ERR_CHART_NO_REPLY	4102	图表不能回应
ERR_CHART_NOT_FOUND	4103	图表未发现
ERR_CHART_NO_EXPERT	4104	图表中没有EA交易可以处理事件
ERR_CHART_CANNOT_OPEN	4105	图表打开错误
ERR_CHART_CANNOT_CHANGE	4106	改变图表交易品种或周期失败
ERR_CHART_WRONG_TIMER_PARAMETER	4107	定时器错误参量
ERR_CHART_CANNOT_CREATE_TIMER	4108	增加定时器失败
ERR_CHART_WRONG_PROPERTY	4109	错误图表属性ID
ERR_CHART_SCREENSHOT_FAILED	4110	错误生成截屏

ERR_CHART_NAVIGATE_FAILED	4111	错误操作图表
ERR_CHART_TEMPLATE_FAILED	4112	错误申请模板
ERR_CHART_WINDOW_NOT_FOUND	4113	未找到窗口包含的指标
ERR_CHART_INDICATOR_CANNOT_ADD	4114	错误添加指标到图表
ERR_CHART_INDICATOR_CANNOT_DEL	4115	
ERR_CHART_INDICATOR_NOT_FOUND	4116	
图解对象		
ERR_OBJECT_ERROR	4201	图解对象的误差
ERR_OBJECT_NOT_FOUND	4202	未发现图解对象
ERR_OBJECT_WRONG_PROPERTY	4203	图解对象属性的错误ID
ERR_OBJECT_GETDATE_FAILED	4204	不能与值相一致获得日期
ERR_OBJECT_GETVALUE_FAILED	4205	不能与日期相一致获得值
买卖信息		
ERR_MARKET_UNKNOWN_SYMBOL	4301	未知交易品种
ERR_MARKET_SELECT_ERROR	4302	市场报价中未被挑选出来的交易品种
ERR_MARKET_WRONG_PROPERTY	4303	交易品种属性的错误标识符
ERR_MARKET_LASTTIME_UNKNOWN	4304	时间的最后标志未识别（无标志）
历史访问		
ERR_HISTORY_NOT_FOUND	4401	需求历史未找到
ERR_HISTORY_WRONG_PROPERTY	4402	历史属性的错误ID
全局变量		
ERR_GLOBALVARIABLE_NOT_FOUND	4501	客户端全局变量未找到
ERR_GLOBALVARIABLE_EXISTS	4502	相同名称客户端全局变量已经存在
ERR_MAIL_SEND_FAILED	4510	发送邮件失败
ERR_PLAY_SOUND_FAILED	4511	声音播放失败
ERR_MQL5_WRONG_PROPERTY	4512	程序属性错误标识符
ERR_TERMINAL_WRONG_PROPERTY	4513	客户端属性错误标识符
ERR_FTP_SEND_FAILED	4514	通过ftp发送文件失败
自定义指标缓冲区		

ERR_BUFFERS_NO_MEMORY	4601	没有足够内存建立指标缓冲区
ERR_BUFFERS_WRONG_INDEX	4602	错误指标缓冲区索引
自定义指标属性		
ERR_CUSTOM_WRONG_PROPERTY	4603	自定义指标的错误ID
账户		
ERR_ACCOUNT_WRONG_PROPERTY	4701	错误账户属性ID
ERR_TRADE_WRONG_PROPERTY	4751	错误交易属性ID
ERR_TRADE_DISABLED	4752	EA交易禁止交易
ERR_TRADE_POSITION_NOT_FOUND	4753	未找到位置
ERR_TRADE_ORDER_NOT_FOUND	4754	未找到命令
ERR_TRADE_DEAL_NOT_FOUND	4755	未找到订单
ERR_TRADE_SEND_FAILED	4756	交易需求发送失败
指标		
ERR_INDICATOR_UNKNOWN_SYMBOL	4801	未知交易品种
ERR_INDICATOR_CANNOT_CREATE	4802	不能创建指标
ERR_INDICATOR_NO_MEMORY	4803	没有足够内存添加指标
ERR_INDICATOR_CANNOT_APPLY	4804	指标不能适应另一指标
ERR_INDICATOR_CANNOT_ADD	4805	错误申请指标到图表
ERR_INDICATOR_DATA_NOT_FOUND	4806	需求数据未找到
ERR_INDICATOR_WRONG_HANDLE	4807	错误指标处理
ERR_INDICATOR_WRONG_PARAMETERS	4808	当建立指标时使用的错误参量数
ERR_INDICATOR_PARAMETERS_MISSING	4809	当建立指标时没有参量
ERR_INDICATOR_CUSTOM_NAME	4810	在数组中的第一参量一定与自定义指标相同
ERR_INDICATOR_PARAMETER_TYPE	4811	当建立指标时，数组中无效参量类型
ERR_INDICATOR_WRONG_INDEX	4812	错误索引需求的指标缓冲区
市场深度		
ERR_BOOKS_CANNOT_ADD	4901	不能添加市场深度
ERR_BOOKS_CANNOT_DELETE	4902	不能移动市场深度
ERR_BOOKS_CANNOT_GET	4903	深度市场数据未获得
ERR_BOOKS_CANNOT_SUBSCRIBE	4904	从市场深度接收新数据时出错
文件操作		

ERR_TOO_MANY_FILES	5001	不能同时打开超过64个文件
ERR_WRONG_FILENAME	5002	无效文件名
ERR_TOO_LONG_FILENAME	5003	太长文件名
ERR_CANNOT_OPEN_FILE	5004	文件打开错误
ERR_FILE_CACHEBUFFER_ERROR	5005	没有足够内存隐藏阅读
ERR_CANNOT_DELETE_FILE	5006	文件删除错误
ERR_INVALID_FILEHANDLE	5007	文件处理关闭，或者根本没打开
ERR_WRONG_FILEHANDLE	5008	错误文件处理
ERR_FILE_NOTTOWRITE	5009	文件必须打开编辑
ERR_FILE_NOTTOREAD	5010	文件必须打开阅读
ERR_FILE_NOTBIN	5011	文件必须以二进制打开
ERR_FILE_NOTTXT	5012	文件必须以文本形式打开
ERR_FILE_NOTTXTORCSV	5013	文件必须以文本或者CSV打开
ERR_FILE_NOTCSV	5014	文件必须以CSV格式打开
ERR_FILE_READERROR	5015	文件阅读错误
ERR_FILE_BINSTRINGSIZE	5016	字符串大小必须制定，因为文件以二进制打开
ERR_INCOMPATIBLE_FILE	5017	文本文件必须以字符串数组打开，或其他数组-二进制
ERR_FILE_IS_DIRECTORY	5018	没有文件，只有目录
ERR_FILE_NOT_EXIST	5019	文件不存在
ERR_FILE_CANNOT_REWRITE	5020	文件不能重写
ERR_WRONG_DIRECTORYNAME	5021	错误目录名称
ERR_DIRECTORY_NOT_EXIST	5022	目录不存在
ERR_FILE_ISNOT_DIRECTORY	5023	只有文件，没有目录
ERR_CANNOT_DELETE_DIRECTORY	5024	目录不能更改
ERR_CANNOT_CLEAN_DIRECTORY	5025	清除目录失败（或许一个或多个文件阻塞因而操作失败）
字符串分配		
ERR_NO_STRING_DATE	5030	字符串无日期
ERR_WRONG_STRING_DATE	5031	字符串错误日期
ERR_WRONG_STRING_TIME	5032	字符串错误时间
ERR_STRING_TIME_ERROR	5033	错误转变字符串到日期
ERR_STRING_OUT_OF_MEMORY	5034	没有足够空间建立字符串

ERR_STRING_SMALL_LEN	5035	字符串太短
ERR_STRING_TOO_BIGNUMBER	5036	太多数组，比ULONG_MAX多
ERR_WRONG_FORMATSTRING	5037	无效格式字符串
ERR_TOO_MANY_FORMATTERS	5038	格式说明符数量比参量多
ERR_TOO_MANY_PARAMETERS	5039	参量数多于格式说明符
ERR_WRONG_STRING_PARAMETER	5040	字符串类型破坏参量
ERR_STRINGPOS_OUTOFRANGE	5041	字符串外部位置
ERR_STRING_ZEROADDED	5042	字符串末尾添加0，无用操作
ERR_STRING_UNKNOWNTYPE	5043	当转换字符串时的未知数据类型
ERR_WRONG_STRING_OBJECT	5044	破坏字符串对象
数组操作		
ERR_INCOMPATIBLE_ARRAYS	5050	复制不兼容数组，字符串数组只能复制成字符串数组，和数字数组-只在数字数组中
ERR_SMALL_ASERIES_ARRAY	5051	以AS_SERIES形式接收数据，不够大
ERR_SMALL_ARRAY	5052	太小数组，启动位置在数组外
ERR_ZEROSIZE_ARRAY	5053	0长度数组
ERR_NUMBER_ARRAYS_ONLY	5054	一定是数字数组
ERR_ONEDIM_ARRAYS_ONLY	5055	一定是一维数组
ERR_SERIES_ARRAY	5056	不能使用时序列
ERR_DOUBLE_ARRAY_ONLY	5057	必须是双精度类型数组
ERR_FLOAT_ARRAY_ONLY	5058	必须是浮点型数组
ERR_LONG_ARRAY_ONLY	5059	必须是长型数组
ERR_INT_ARRAY_ONLY	5060	必须是整型数组
ERR_SHORT_ARRAY_ONLY	5061	必须是短型数组
ERR_CHAR_ARRAY_ONLY	5062	必须是图表型数据
自定义错误		
ERR_USER_ERROR_FIRST	65536	User defined 错误以该代码起始

输入和输出常量

常量：

- [文件开始标签](#)
- [内部文件位置](#)
- [代码页使用](#)
- [邮箱](#)

文件开始标签

文件开始标签指定文件访问方法，标签能如下定义：

标识符	值	描述
FILE_ READ	1	文件只读，使用 FileOpen() 标签。当打开文件规格需要使用FILE_ WRITE 和/或 FILE_ READ
FILE_ WRITE	2	文件书写开放，使用 FileOpen() 标签。当打开文件规格需要使用FILE_ WRITE 和/或 FILE_ READ
FILE_ BIN	4	二进制阅读/书写模式（不需要转换字符串），使用 FileOpen() 标签
FILE_ CSV	8	CSV文件（所有元素转换到恰当的字符串类型，统一编码或者ansi，并被分离器分离）使用 FileOpen() 标签
FILE_ TXT	16	简单文本文件（与CSV文件相同，但不考虑分离器），使用 FileOpen() 标签
FILE_ ANSI	32	ANSI类型字符串（一字节交易品种），使用 FileOpen() 标签
FILE_ UNICODE	64	UNICODE类型字符串（两字节交易品种），使用 FileOpen() 标签
FILE_ SHARE_ READ	128	多个程序共享阅读，使用 FileOpen() 标签, <div>FILE_ WRITE / FILE_ READ</div>
FILE_ SHARE_ WRITE	256	多个程序分享输入，使用 FileOpen() 标签, <div>FILE_ WRITE / FILE_ READ</div>
FILE_ REWRITE	512	使用 FileCopy() 和 FileMove() 函数可以重新输入文件，文件应该在书写时存在或者被打开，否则文件不能打开。
FILE_ COMMON	4096	所有客户端在共同文件夹中的文件路径，使用 FileIsExist() 函数中的 FileOpen() ， FileCopy() ， FileMove() 标签。

当打开一个文件时，一个或多个标签能够制定。有一个综合标签，通过逻辑OR(|) 可以连接标签，并放置在两个举例类型标签中。例如，用CSV格式打开一个文件同时阅读和输入时，会制定结合FILE_ READ|FILE_ WRITE|FILE_ CSV.

示例：

```
int filehandle=FileOpen(filename,FILE_READ|FILE_WRITE|FILE_CSV);
```

当制定阅读和书写标签时，有几种特征：

- 如果指定FILE_ READ，想要打开当前文件。如果文件不存在，文件打开失败，不能创建新文件。
- FILE_ READ|FILE_ WRITE – 如果指定文件名称不存在，创建新文件。

- FILE_ WRITE – 以0值创建文件。

当打开文件规格需要使用FILE_ WRITE 和/或 FILE_ READ。

阅读文件定义标签，控制优先。最高标签是 FILE_ CSV，然后是FILE_ BIN，FILE_ TXT 最后。因此，如果同时指定多个标签，(FILE_ TXT|FILE_ CSV 或者 FILE_ TXT|FILE_ BIN 或者 FILE_ BIN|FILE_ CSV)，会使用最高优先权标签。

编码类型定义标签优先，FILE_ UNICODE最高优先权，然后是FILE_ ANSI，因此，如果指定连接 FILE_ UNICODE|FILE_ ANSI，使用FILE_ UNICODE标签

如果FILE_ UNICODE 或者 FILE_ ANSI都没表明，暗含FILE_ UNICODE，如果 FILE_ CSV，FILE_ BIN，FILE_ TXT 都未指定，暗含FILE_ CSV。

如果以文本文件类型打开阅读 (FILE_ TXT 或者 FILE_ CSV)，文件开始有特殊的两个自己显示 0xff,0xfe，编码标签 FILE_ UNICODE，指定FILE_ ANSI。

另见

[文件函数](#)

内置文件位置

大多数的[文件函数](#)和数据读/写操作连接在一起，与此同时，使用 [FileSeek\(\)](#) 可以规定文件指针位置到内置文件中，这样将会执行下一项读或写的操作。ENUM_FILE_POSITION表达式包括有效指针位置，下一项操作时可以指定以字节转换。

ENUM_FILE_POSITION

标识符	描述
SEEK_SET	文件起点
SEEK_CUR	文件指针的当前位置
SEEK_END	文件末尾

另见

[文件结束](#)， [文件结束线](#)

在字符串转换操作使用代码页

当转换字符串变量到字符型数组中时，使用MQL5默认代码相当于ANSI窗口操作系统 (CP_ ACP) 。如果指定不同类型代码，可以使用 [CharArrayToString\(\)](#)， [StringToCharArray\(\)](#) 和 [FileOpen\(\)](#)。函数建立额外参量。

图表为最受欢迎的代码页列出了内置参量，未提及的代码页通过代码与页相一致来指定。

内置代码页常量

常量	值	描述
CP_ ACP	0	当前窗口ANSI代码页
CP_ OEMCP	1	当前系统OEM代码页
CP_ MACCP	2	当前系统Macintosh代码页。 注释：值主要在早期创建的程序代码中使用，现在没有用处了，新的Macintosh电脑使用统一码。
CP_ THREAD_ ACP	3	当前线路使用ANSI窗口代码
CP_ SYMBOL	42	交易品种代码页
CP_ UTF7	65000	UTF-7代码页
CP_ UTF8	65001	UTF-8代码页

另见

[客户端属性](#)

对话框常量

该章包括返回[MessageBox\(\)](#) 函数代码。万一已经按下ESC键或者取消按钮，如果消息窗口有取消按钮，函数返回IDCANCEL。如果没有取消按钮，按ESC也不起作用。

常量	值	描述
IDOK	1	按“确认”键
IDCANCEL	2	按“取消”键
IDABORT	3	按“停止”键
IDRETRY	4	按“重试”键
IDIGNORE	5	按“忽略”键
IDYES	6	按“是”键
IDNO	7	按“否”键
IDTRYAGAIN	10	按“再试一次”键
IDCONTINUE	11	按“继续”键

[MessageBox\(\)](#) 函数的主要标签定义常量和对话框窗口行为，它的值包括综合如下标签组：

常量	值	描述
MB_OK	0x00000000	通知窗口只包括一个按钮：确定 默认
MB_OKCANCEL	0x00000001	通知窗口包括两个按钮：确定和取消
MB_ABORTRETRYIGNORE	0x00000002	通知窗口包括三个按钮：停止,重试 和 忽略
MB_YESNOCANCEL	0x00000003	通知窗口包括三个按钮：是, 否和取消
MB_YESNO	0x00000004	通知窗口包括两个按钮：是 和 否
MB_RETRYCANCEL	0x00000005	通知窗口包括两个按钮：重试 和 取消
MB_CANCELTRYCONTINUE	0x00000006	通知窗口包括三个按钮：取消, 再试一次, 继续

在消息窗口中展示icon，有必要指定额外标签：

常量	值	描述
MB_ICONSTOP, MB_ICONERROR, MB_ICONHAND	0x00000010	STOP停止标志icon
MB_ICONQUESTION	0x00000020	icon疑问标志icon
MB_ICONEXCLAMATION,	0x00000030	惊叹/警示标志icon

MB_ICONWARNING		
MB_ICONINFORMATION, MB_ICONASTERISK	0x00000040	i包围标志

如下标签定义默认按钮：

常量	值	描述
MB_DEFBUTTON1	0x00000000	如果另外按钮MB_ DEFBUTTON2, MB_ DEFBUTTON3, 或者 MB_ DEFBUTTON4 不执行，第一按钮MB_ DEFBUTTON1 —默认。
MB_DEFBUTTON2	0x00000100	默认第二按钮
MB_DEFBUTTON3	0x00000200	默认第三按钮
MB_DEFBUTTON4	0x00000300	默认第四按钮

MQL5 Programs

操作MQL5程序，必须要编译（编译按钮或者F7键），要进行无错误编辑（可以传递警告信息，可供分析），这一过程，同样名称和EX5延期的可执行文件会增加到目录中，terminal_dir\MQL5\Experts, terminal_dir\MQL5\indicators 或者 terminal_dir\MQL5\scripts.这些文件可以执行。

MQL5程序的操作特点可如下形容：

- [程序运行](#) – 调用预先定义的事件处理器的命令；
- [客户端事件](#) – 事件描述，在程序中可以处理；
- [调用输入函数](#) – 描述命令，允许参量，搜索细节并为输入函数同意调用；
- [运行错误](#) – 获得关于运行时间和关键性误差的信息。

EA交易，自定义指标和脚本在导航窗口中附属于公开图表Drag'n'Drop操作

对于停止操作的专家系统来说，通过图表快捷菜单选择Expert Advisors-Remove来删除，EA交易的操作也会受到Enable/disable Expert Advisors按钮影响。

为了停止自定义指标，需要从图表中移除。

直到从图表中移除才能运行自定义指标和EA交易；附加在EA交易和指标中的信息在客户端起始位置保存。

在操作完成时脚本执行一次后就自动删除或改变当前图表信息，否则客户端会停止工作，在重启客户端脚本后并不能开始，因为他们的信息并未保存。

EA交易的一个最大化，一个脚本和在一个图表中指标无限的操作数量。

程序运行

程序附属于图表之后，上传客户端内存并进行全局变量[初始化](#)，如果分类中的某些全局变量有[构造函数](#)，在[全局变量](#)初始化时调用构造函数。

在程序等待客户端[事件](#)之后，每个MQL5程序应该至少有一个[事件处理器](#)，否则就不会执行下载程序，事件处理程序有预定义名称，参量和返回类型。

类型	函数名称	参量	应用	注释
int	OnInit	无	EA交易和指标	初始化 事件处理器，允许使用缺省返回类型
void	OnDeinit	const int reason	EA交易和指标	无法初始化 事件处理器
void	OnStart	无	脚本	启动 事件处理器
int	OnCalculate	const int rates__total, const int prev__calculated, const datetime &Time[], const double &Open[], const double &High[], const double &Low[], const double &Close[], const long &TickVolume [], const long &Volume[], const int &Spread[]	指标	所有价格计算处理器
int	OnCalculate	const int rates__total, const int prev__calculated, const int begin, const double &price[]	指标	计算 单个数据数组事件处理器 指标不能同时有两个事件处理器 只有一个事件处理器处理数据数组
void	OnTick	无	EA交易	新订单 事件处理器，处理新接收的文件，该类型没有收到其他文件
void	OnTimer	无	EA交易和指标	定时器 事件处理器
void	OnTrade	无	EA交易	交易 事件处理器
double	OnTester	无	EA交易	测试 事件处理器
void	OnChartEvent	const int id, const long &lparam, const double &dparam, const string &sparam	EA交易和指标	图表事件 事件处理器

void	OnBookEvent	const string &symbol_name	EA交易	预定事件 事件处理器
------	-----------------------------	------------------------------	------	----------------------------

客户端在事件队列中添加版面事件，因此会一连串地处理接收到的类似命令文件，对于NewTick事件有一个例外，如果队列中已经有一个事件或者该事件已被处理过，新的NewTick事件就不能入列。

队列中的事件大小是受约束的，一旦队列溢出，为了接收新事件，未被处理的旧事件就会被删除。因此，推荐使用有效事件处理器，不推荐使用无线循环（脚本是例外，只处理开始事件）

[函数库](#) 不处理任何事件。

加载卸载指标

如下步骤装载指标：

- 指标附属于图表；
- 客户端开始（如果指标粘贴在图表之前，那是用来阻止客户端的）；
- 装载样本（在样本中，指标添加到图表指定位置）；
- 剖面转变（指标粘贴在其中一个剖面图里）；
- 图表中的交易品种或者时间表的转变，需要附加指标；
- 一个指标重新编译后，指标粘贴在一个图表上；
- 改变指标的 [输入参量](#)。

如下步骤卸载指标：

- 从图表中分类指标；
- 终端关机（指标附属在图表上）；
- 如果指标附加在图表上，下载一个模板；
- 关闭一个图表，指标是附加的；
- 改变一个剖面，改变指标附加的图表中的一个剖面；
- 图表中交易品种或者时间表改变，指标附加；
- 改变指标中的 [输入参量](#)。

EA交易加载卸载

如下步骤下载EA交易：

- 图表上添加EA交易；
- 程序启动（EA交易添加到图表之前，关闭客户端）；
- 下载一个样本（在样本中添加到图表的EA交易是指定的）；
- 改变剖面（EA交易添加到图表中一项剖面图中）。
- (
-)

如下步骤卸载EA交易：

- 从图表中分离EA交易；
- 如果新的EA交易添加到图表中，另一个EA交易已经附加，EA交易就被卸载；
- 关掉客户端（EA交易添加到图表中）；
- 装载模板，EA交易添加到图表中；
- 关闭添加EA交易的图表；
- 剖面转变，EA交易添加到图表中一个剖面图中。

（
）。

添加EA交易的图表中的交易品种和时间表改变了，EA加以就不能加载或者卸载。一旦客户端随后以新的交易品种/时间表调用在旧的交易品种/时间表的 [OnDeinit\(\)](#) 和 [OnInit\(\)](#) 处理器，全局变量的值和 [静态变量](#) 不能重新设置。EA交易中接收的所有事件，在初始化之前就完整([OnInit\(\)](#) function) 跳过。

脚本加载卸载

在添加到图表过后脚本就被迅速装载，在完成操作之后会迅速被卸载，[OnInit\(\)](#) 和 [OnDeinit\(\)](#) 用来调用脚本文件。

当程序卸载后（从图表中删除），[全局](#)变量的客户端程序不能初始化并删除队列中的事件，这样所有的[字符](#)类型常量都会重设，[动态数组对象](#) 的存储单元会分配[析构函数](#) 也会调用。

为了更好地理解EA交易的操作，我们推荐如下的EA交易编译代码，并执行装载/卸载，模板转变，样本转变，事件表转变等等：

示例：

```
//+-----+
//|                                     TestExpert.mq5 |
//|                                     Copyright 2009, MetaQuotes Software Corp. |
//|                                     http://www.mql5.com |
//+-----+

#property copyright "2009, MetaQuotes Software Corp."
#property link      "http://www.mql5.com"
#property version   "1.00"

class CTestClass
{
public:
    CTestClass() { Print("CTestClass constructor"); }
    ~CTestClass() { Print("CTestClass destructor"); }
};

CTestClass global;

//+-----+
//| 专家初始化函数 |
//+-----+

int OnInit()
{
```

```
//---
    Print("Initialisation");
//---
    return(0);
}
//+-----+
//|  专家无法初始化函数          |
//+-----+
void OnDeinit(const int reason)
{
//---
    Print("Deinitialisation with reason",reason);
}
//+-----+
//|  专家订单号函数              |
//+-----+
void OnTick()
{
//---

}
//+-----+
```

另见

[客户端事件](#) , [事件处理程序](#)

客户端事件

初始化

在客户端下载程序后（EA交易或者自定义指标）就开始全局变量初始化过程，发送初始化事件，通过[OnInit\(\)](#) 事件处理器处理过程，当安全性和/或时间表改变时，该程序在MetaEditor编辑器中编译之后，事件生成，EA交易或者自定义指标设置窗口中输入参量改变。在账户改变之后，EA交易也可以初始化，初始化事件不只是为脚本创建的。

无法初始化

在全局变量初始化和程序（EA交易和自定义指标）加载之前，客户端向程序发送[Deinit](#)事件。当客户端关闭时，Deinit就会生成，当图表关闭时，安全性和/或时间表改变，成功的程序重编辑，当输入参量改变，账户改变。

[无法初始化](#) 可以从参量中获得，通过 [OnDeinit\(\)](#) 函数传递，OnDeinit() 函数的运行仅限于2.5秒，如果在此期间，函数没完成，会强制终止，Deinit事件不只为脚本创建的。

启动

[启动](#) 事件在其装载之后对于脚本激活来说是特殊事件，该事件通过 [OnStart](#) 处理器进行处理，开始事件不发送给EA交易或者自定义服务器。

NewTick

如果有新的引用，[NewTick](#)事件会生成，通过EA交易的附加[OnTick\(\)](#) 过程。当先前引用的OnTick函数正在处理的时候，新的引用就会接收，新的引用会被EA交易忽视，因为类似事件不会列入队伍里。

当程序运行时，接收到的所有新的引用都会忽视，直到OnTick() 完成。只有在接收新引用后该函数才会运行，无论自动交易是否允许NewTick事件都会产生（"Allow/prohibit Auto trading"按钮），自动交易引用禁止，EA交易发送的交易请求是不允许的，而EA交易仍旧进行。

自动交易禁令通过按适当按钮会停止当前执行的OnTick() 函数，当EA交易属性开始时，OnTick() 并不能启动。当EA交易运行时，属性窗口不能打开。

计算

[计算](#)事件只在指标完成初始化事件并改变价格数据后开始生成，通过 [OnCalculate](#) 函数执行。

定时器

[定时器](#) 事件通过EA交易客户端定期执行，并通过[EventSetTimer](#) 函数被定时器激活。通常情况下，该函数通过OnInit调用。定时器事件过程通过[OnTimer](#)函数执行。在EA交易操作完成之后，有必要通过[EventKillTimer](#)函数删除定时器，通常叫做OnDeinit函数。

交易

当交易服务器完成一个交易操作，交易事件生成，交易事件通过 [OnTrade\(\)](#) 函数处理，有如下操作步骤：

- 发送、修改或者删除待办订单；
- 因为金额不满或者延期而取消待办订单；
- 激活待办订单；

- 打开，增加或者关闭位置（或者部分位置）；
- 修改打开位置（改变站点-止损数值和/或目标数值）。

测试

测试事件在EA交易测试历史数据完毕后产生，该事件通过 [OnTester\(\)](#) 函数处理。

ChartEvent

当用户使用图表时，客户端产生事件，即ChartEvent ..

- 当图表窗口聚集时，按关键字按钮；
- 创建[图解对象](#) ..
- 删除[图解对象](#) ..
- 鼠标点击图表的图解对象；
- 使用鼠标移动图解对象；
- 在标签编辑中进行文本末尾编辑；

自定义事件ChartEvent，可以通过使用[EventChartCustom](#)函数以任何MQL5程序发送到EA交易中，该事件通过函数 [OnChartEvent](#) 处理。

BookEvent

BookEvent 事件在市场深度改变后通过客户端产生，通过 [OnBookEvent](#)函数执行，开始BookEvent程序特定交易对象，有必要通过 [MarketBookAdd](#) 函数为该事件订阅交易对象。

为从BookEvent中取消特定的交易品种，调用 [MarketBookRelease](#) 函数是有必要的。BookEvent事件是广播类型事件-它表示足以为一个事件订阅一个EA交易，而其他所有EA交易都有OnBookEvent事件，都会接收它。有必要分析交易品种的名称，以一个参量传递处理器。

另见

[事件处理程序](#)， [程序运行](#)

MQL5

MQL5

:

- [PlaySound\(\)](#) ;
- [ObjectCreate\(\)](#) _____

OBJ_BITMAP OBJ_BITMAP_LABEL

PlaySound()

[PlaySound\(\)](#):

```
//+-----+
//|  функция вызывает штатную OrderSend() и проигрывает звук  |
//+-----+
void OrderSendWithAudio(MqlTradeRequest &request, MqlTradeResult &result)
{
    //-- отправим запрос на сервер
    OrderSend(request,result);
    //-- если запрос принят, играем звук Ok.wav
    if(result.retcode==TRADE_RETCODE_PLACED) PlaySound("Ok.wav");
    //-- при неудаче выдаем тревожный звук из файла timeout.wav
    else PlaySound("timeout.wav");
}
```

Ok.wav timeoit.wav,

— \Sounds. —

MetaTrader 5.

mql5-

:

```
//--- Папка, в которой хранятся данные терминала
string terminal_path=TerminalInfoString(TERMINAL_PATH);
```

— \Sounds, —

— \MQL5.

" "-"

"

:

```
//--- Папка, в которой хранятся данные терминала
string terminal_data_path=TerminalInfoString(TERMINAL_DATA_PATH);
```

```

,
_      _      Demo.wav
_      _      \MQL5\Files,      PlaySound( )
:

```

```

//--- проиграем звуковой файл Demo.wav из папки каталог_данных_терминала\MQL5\Files\D
PlaySound("\\Files\\Demo.wav");

```

```

,
"\"
"\".

```

ObjectCreate()

```

,
"      ObjectCreate( )
" ( OBJ_BITMAP_LABEL) .

```

```

string label_name="currency_label";      // имя объекта OBJ_BITMAP_LABEL
string euro      ="\\Images\\euro.bmp";  // путь к файлу каталог_данных_терминала\M
string dollar    ="\\Images\\dollar.bmp"; // путь к файлу каталог_данных_терминала\M
//+-----+
//| Expert initialization function |
//+-----+
int OnInit()
{
//--- создадим кнопку OBJ_BITMAP_LABEL, если ее еще нет
if(ObjectFind(0,label_name)<0)
{
//--- попробуем создать объект OBJ_BITMAP_LABEL
bool created=ObjectCreate(0,label_name,OBJ_BITMAP_LABEL,0,0,0);
if(created)
{
//--- привяжем кнопку к правому верхнему углу графика
ObjectSetInteger(0,label_name,OBJPROP_CORNER,CORNER_RIGHT_UPPER);
//--- теперь настроим свойства объекта
ObjectSetInteger(0,label_name,OBJPROP_XDISTANCE,100);
ObjectSetInteger(0,label_name,OBJPROP_YDISTANCE,50);
//--- сбросим код последней ошибки в 0
ResetLastError();
//--- загрузим картинку для состояния кнопки "Нажата"
bool set=ObjectSetString(0,label_name,OBJPROP_BMPFILE,0,euro);
//--- проверим результат
if(!set)

```

```

    {
        PrintFormat("Не удалось загрузить картинку из файла %s. Код ошибки %d", eu
    }
    ResetLastError();
    //--- загрузим картинку для состояния кнопки "Отжата"
    set=ObjectSetString(0,label_name,OBJPROP_BMPFILE,1,dollar);

    if(!set)
    {
        PrintFormat("Не удалось загрузить картинку из файла %s. Код ошибки %d",do
    }
    //--- отдадим графику команду на обновление, чтобы кнопка появилась сразу же
    ChartRedraw(0);
}
else
{
    //--- объект создать не удалось, сообщим об этом
    PrintFormat("Не удалось создать объект OBJ_BITMAP_LABEL. Код ошибки %d",GetL
}
}
//---
return(0);
}
//+-----+
//| Expert deinitialization function |
//+-----+
void OnDeinit(const int reason)
{
    //--- удалим объект с графика
    ObjectDelete(0,label_name);
}

```

currency_label

OnInit() .

euro

dollar,

:

```

string euro      ="\\Images\\euro.bmp";    // путь к файлу каталог_данных_терминала\M
string dollar    ="\\Images\\dollar.bmp";  // путь к файлу каталог_данных_терминала\M

```

\MQL5\Images.

OBJ_BITMAP_LABEL

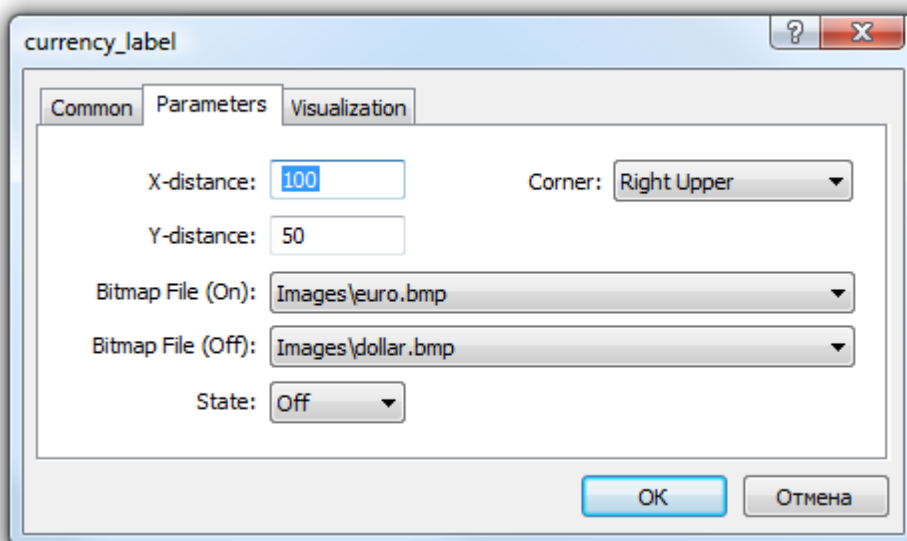
,

(

)

: euro.bmp

dollar.bmp.



```

OBJ__BITMAP__LABEL (
    "
OBJ__BITMAP

```

```

OBJPROP_BMPFILE,
OBJ__BITMAP OBJ__BITMAP__LABEL,

```

mq15-

mq15-

mq15-

```

MQL5,
#resource:

```

```

#resource путь_к_файлу_ресурса

```

```

#resource

```

EX5.

EX5-

Genetic map of the MQL5 gene region on chromosome 16. The map shows the MQL5 gene structure with exons (EX5, EX5-, EX5, EX5-) and introns (BMP, WAV). The mql5- mutation is indicated by a red arrow. A scale bar indicates 16 Mb.

The image displays a diagram related to XML resource file paths. At the top center, there is a yellow rectangular box containing the text "#resource "<". Below this box, the code snippet "<resource "<путь_к_файлу_ресурса">" is shown. The rest of the image features several scattered symbols and fragments of code, including "<", ">", "?", ":", "\\\"", "\\\\", "\\MQL5\\", "\\\\", "\\..\\", and ":".

```
//--- правильное указание ресурсов
#resource "\\Images\\euro.bmp" // euro.bmp находится в каталог_данных_терминала\MQL5\
#resource "picture.bmp" // picture.bmp находится в том же каталоге, где и исхо
#resource "Resource\\map.bmp" // ресурс находится в папке каталог_исходного_файла\Re

//--- неправильное указание ресурсов
#resource ":picture_2.bmp" // нельзя использовать ":"
#resource "..\\picture_3.bmp" // нельзя использовать ".."
#resource "\\Files\\Images\\Folder First\\My panel\\Labels\\too long path.bmp" //боль
```

#resource,

.

,

":."

:

```
//--- примеры указания ресурсов и их имена в комментарии
#resource "\\Images\\euro.bmp"           // имя ресурса - Images\\euro.bmp
#resource "picture.bmp"                  // имя ресурса - picture.bmp
#resource "Resource\\map.bmp"           // имя ресурса - Resource\\map.bmp
#resource "\\Files\\Pictures\\good.bmp" // имя ресурса - Files\\Pictures\\good.bmp
#resource "\\Files\\Demo.wav";           // имя ресурса - Files\\Demo.wav"
#resource "\\Sounds\\thrill.wav";        // имя ресурса - Sounds\\thrill.wav"
...

//--- использование ресурсов
ObjectSetString(0,bitmap_name,OBJPROP_BMPFILE,0,"::Images\\euro.bmp");
...
ObjectSetString(0,my_bitmap,OBJPROP_BMPFILE,0,"::picture.bmp");
...
set=ObjectSetString(0,bitmap_label,OBJPROP_BMPFILE,1,"::Files\\Pictures\\good.bmp");
...
PlaySound("::Files\\Demo.wav");
...
PlaySound("::Sounds\\thrill.wav");
```

OBJ_BITMAP OBJ_BITMAP_LABEL

OBJPROP_BMPFILE

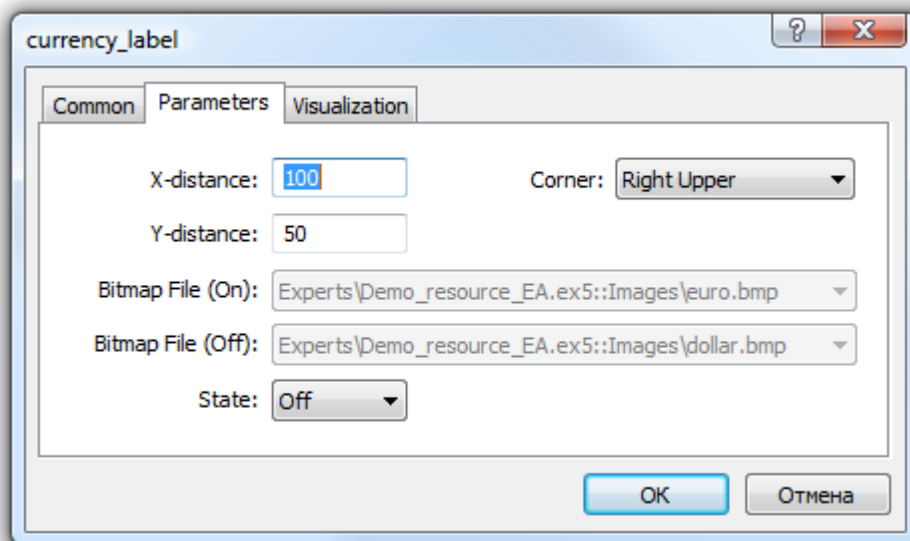
OBJ_BITMAP_LABEL

euro.bmp dollar.bmp.

```
#resource "\\Images\\euro.bmp"; // euro.bmp находится в каталог_данных_терминала\M
#resource "\\Images\\dollar.bmp"; // dollar.bmp находится в каталог_данных_терминала
```

BitMap File (On) BitMap File (Off)

:



mqI5-

mqI5-

EX5.

EX5

mqI5-

< — — — — — EX5>::
 Draw_Triangles_Script.mq5
 triangle.bmp:

```
#resource "\\Files\\triangle.bmp"
```

"Files\triangle.bmp",

":":

```
//--- использование ресурса в самом скрипте  
ObjectSetString(0,my_bitmap_name,OBJPROP_BMPFILE,0,"::Files\\triangle.bmp");
```

EX5-

\MQL5\

EX5-

- Draw_Triangles_Script.ex5.

\MQL5\

Scripts\,

```
//--- использование ресурса скрипта в эксперте  
ObjectSetString(0,my_bitmap_name,OBJPROP_BMPFILE,0,"\\Scripts\\Draw_Triangles_Script.
```


EX5-

Draw_Triangles_Script.ex5

```
//--- запрос ресурса скрипта в эксперте без указания пути
```

```
ObjectSetString(0,my_bitmap_name,OBJPROP_BMPFILE,0,"Draw_Triangles_Script.ex5::Files\
```

```
\MQL5\Experts\,
```

```
\MQL5\Experts\.
```

[PlaySound\(\)](#), [ObjectSetInteger\(\)](#), [ChartApplyTemplate\(\)](#), [文件函数](#)

调用输入函数

在执行MQL5程序的过程中，客户端较早使用输入函数订阅。这意味着，如果服务器调用输入函数，在程序加载过程中，类似模块(ex5 or dll) 就会装载，MQL5和DLL数据库通过调用模块执行。

不推荐使用模块的全部定义名称来加载，比如Drive:\Directory\FileName.Ext。MQL5数据库从terminal_dir\MQL5\Libraries文件夹中加载，如果数据库没有找到，客户端会尝试加载到terminal_dir\experts文件夹中。

系统数据库（DLL）通过操作系统规则加载，如果数据库已经加载完毕（例如，另外的EA交易，甚至是其他的客户端，并联在其中运行），然后到已经建立好的数据库中使用要求，否则，它会按如下顺序搜索：

1. 目录，从模块输入点dll开始, 该模块是EA交易，脚本，指标或者EX5数据库；
2. 目录 terminal_ _ data_ directory\MQL5\Libraries ([TERMINAL_DATA_PATH](#)\MQL5\Libraries) ；
3. 目录，MT5客户端开始启动；
4. 系统目录；
5. 窗口目录；
6. 当前目录；
7. PATH系统变量的目录列表。

如果DLL数据库使用另外的DLL工作，当没有第二个DLL时，第一个不能下载。

在EA交易（脚本，指标）下载后，所有EX5数据库模块的普通列表建立。通过使用加载的EA交易（脚本，指标）和列表数据库，之前加载过多次使用的EX5数据库模块是需要的，数据库使用EA交易（脚本，指标）中的 [预定义变量](#) 来调用。

输入数据库EX5在如下列表中搜寻：

1. 目录，与建立的EA交易（脚本，指标）路径相关，输入EX5；
2. 目录terminal_ directory\MQL5\Libraries ；
3. 目录MQL5\Libraries在所有MT5客户端的共同目录中 (Common\MQL5\Libraries) 。

函数 [imported](#) DLL在MQL5程序中一定要确定API窗口允许使用，为了确保同意，文本程序添加到C或者C++，使用关键字 `__stdcall`，针对 Microsoft(r) 编译器，该协议具有如下特性：

- 访客（在我们系统中是MQL5程序）应该注意到函数调用的原型（从DLL中输入），为了适当结合参量堆积；
- 访客（在我们系统中是MQL5程序）以倒序方式将参量进行堆积，从右至左-在此命令中，输入函数从参量中传递；
- 参量通过值传递，除了通过引用明确传递（在我们这一排）；
- 通过阅读参量传递，输入函数独立清除堆积。

当描述输入函数的原型，可以使用默认参量。

如果类似数据库不能加载，或者DLL禁止使用，或者输入函数没找到-EA交易在分目录（子文件）发出消息“EA交易停止”来停止操作。EA交易在重启后会自动运行，在属性表格打开或者按下确认键之后，EA交易重启后重新编译。

传递参量

所有 [简单型](#) 参量通过值传递，除非明确表明通过引用传递，当[字符串](#)传递时，缓冲区的地址复制字符串传递，如

果字符串通过引用传递，字符串缓冲区的地址不用复制就能从DLL传递输入函数。

[结构](#)包括动态数组，字符串，分类和其他复杂结构，和枚举对象的静态或者[动态数组](#)，不能通过输入函数传递参量。

当传递数组到DLL，数据缓冲区开头的地址总是能传递（不考虑[AS__SERIES](#)标签）。在DLL中的函数与AS__SERIES标签无关，传递数组是不明确长度的静态数组，规定数组大小应使用额外参量。

运行时间出错

客户端执行子系统在MQL5程序运行中任何情况下都有机会存储[错误代码](#)，对于每个可执行的MQL5程序来说都有预定义变量 `_LastError`。

在开始 `OnInit` 函数时，重置 `_LastError` 变量，在计算或者内部函数的调用过程中，任何错误情况都会发生，`_LastError` 变量接收类似错误代码，存储在变量中的值可以通过 `GetLastError()` 函数获得。

有几个关键性误差，迅速出现在程序终端：

- 除以0；
- 超出数组范围；
- 使用错误 [对象指针](#)。

预定义变量

对于每个可执行的MQL5程序的一套变量都是支持的，可以通过MQL5程序反应当前即时价格（EA交易、脚本或者自定义指标）。

预先定义变量的值在MQL5程序启动之前通过客户端建立，预先定义变量是不变的，也不会通过MQL5程序转变。例外是，特殊变量 `_LastError`，通过 [ResetLastError](#) 函数重设至0

变量	值
_Digits	小数位数字
_Point	在引用货币中当前交易品种的大小
_LastError	最后的错误代码
_Period	当前表格的时间表
_StopFlag	停止标记程序
_Symbol	当前图表的交易品种名称
_UninitReason	阻止初始化原因代码

数据库被程序变量调用使用。

int __Digits

__Digits变量保留小数点后几位，定义当前图表交易品种的价格精确度

可以使用 [Digits\(\)](#) 函数。

double __Point

__Point变量包括货币报价中当前交易品种的大小点

可以使用 [Point\(\)](#) 函数。

int __LastError

__LastError 变量包括最后 [错误](#) 代码，在MQL5程序运行中发生，使用 [ResetLastError\(\)](#) 值能重设成0。

获得最后错误代码，使用 [GetLastError\(\)](#) 函数。

int __Period

__Period变量包括当前图表时间表的值。

也可以使用 [Period\(\)](#) 函数。

另见

[PeriodSeconds](#) , [图表时间表](#) , [日期和时间](#) , [对象可见性](#)

bool __StopFlag

__StopFlag变量包括MQL5程序停止标签，当客户端想要停止程序，建立__StopFlag 变量到真值。

为了检测__StopFlag，你可以使用 [IsStopped](#) 函数。

string __Symbol

__Symbol变量包括当前图表的交易品种名称。

也可以使用 [Symbol\(\)](#) 函数。

int __UninitReason

__UninitReason变量包括 [无法初始化](#) 程序中的代码。

通常，[UninitializeReason\(\)](#) 函数中包括代码。

普通函数

专业集合里不包括的一般用途函数如下表。

函数	命令
Alert	在独立窗口中显示消息
CheckPointer	返回目标指针类型
Comment	图表左上角输出注解文本
DebugBreak	排除故障中的程序断点
ExpertRemove	停止智能交易并从图表中卸载
GetPointer	返回对象 指针
GetTickCount	从系统开始，返回已经过去的毫秒的数量
MessageBox	创建，显示消息盒子，然后管理
PeriodSeconds	在周期中返回秒钟数量
PlaySound	播放第二个文件
Print	显示未经处理的消息
PrintFormat	在目标文件中按照原来格式，格式刷建立的交易品种和值
ResetLastError	建立预定义变量 _LastError 的值到0
SendFTP	在"Publisher" 标签窗口中设置发送文件地址
SendMail	在"Mailbox" 标签窗口中发送邮件到指定地址
Sleep	延迟执行当前EA交易或者脚本的规定间隔
TerminalClose	命令客户端完成操作
TesterStatistics	
ZeroMemory	通过引用重设变量传递，该变量可以是任何类型，除了分类和结构都有构造函数

Alert

在单独窗口中显示信息。

```
void Alert(  
    argument,      // 第一值  
    ...           // 其他值  
);
```

参量

argument

[in] 任意值，被逗号分开，分开信息输出分类量，可以使用线路供应字符"\n" 或者 "\r\n". 参量的数量不能超过64个

返回值

没有返回值。

注释

数组不能通过Alert() 函数传递，数组应该输出元素，双精度类型数据可以输出小数点后8位以上，浮点类型数据可以输出小数点后5位，不同精度起源不同数字或者系统模式，使用 [DoubleToString\(\)](#)。

布尔型数据以true或者false字符串输出，日期以 YYYY.MM.DD HH:MI:SS格式，使用 [TimeToString\(\)](#) 日期另外模式显示日期，颜色类型数据既可以RGB字符串或者颜色名称输出，颜色存在颜色设置中。

CheckPointer

函数返回 [指针](#)对象类型

```
ENUM_POINTER_TYPE CheckPointer(
    object* anyobject    // 对象指针
);
```

参量

anyobject
[in] 对象指针。

返回值

从 [ENUM_POINTER_TYPE](#) 枚举中返回值。

注释

想要调用不正确的指针到[危险终止](#)结果程序中，在使用指针之前，调用CheckPointer函数是有必要的，如下条件下，指针是不正确的：

- 指针等于 [NULL](#)；
- 使用 [delete](#) 操作删除对象；

该函数可以用来正确检测指针，非零值警告指针在访问时使用。

示例：

```
//+-----+
//|   通过删除元素来删除列表   |
//+-----+
void CMyList::Destroy()
{
    //--- 循环工作服务指针
    CItem* item;
    //--- 循环检测并删除动态指针
    while(CheckPointer(m_items)!=POINTER_INVALID)
    {
        item=m_items;
        m_items=m_items.Next();
        if(CheckPointer(item)==POINTER_DYNAMIC)
        {
            Print("Dynamyc object ",item.Identifier()," to be deleted");
            delete (item);
        }
        else Print("Non-dynamic object ",item.Identifier()," cannot be deleted");
    }
    //---
}
```

另见

[对象指针](#) - [检测对象指针](#) , [对象删除操作符的删除](#)

Comment

图表左上角函数通过用户输出注解。

```
void Comment(  
    argument,      // 第一值  
    ...           // 下一值  
);
```

参量

...

[in]任何值，逗号分隔。划线输出信息到几条路线，使用一条换行符 "\n" 或者 "\r\n"，参量数不能超过64个。输入注解文本（包括隐形交易品种）的总长度不能超过2045个字节（在输出过程中超出的交易品种会被删去）

返回值

没有返回值

注释

数组不能通过Comment() 函数传递，数组必须一个接一个元素进入。

双精度型数组的精确度保留小数点后16位，以传统或者科学的方式输出，以保证接入密切。浮点型数据保留小数点后5位，为了以另一精确度或者预定义模式输出数字，使用 [DoubleToString\(\)](#) 函数。

布尔型数据以true或者false输出，数据显示成 YYYY.MM.DD HH:MI:SS.显示另一模式数据，使用 [TimeToString\(\)](#)。颜色型数据以RGB或者颜色名称显示，如果该颜色在颜色设置中存在。

示例：

```
void OnTick()  
{  
    //---  
    double Ask,Bid;  
    int Spread;  
    Ask=SymbolInfoDouble(Symbol(),SYMBOL_ASK);  
    Bid=SymbolInfoDouble(Symbol(),SYMBOL_BID);  
    Spread=SymbolInfoInteger(Symbol(),SYMBOL_SPREAD);  
    //--- 3行输出值  
    Comment(StringFormat("Show prices\nAsk = %G\nBid = %G\nSpread = %d",Ask,Bid,Spread)  
}
```

另见

[ChartSetString](#) , [ChartGetString](#)

DebugBreak

在排除故障中的程序断点。

```
void DebugBreak();
```

返回值

没有返回值。

注释

如果程序以调试的方式打开，MQL5程序的执行被打断。函数可以使用变量的观察值和/或一步步执行。

ExpertRemove

函数停止 [EA交易](#) 并从图表中卸载。

```
void ExpertRemove();
```

返回值

没有返回值。

注释

一旦调用ExpertRemove() 函数，EA交易不会立即停止，设立一个停止EA操作的标志，任何事件都不能处理，调用 [OnDeinit\(\)](#) ，然后EA交易就会卸载并且从图表中删除。

示例：

```
//+-----+
//|                                     Test_ExpertRemove.mq5 |
//|                                     Copyright 2009, MetaQuotes Software Corp. |
//|                                     http://www.mql5.com |
//+-----+
#property copyright "2009, MetaQuotes Software Corp."
#property link      "http://www.mql5.com"
#property version   "1.00"
input int ticks_to_close=20; // EA 卸载前的订单号
//+-----+
//| 专家无法初始化函数                                |
//+-----+
void OnDeinit(const int reason)
{
//---
    Print(TimeCurrent(),": ",__FUNCTION__," reason code = ",reason);
//--- "清除" 注释
    Comment("");
//---
}
//+-----+
//| 专家订单号函数                                |
//+-----+
void OnTick()
{
    static int tick_counter=0;
//---
    tick_counter++;
    Comment("\nBefore unloading expert advisor ",__FILE__," left",
        (ticks_to_close-tick_counter)," ticks");
//--- 以前
    if(tick_counter>=ticks_to_close)
    {
        ExpertRemove();
    }
}
```

```
Print(TimeCurrent(),": ",__FUNCTION__," expert advisor will be unloaded");  
}  
Print("tick_counter =",tick_counter);  
//---  
}  
//+-----+
```

另见

[程序运行](#)， [客户端事件](#)

GetPointer

函数返回 [指针](#) 对象。

```
void* GetPointer (
    any_class anyobject    // 任何类的对象
);
```

参量

anyobject
[in] 任一类的对象。

返回值

函数返回对象指针

注释

只有分类对象有指标，例如[架构](#)和简单类型变量没有指标。分类对象不能使用new() 操作建立，但是，例如，数组对象会自动建立，直到产生指标。但是该指标会是自动类型POINTER_ AUTOMATIC，因此[delete\(\)](#) 操作不能应用，该类型指标与动态 [POINTER_ AUTOMATIC](#) 类型指标不同。

结构类型变量和简单类型没有指标，也禁止应用GetPointer() 函数。也禁止以函数论据传递指标，在此情况下，编译器会通知出现一个错误。

想要调用引起程序 [危险终止](#) 的错误指标，这就是使用指标之前调用 [CheckPointer\(\)](#) 函数的原因，如下情况指标错误：

- 指标等于 [NULL](#) ；
- 使用 [delete](#) 操作对象删除；

该函数可用来有效检验指标，保证非零值，该指标可用于访问。

示例：

```
//+-----+
//|                                     Check_GetPointer.mq5 |
//|                                     Copyright 2009, MetaQuotes Software Corp. |
//|                                     http://www.mql5.com |
//+-----+
#property copyright "2009, MetaQuotes Software Corp."
#property link      "http://www.mql5.com"
#property version   "1.00"

//+-----+
//|  执行列表元素的类                      |
//+-----+
class CItem
{
    int          m_id;
    string       m_comment;
    CItem*       m_next;
```

```

public:
    CItem() { m_id=0; m_comment=NULL; m_next=NULL; }
    ~CItem() { Print("Destructor of ",m_id,
                    (CheckPointer(GetPointer(this))==POINTER_DYNAMIC
                     "dynamic":"non-dynamic")); }

    void Initialize(int id,string comm) { m_id=id; m_comment=comm; }
    void PrintMe() { Print(__FUNCTION__,":",m_id,m_comment); }
    int Identifier() { return(m_id); }
    CItem* Next() {return(m_next); }
    void Next(CItem *item) { m_next=item; }
};

//+-----+
//| 列表中最简单的类 |
//+-----+
class CMyList
{
    CItem* m_items;
public:
    CMyList() { m_items=NULL; }
    ~CMyList() { Destroy(); }

    bool InsertToBegin(CItem* item);
    void Destroy();
};

//+-----+
//| 从开始嵌入列表元素 |
//+-----+
bool CMyList::InsertToBegin(CItem* item)
{
    if(CheckPointer(item)==POINTER_INVALID) return(false);
//---
    item.Next(m_items);
    m_items=item;
//---
    return(true);
}

//+-----+
//| 通过删除元素删除列表 |
//+-----+
void CMyList::Destroy()
{
//--- 循环工作服务指针
    CItem* item;
//--- 循环检测并删除动态指针
    while(CheckPointer(m_items)!=POINTER_INVALID)
    {
        item=m_items;
        m_items=m_items.Next();
        if(CheckPointer(item)==POINTER_DYNAMIC)

```

```

        {
            Print("Dynamyc object ",item.Identifier()," to be deleted");
            delete (item);
        }
        else Print("Non-dynamic object ",item.Identifier()," cannot be deleted");
    }
//---
}
//+-----+
//| 脚本程序启动函数 |
//+-----+
void OnStart()
{
    CMyList list;
    CItem items[10];
    CItem* item;
//--- 创建并加入一个动态对象指针到列表
    item=new CItem;
    if(item!=NULL)
    {
        item.Initialize(100,"dynamic");
        item.PrintMe();
        list.InsertToBegin(item);
    }
//--- 添加自动指针到列表
    for(int i=0; i<10; i++)
    {
        items[i].Initialize(i,"automatic");
        items[i].PrintMe();
        item=GetPointer(items[i]);
        if(CheckPointer(item)!=POINTER_INVALID)
            list.InsertToBegin(item);
    }
//--- 在列表开始添加一个以上动态对象指针
    item=new CItem;
    if(item!=NULL)
    {
        item.Initialize(200,"dynamic");
        item.PrintMe();
        list.InsertToBegin(item);
    }
//--- 删除所有列表元素
    list.Destroy();
//--- 脚本结束后所有列表元素都被删除
//--- 末端看到专家标签
}

```

另见

[对象指针](#)，[检测对象指针](#) [~对象删除操作符的删除](#)

GetTickCount

自从程序开始，GetTickCount() 函数返回已过去的毫秒的数量。

```
uint GetTickCount();
```

返回值

无符号整型值

注释

系统定时器限制计数器，时间以一种无符号的整数来存储，因此，如果电脑不间断的工作，每49.7就会溢满。

MessageBox

建立并显示对话框并管理它，一个对话框包括信息和页眉，任意预定义标签和命令按钮的组合。

```
int MessageBox(  
    string text,           // 信息文本  
    string caption=NULL,  // 对话框表头  
    int flags=0           // 对话框按钮设置  
);
```

参量

text

[in] 文本，包括信息输出。

caption=NULL

[in] 可选文本出现在方框中，如果参量是空的，EA交易名称在方框中显示。

flags=0

[in] 可选项 [标记](#) 规定外观和信息框的行为，标签可与特殊标签组结合。

返回值

如果函数成功执行，返回值是 [MessageBox\(\)](#) 返回代码中值的一个。

注释

函数不可以从自定义指标中调用，因为指标显示接口线路并不能减速。

PeriodSeconds

在一个周期内函数二次返回数字。

```
int PeriodSeconds(  
    ENUM_TIMEFRAMES period=PERIOD_CURRENT // 图表周期  
);
```

参量

period=PERIOD_CURRENT

[in] [ENUM_TIMEFRAMES](#)中的图表周期值，如果参量未标明，在程序运行中，返回当前图表周期二次数字。

返回值

所选周期中的二次数字。

另见

[_Period](#), [图表时间表](#), [日期和时间](#), [对象可见性](#)

Playsound

用来播放声音文件。

```
bool   Playsound(  
    string filename    // 文件名  
);
```

参量

filename

[in] 到声音文件的路径。

返回值

true – 如果发现文件， 否则 - false.

注释

文件加载到terminal_ directory\Sounds 的子目录中，只能播放wav类型文件。

另见

Print

在EA交易日志中输入消息。参量可以是各种类型。

```
void Print(  
    argument,    // 第一值  
    ...          // 下一值  
);
```

参量

...

[in] 逗号分割任意值，参量数不能超过64

注释

数组不能通过Print() 函数传递，数组必须按元素一个接一个输入

双精度型数据多余小数点后16位字节显示，或以传统或以科学角度输出，根据接入口的紧凑性。浮点类型数据在小数点后5位字节输出，输出的实型数据和另一精确度的预先格式，使用 [DoubleToString\(\)](#) 函数。

布尔型数据的输出以true或者false标准，数据以 YYYY.MM.DD HH:MI:SS为标准，在另一版本表示数据，使用 [TimeToString\(\)](#)，如果颜色类型显示在操作中，颜色类型数据以RGB，或颜色名称来返回。

示例：

```
Print("Time of the last received quote ",TimeCurrent());
```

另见

[PrintFormat](#) , [StringFormat](#)

PrintFormat

交易品种的形式和进入，EA交易的值与当前形式保持一致。

```
void PrintFormat(
    string format_string,    // 格式行
    ...                     // 简单类型值
);
```

参量

- format_string
- [in] A格式包括简单交易品种，如果模式线接下来是参数，也包括模式规格
- ...
- [in] 简单类型的任何值都被逗号分开，参数的全部数量都不能超过64个，包括格式化路线

返回值

字符串。

注释

用 printf() 。

如果格式路线随后是参量，这条路线一定包括这些参量表示输出的格式，规格总是以百分比符号(%) 开始
格式路线从左至右阅读，当遇见第一个格式（如果有的话），在格式线路后的第一个参量值根据当前规格改变并输出。第二个格式根据第二个参量改变并输出，以此类推直到格式化线路结束。

格式规格如下形式：

%[flags][width][.precision][{ h | l | ll | l32 | l64 }]type

每个格式规格的域或者是一个简单的交易对象，或者表示一般选项的数字，最简单的规格只包括百分号(%) 和定义成 输出参量类型 的交易品种（例如%s），如果在格式线上需要输出当前符号，使用格式规格%%。

标记

标记	描述	默认动作
- (减号)	活字宽度左对齐	右对齐
+ (加号)	+ or 的输出-符号类型的符号值	符号只在负值中显示
0 (零)	在当前 宽度 输出值前添加0，如果0标签指定整数格式(i, u, x, X, o, d) ，设立精确度格式（例如， %04.d），然后忽略0	什么都不添加
空间	如果有符号和正值，在输出值中显示空间	不插入空间

#	如果连用格式 o, x 或者 X,在输出0值之前,分别添加0x 或者 0X	什么都不添加
	如果连用格式 e, E, a 或者 A,值会显示小数点	如果小数部分非零,显示小数点
	如果连用格式g或者G,标签在输出值中存在小数点,阻止切断前导零,当一起使用格式c, d, i, u, s时,忽视标签#	如果小数部分非零,显示小数点,切断前导零

宽度

建立非负小数,格式化值中的交易品种输出的最小数量。如果输出交易品种数量少于活字宽度,类似的空间数量依据队列(标签-)从左至右添加,如果有标签0(0),类似0数值在输出值之前添加,如果输出值数量比活字宽度大,输出值不会被切断

如果星号(*)规定宽度,整型值应该在传递参量列表中类似位置显示,用来说明输出值的宽度。

精确度

非负的小数建立在输出精确度-小数点后几位数字。与规定宽度不同,精确度规格切断部分类型。

使用规格精确度与不同的板式 [类型](#) 不同

类型	描述	默认动作
a, A	精确度说明在小数点后建立的数字数量	默认精确度-6
c, C	不使用	
d, i, u, o, x, X	集合输出数字的最少数量,如果数字数量与常量差不多但小于精确度,在输出值左边加上0,如果输出数字的数量比规定精确度多,输出值不能中断。	默认精确度-1
e, E, f	在小数点后设置若干输出数量,最后的数字四舍五入	默认精确度-6,如果建立的精确度是0或者小数部分空缺,不显示小数点
g, G	设立有意义数字的最大数量	输出六位有意义数字
s, S	建立线型输出交易品种数量,如果线的长度超过精确度,线就会切断	整个线路输出

h | l | ll | l32 | l64

数据大小说明,以参量传递。

参量类型	使用前缀	类型共同说明符
整型	l (小写字母 L)	d, i, o, x, or X
无符号整型	l (小写字母 L)	o, u, x, or X
长整型	ll (两个小写字母 L)	d, i, o, x, or X

短整型	h	d, i, o, x, or X
无符号短整型	h	o, u, x, or X
整型	l32	d, i, o, x, or X
无符号整型	l32	o, u, x, or X
长整型	l64	d, i, o, x, or X
无符号长整型	l64	o, u, x, or X

类型

类型说明符只在强制域格式化输出

符号	类型	输出格式
c	整型	短型交易品种 (统一码)
C	整型	图表型交易品种(ANSI)
d	整型	小数整数记号
i	整型	小数整数记号
o	整型	未标签的八进制整数
u	整型	未标签的小数整数
x	整型	未标签的十六进制整数, 使用"abcdef"
X	整型	未标签的十六进制整数, 使用 "ABCDEF"
e	双精度	在 [-] d.dddde[sign] ddd, where d-十进制数字, dddd -一位或更多小数位, ddd -三位数字觉得类型大小 格式中真值, 标志-正负号
E	双精度	与e格式相同, 除了典型标志通过大写字母标出 (E代替e)
f	双精度	格式中的真值[-] dddd.dddd, dddd -一位或更多小数位数, 在小数点前显示数字的数量取决于符号值的大小, 小数点后的数字数量取决于需求
g	双精度	f或者e格式的输出真值取决于输出量更紧密
G	双精度	F或者E的输出真值取决于输出量更紧密
a	双精度	格式中的实数 [-]0xh.hhhh p±dd, h.hhhh -十六进制格式的尾数, 使用"abcdef", dd -一个或者更多典型数字, 小数位的数量由 精确度说明 决定。
A	双精度	格式中的实数 [-]0xh.hhhh P±dd, h.hhhh-十六进制格式的尾数, 使用"ABSDEF", dd -一个或者更多典型数字, 小数位的数量由 精确度说明 决定。
s	字符串	路线输出

取代PrintFormat() , 可以使用printf() 。

示例：

```
PrintFormat("Output DBL_MAX in a compact scientific form %e", DBL_MAX);  
printf("Output double %.15e", DBL_MAX);  
printf("Output double %15e", DBL_MAX);  
printf("Output double %15.10e", DBL_MAX);  
printf("Output double %15.10f", DBL_MAX);  
  
printf("Output DBL_MAX %e", 10);  
printf("Output double %d", DBL_MAX);  
printf("Output DBL_MAX %s", 10);  
printf("Output double %s", DBL_MAX);
```

另见

[StringFormat](#) , [DoubleToString](#) , [真实类型（双精度，浮点）](#)

ResetLastError

设立预先定义常量 [__LastError](#) 的值是0。

```
void ResetLastError();
```

返回值

没有返回值

注释

如果需要注释函数 [GetLastError\(\)](#) 是zero _LastError 变量, 在函数调用之前 [错误](#) 外观检测后通常调用 ResetLastError() 。

SetUserError

创建预先参量 `__LastError` , 使其值等于 `ERR_USER_ERROR_FIRST` + `user_error`

```
void SetUserError(  
    ushort user_error,    // 错误号  
);
```

参量

`user_error`

[in] 用户设置的[错误](#)编号。

返回值

没有返回值

注释

在使用 `SetUserError(user_error)` 函数发生错误后, `GetLastError()` 返回值等于 `ERR_USER_ERROR_FIRST` + `user_error`.

注释 :

```
void OnStart()  
{  
    //--- 设置错误号 65537=( ERR_USER_ERROR_FIRST +1)  
    SetUserError(1);  
    //--- 获得上一个错误代码  
    Print("GetLastError = ",GetLastError());  
    /*  
        Result  
        GetLastError = 65537  
    */  
}
```

SendFTP

在地址中发送文件，指定在Publisher窗口中。

```
bool  SendFTP(  
    string  filename,           // 通过ftp发送的文件  
    string  ftp_path=NULL      // ftp服务器上传的文件  
);
```

参量

filename

[in] 发送文件名称

ftp_path=NULL

[in] FTP目录，如果目录未标明，直接在使用设置中描述

返回值

错误返回是“false”。

注释

发送文件应该在 terminal_directory\MQL5\files 或子文件夹中保存，如果在设置中未标明FTP地址或者访问密码，无法发送。

SendMail

在Mailbox建立的窗口中发送指定地址邮件。

```
bool SendMail(  
    string subject,      // 表头  
    string some_text     // email 文本  
);
```

参量

subject

[in] 邮件标题

some_text

[in] 邮件主题

返回值

如果邮件发送到派送队列则true，否则是false

注释

发送在设置中能被禁止，邮件地址可能未标明，查看错误信息调用 [GetLastError\(\)](#)

Sleep

在当前执行的EA交易或脚本中，该函数是指在指定的时间间隔内暂停交易业务。

```
void Sleep(  
    int milliseconds    // 间隔  
);
```

参量

milliseconds

[in] 毫秒之内的内部延期

返回值

没有返回值

注释

The Sleep() 函数不能被自定义指标调用，因为指标是以界面线路通过并不能够放慢速度，智能交易每0.1 second秒会检测。

TerminalClose

该函数命令服务器完整操作。

```
bool TerminalClose(
    int ret_code    // 关闭客户端代码
);
```

参量

ret_code

[in] 返回代码，在操作完成后通过客户端返回。

返回值

函数成功true，否则false。

注释

TerminalClose() 函数并不能立即阻止客户端，它只命令客户端完成操作。

EA交易的代码叫做TerminalClose()，它来执行立即操作（比如，所有先前开放文件在正常模式下都要先关闭），该函数的调用通过 [返回操作符](#) 执行。

ret_code 参数程序分析原因返回的必要代码，在命令提示符启动后，客户端操作终止

示例：

```
//--- 输入参量
input int   tiks_before=500; // 订单号数目直到终止
input int   pips_to_go=15;   // pips 间的距离
input int   seconds_st=50;   // 给予EA交易的秒数
//--- 全局
datetime    launch_time;
int         tick_counter=0;
//+-----+
//| 专家无法初始化函数                      |
//+-----+
void OnDeinit(const int reason)
{
    //---
    Print(__FUNCTION__, " reason code = ", reason);
    Comment("");
}
//+-----+
//| 专家订单号函数                          |
//+-----+
void OnTick()
{
    static double first_bid=0.0;
    MqlTick      tick;
```

```

double      distance;
//---
SymbolInfoTick(_Symbol,tick);
tick_counter++;
if(first_bid==0.0)
{
    launch_time=tick.time;
    first_bid=tick.bid;
    Print("first_bid =",first_bid);
    return;
}
//--- pips 的价格距离
distance=(tick.bid-first_bid)/_Point;
//--- 通过EA操作显示告示
string comm="From the moment of start:\r\n\x25CF elapsed seconds: "+
            IntegerToString(tick.time-launch_time)+" ;"+
            "\r\n\x25CF ticks received: "+(string)tick_counter+" ;"+
            "\r\n\x25CF price went in points: "+StringFormat("%G",distance);
Comment(comm);
//--- 检测关闭程序端状态的部分
if(tick_counter>=ticks_before)
    TerminalClose(0);    // 推出订单号计数器
if(distance>pips_to_go)
    TerminalClose(1);    // 同于 pips_to_go 增长 pips 数量
if(distance<-pips_to_go)
    TerminalClose(-1);   // 同于 pips_to_go 下降 pips 的数量
if(tick.time-launch_time>seconds_st)
    TerminalClose(100);  // 超时终止
//---
}

```

另见

[程序运行 - 执行错误, 无法初始化原因](#)

TesterStatistics

```
double TesterStatistics(  
    ENUM_STATISTICS statistic_id // идентификатор  
);
```

statistic_id

[in]

[ENUM_STATISTICS](#).

[OnTester\(\)](#)

[OnDeinit\(\)](#)

TesterWithdrawal

在测试过程中特殊函数模拟取消内存操作，可以用于资产管理系统中。

```
bool TesterWithdrawal(  
    double money    // 出金总额  
);
```

参量

money

[in] 要求出金的总钱数（用如今货币）

返回值

成功，返回 true，否则 - false.

ZeroMemory

通过引用，函数重置变量

```
void ZeroMemory(  
    void & variable    // 变量重置  
);
```

参量

variable

[in] [out] 变量通过引用传递，然后重置（从0开始初始化）

返回值

没有返回值

注释

如果函数参数是字符串，调用为NULL值

对于简单类型或数组，和由这些类型组成的结构和分类，都要简单重置

该对象包括串型和动态数组，ZeroMemory() 调用每个元素

对于没有被参量修饰语保护的数组，所有元素都归零。

复杂对象的数字，ZeroMemory() 调用每个元素

ZeroMemory() 不能应用到受 [成员](#) 或 [继承算法](#) 保护的分类中

使用数组的函数组

数组的最大维数为四维。每个维数被索引编为从0 至dimension_size-1。事实上，第一维数组的50 个，在调用时第一个数组显示为[0],最后一个数组显示为[49]。

函数	功能
ArrayBsearch	在第一函数维度索引第一组发现的元素
ArrayCopy	复制一组数组到另一组
ArrayFree	解放任意动态数组的缓冲区并建立0维度大小
ArrayGetAsSeries	检测数组标引的导向
ArrayInitialize	给数字数组所有元素一个单独的值
ArrayIsSeries	检测是否数组是时序列
ArrayIsDynamic	检测是否数组是动态的
ArrayMaximum	检测元素的最大值
ArrayMinimum	检测元素的最小值
ArrayRange	在数组的指定维度返回元素数量
ArrayResize	在数组的第一维尺寸设定新大小
ArraySetAsSeries	指明数组标引的导向
ArraySize	返回数组中元素的数量
ArraySort	通过第一维度排列数值数组

ArrayBsearch

在一维数组里，该函数搜寻额定值。

搜寻双精度数组

```
int ArrayBsearch(  
    double    array[],           // 用来搜索的数组  
    double    searched_value     // 搜索什么  
);
```

搜寻浮点型数组

```
int ArrayBsearch(  
    float     array[],           // 用来搜索的数组  
    float     searched_value     // 搜索什么  
);
```

搜寻长整型数组

```
int ArrayBsearch(  
    long      array[],           // 用来搜索的数组  
    long      searched_value     // 搜索什么  
);
```

搜寻整型数组

```
int ArrayBsearch(  
    int       array[],           // 用来搜索的数组  
    int       searched_value     // 搜索什么  
);
```

搜寻短整型数组

```
int ArrayBsearch(  
    short     array[],           // 用来搜索的数组  
    short     searched_value     // 搜索什么  
);
```

搜寻字符型数组

```
int ArrayBsearch(  
    char      array[],           // 用来搜索的数组  
    char      searched_value     // 搜索什么  
);
```

参量

`array[]`
[in] 搜寻数字数组。

`searched_value`
[in] 搜寻值

返回值

函数返回创立元素索引，如果需要值没找到，函数返回最相近值。

注释

二进制检索过程只在存储数组中进行，使用 [ArraySort\(\)](#) 函数分类数字数组。

ArrayCopy

复制一组数组到另一组。

```
int ArrayCopy(  
    void dst_array[],           // 目标数组  
    void src_array[],          // 源数组  
    int dst_start=0,           // 写入目标数组的指数  
    int src_start=0,           // 源数组的最初指数  
    int cnt=WHOLE_ARRAY        // 元素数量  
);
```

参量

dst_array[]
[out] 目标数组

src_array[]
[in] 源数组

dst_start=0
[in] 从目标数组的第几位开始写入，默认为0。

src_start=0
[in] 从源数组的第几位开始读取，默认为0。

cnt=-1
[in] 复制数组的数量，默认全部数组复制(cnt=WHOLE_ARRAY)。

返回值

返回复制元素的数量

注释

如果cnt<0 or cnt>src_ size-src_ start,所有余下的数组部分都被复制，数组从左至右复制，对于一系列数组来说，开始位置为从左至右进行复制正确定义，如果数字自身复制，该结果未定义。

如果数组存在不同种类型，在复制期间试着改变源数组的每个元素到目标数组。字符串型数组只可以复制到字符串型数组中间，[类和结构](#)数组包括的需要初始化的对象不能复制。结构的一组数组只可以复制成相同类型数组。

ArrayFree

任何动态数组会空出来缓冲区并建立0维大小

```
void ArrayFree(  
    void array[]    // 数组  
);
```

参量

array[]
[in] 动态数组。

返回值

没有返回值

ArrayGetAsSeries

检测数组索引的导向

```
bool ArrayGetAsSeries(  
    void array    // 用于检测的数组  
);
```

参量

array

[in] 检测数组

返回值

返回 [true](#)，如果规定数组建立了AS_ SERIES 标志，如下，访问数组以时序列无序执行，一个 [时间序列](#) 与普通数组通常在从未到头（从最新数据到最老）执行时序列元素索引不同。

注释

检验数组是否属于时序列，使用 [ArrayIsSeries\(\)](#) 函数。价格数组通过输入参量无需强制传递到[OnCalculate\(\)](#) 函数中。必要的搜索导向可以通过使用 [ArraySetAsSeries\(\)](#) 函数建立。

另见

[接入时间序列](#) - [ArraySetAsSeries](#)

ArrayInitialize

函数通过预设值初始化一组数字数组。

```
void ArrayInitialize(  
    double array[],      // 初始化的数组  
    double value         // 将被设置的值  
);
```

参量

array[]

[out] 需要初始化的数值数组

value

[in] 为所有数组元素建立新值

返回值

没有返回值

注释

[ArrayResize\(\)](#) 函数允许建立为无需人工安置扩散而保留的数组内存大小。它为了更好的属性而实施，因为内存重新安置操作相当慢。

使用 [ArrayInitialize\(\)](#) 初始化不表示储备元素相同值分配给数组。使用 [ArrayResize\(\)](#) 函数进行数组的长远扩展，元素会添加到数组末尾，其他值是未下定义的病情在大多数情况下不等于 `init_value`。

示例：

```
void OnStart()  
{  
    //--- 动态数组  
    double array[];  
    //--- 数组大小设为100个元素并为另10个元素保留缓冲区  
    ArrayResize(array,100,10);  
    //--- 初始化带有EMPTY_VALUE=DBL_MAX值的数组元素  
    ArrayInitialize(array,EMPTY_VALUE);  
    Print("Values of 10 last elements after initialization");  
    for(int i=90;i<100;i++) printf("array[%d] = %G",i,array[i]);  
    //--- 数组扩展5个元素  
    ArrayResize(array,105);  
    Print("Values of 10 last elements after ArrayResize(array,105)");  
    //--- 最后的5个数组值可以从保留的缓冲区获得  
    for(int i=95;i<105;i++) printf("array[%d] = %G",i,array[i]);  
}
```

ArrayIsDynamic

判断动态数组的函数。

```
bool ArrayIsDynamic(  
    void array[] // 已检测的数组  
);
```

参量

array[]
[in] 检测数组.

返回值

数组是[动态](#)，返回 true，否则返回 false。

另见

[接入时间序列和指标](#)

ArrayIsSeries

检测时间序列数组的函数。

```
bool ArrayIsSeries(
    void array[] // 已经检测的数组
);
```

参量

array[]
[in] 检测数组

返回值

如果检测数组是时序列返回真值，否则返回错误值。数组以参量形式传递到[OnCalculate\(\)](#)函数一定通过[ArrayGetAsSeries\(\)](#)检测访问数组元素命令

示例:

```
#property indicator_chart_window
#property indicator_buffers 1
#property indicator_plots 1
//---- 标签图1
#property indicator_label1 "Label1"
#property indicator_type1 DRAW_LINE
#property indicator_color1 clrRed
#property indicator_style1 STYLE_SOLID
#property indicator_width1 1
//--- 指标缓冲区
double Label1Buffer[];
//+-----+
//| 自定义指标初始化函数 |
//+-----+
void OnInit()
{
//--- 指标缓冲区绘图
SetIndexBuffer(0,Label1Buffer,INDICATOR_DATA);
//---
}
//+-----+
//| 自定义指标重复函数 |
//+-----+
int OnCalculate(const int rates_total,
                const int prev_calculated,
                const datetime &time[],
                const double &open[],
                const double &high[],
                const double &low[],
                const double &close[],
                const long &tick_volume[],
```

```
        const long &volume[],
        const int &spread[])
    {
//---
        if(ArrayIsSeries(open))
        {
            Print("open[] is timeseries");
        }
        else
        {
            {
                Print("open[] is not timeseries!!!");
            }
        }
//--- 为下次调用返回prev_calculated值
        return(rates_total);
    }
```

另见

[接入时间序列和指标](#)

ArrayMaximum

在一维数字数组里该函数搜寻最大元素

```
int ArrayMaximum(  
    double array[],           // 用于搜索的数组  
    int start=0               // 启动用于检测的指标  
    int count=WHOLE_ARRAY,    // 检测元素数量  
);
```

参量

array[]

[in] 数字数组，建立搜寻

start=0

[in] 开始符合性搜索

count=WHOLE_ARRAY

[in] 搜索元素的数量。默认在整个数组 (cnt=[WHOLE_ARRAY](#)) 中搜索。

返回值

该函数返回找到元素的索引算作数组 [系列](#)，如果失败返回-1。

ArrayMinimum

在一维数字数组里该函数搜寻最小元素

```
int ArrayMinimum(  
    double array[],           // 用来搜索的数组  
    int start=0               // 启动用于检测的指标  
    int count=WHOLE_ARRAY,    // 检测元素数量  
);
```

参量

array[]

[in] 搜索数字数组

start=0

[in] 标引开始检测

count=WHOLE_ARRAY

[in] 搜索元素数量，默认搜索全部数组 (cnt=[WHOLE_ARRAY](#)) .

返回值

函数返回发现元素的标引索引，考虑数组[系列](#)，失败返回-1

ArrayRange

在所选数组规格中该函数返回元素数字

```
int ArrayRange(  
    void array[],          // 用于检测的数组  
    int rank_index         // 维数  
);
```

参量

array[]
[in] 检测数组

rank_index
[in] 标准索引

返回值

在所选数组规格中的元素数量

注释

从零开始搜索，维度尺寸大于最大搜索1

ArrayResize

函数建立新的第一维大小

```
int ArrayResize(  
    void array[],           // 引用传递的数组  
    int new_size,           // 新数组大小  
    int reserve_size=0      // 保留尺寸值 (过量)  
);
```

参量

array[]

[out] 数组变化尺寸

new_size

[in] 第一维度新尺寸

reserve_size=0

[in] 获得存储的分散式大小

返回值

如果执行成功，在数组中改变规格后返回所有元素计数，否则，返回-1，数组不调整大小

注释

在内存分配中考虑reserve_size参量，如果指定参量，则设置了数组的附加内存大小。重复调用ArrayResize不能导致物理内存的重复分配，只有第一数组尺寸在保留内存中被调整。

函数只用于[动态数组](#)。注意的是不能改变通过函数[SetIndexBuffer\(\)](#)设置指标缓冲区的动态数组的大小。对于指标缓冲区，所有改变大小 的操作都通过程序端的执行子系统完成。

另见

[ArrayInitialize](#)

ArraySetAsSeries

设置标记到选定的动态数组对象的函数，其元件如时间序列中一样被索引。

```
bool ArraySetAsSeries(
    void array[], // 通过引用的数组
    bool set      // true表示倒序索引
);
```

参量

`array[]`
[in][out] 设置数字数组

`set`
[in] 数组索引方向

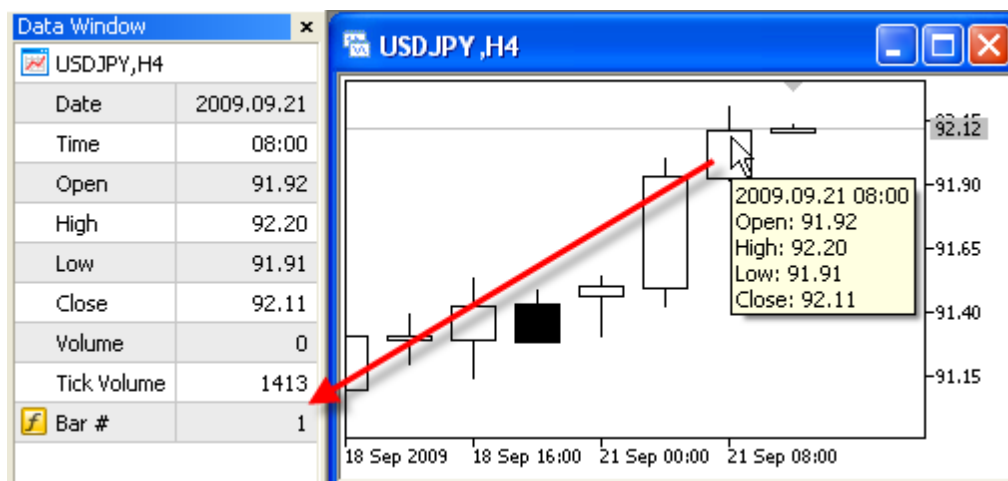
返回值

成功函数返回true，否则 - false.

注释

不能为多维数组或静态数组（数组，方框中的大小在编译步骤已经存在）设置 [AS_SERIES](#) 标记。时间序列中的索引不同于时间序列元件中普通数组，从后往前（从最新的到最原始数据）。

示例：显示柱值的指标



```
//+-----+
//|                                     BarNumber.mq5 |
//|                                     Copyright 2009, MetaQuotes Software Corp. |
//|                                     http://www.mql5.com |
//+-----+

#property copyright "2009, MetaQuotes Software Corp."
#property link      "http://www.mql5.com"
#property version   "1.00"
#property indicator_chart_window
```

```

#property indicator_buffers 1
#property indicator_plots 1
//---- 图的编号
#property indicator_label1 "Numeration"
#property indicator_type1 DRAW_LINE
#property indicator_color1 CLR_NONE
//--- 指标缓冲区
double NumerationBuffer[];
//+-----+
//| 自定义指标初始化函数 |
//+-----+
int OnInit()
{
//--- 自定义缓冲区绘图
SetIndexBuffer(0,NumerationBuffer,INDICATOR_DATA);
//--- 为类似时间序列的缓冲区设置指标
ArraySetAsSeries(NumerationBuffer,true);
//--- 设置数据窗口中显示的精确度
IndicatorSetInteger(INDICATOR_DIGITS,0);
//--- 标识符数组名称在数据窗口中如何显示
PlotIndexSetString(0,PLOT_LABEL,"Bar #");
//---
return(0);
}
//+-----+
//| 自定义指标重复函数 |
//+-----+
int OnCalculate(const int rates_total,
               const int prev_calculated,
               const datetime &time[],
               const double &open[],
               const double &high[],
               const double &low[],
               const double &close[],
               const long &tick_volume[],
               const long &volume[],
               const int &spread[])
{
//--- 存储当前打开的零柱时间
static datetime currentBarTimeOpen=0;
//--- 返回访问数组时间[] - 像时间序列一样
ArraySetAsSeries(time,true);
//--- 如果零柱时间不同于已经存储的
if(currentBarTimeOpen!=time[0])
{
//--- 从当前图表到图表深度列举所有柱
for(int i=rates_total-1;i>=0;i--) NumerationBuffer[i]=i;
currentBarTimeOpen=time[0];
}
}

```

```
    }  
    // --- 为下次调用返回prev_calculated值  
    return(rates_total);  
}
```

另见

[接入时间序列](#) , [ArrayGetAsSeries](#)

ArraySize

函数返回所选数组的元素数量

```
int ArraySize(  
    void array[]    // 检测的数组  
);
```

参量

`array[]`
[in] 任意类型的数组

返回值

整型 值

注释

一维数组，值可以通过 [ArraySize](#) 返回，等于 [ArrayRange](#)(数组,0)

ArraySort

函数从左至右升序排列数字数组。

```
bool ArraySort(  
    number& array[] // 数组排序  
);
```

参量

array[]
[in][out] 数字数组分类

返回值

成功返回真值，否则是错误值。

注释

数组通过 [AS_SERIES](#) 标签建立，并降序排列。

函数转换

函数组提供从一组数据到另一种数据的转换。

[NormalizeDouble\(\)](#) 函数可以特殊标识，它提供必要的价值描述精确度。在交易操作中，如果交易服务器超过了字节要求的精确度，不能使用非标准价格。

函数	功能
CharToString	转换符号代码到字符串
DoubleToString	以规定的精确度转换数字值到文本行
EnumToString	
NormalizeDouble	浮点数转换到规定的精确度
StringToDouble	转换包括数字代表符的字符串到双精度数字
StringToInteger	转换包括数字代表符的字符串到整型数字
StringToTime	转换包括 "yyyy.mm.dd [hh:mi]" 格式的时间或者日期字符串到日期时间型
TimeToString	转换01.01.1970值到 "yyyy.mm.dd hh:mi" 字符串格式
IntegerToString	整型转换成当前长度字符串
ShortToString	转换符号代码（双字节编码）到单字符串
ShortArrayToString	复制数组部分到字符串
StringToShortArray	符号-wise复制字符串到选定的无符号短整型部分数组
CharArrayToString	转换符号代码（ansi）到单符号数组
StringToCharArray	符号-wise 复制双字节编码到ANSI转换的字符串，到选定的无符字符型数组
ColorToString	转换颜色值到"R,G,B"形式的字符串。
StringToColor	转换"R,G,B" 字符串或者颜色名的字符串到颜色类型值
StringFormat	按照现格式转换数字到字符串

CharToString

将字符型转换成字符串型结果。

```
string CharToString(  
    uchar char_code    // 交易品种的数值代码  
);
```

常量

char_code
[in] 统一码交易品种代码

返回值

统一码交易品种的字符串。

CharArrayToString

将双字符型部分数组复制或转换成返回型字符串

```
string CharArrayToString(  
    uchar array[],           // 数组  
    int start=0,             // 数组启动位置  
    int count=-1             // 交易品种数  
    uint codepage=CP_ACP     // 代码页  
);
```

常量

`array[]`

[in] 双字符型数组

`start=0`

[in] 复制开始位置，默认值是0

`count=-1`

[in] 复制的数组元素数量，定义结果字符串长度，默认值是-1，代表从数组末尾开始复制，或者从0终端开始

`codepage=CP_ACP`

[in] 代码页值，为最多使用的 [code pages](#) 提供适当常量。

返回值

字符串

ColorToString

转换颜色值到字符串RGB形式

```
string ColorToString(  
    color color_value,      // 颜色值  
    bool  color_name       // 是否显示颜色名称  
);
```

常量

color_value

[in] 颜色类型变量颜色值

color_name

[in] 如果颜色名称与预定义的 [颜色常量](#)之一相同，有必要返回颜色名称标签。

返回值

字符串颜色描述成RGB，RGB小数位常量从0到255转变成字符串。如果设置color_name=true参量，就会把颜色值转变成颜色名称

示例：

```
string clr=ColorToString(C'0,255,0'); // 绿色  
Print(clr);  
  
clr=ColorToString(C'0,255,0',true);   // 获得颜色常量  
Print(clr);
```

DoubleToString

把数字值转换成文本串

```
string DoubleToString(  
    double value,      // 数字  
    int digits=8       // 小数点后的数字数  
);
```

参量

value

[in] 浮点值

digits

[in]精确模式。如果digits值在0到16之间，字符串描述数值是在小数点后特定的数字。如果digits值在-1到-16之间，字符串描述数值是小数点后的标准数值。在其他任何情况下在小数点后保留8位。

返回值

字符串包括交易品种代表规定精确度数量。

示例：

```
Print("DoubleToString(120.0 + M_PI) : ",DoubleToString(120.0+M_PI));  
Print("DoubleToString(120.0 + M_PI,16) : ",DoubleToString(120.0+M_PI,16));  
Print("DoubleToString(120.0 + M_PI,-16) : ",DoubleToString(120.0+M_PI,-16));  
Print("DoubleToString(120.0 + M_PI,-1) : ",DoubleToString(120.0+M_PI,-1));  
Print("DoubleToString(120.0 + M_PI,-20) : ",DoubleToString(120.0+M_PI,-20));
```

另见

[正常化双精度](#)， [字符串到双精度](#)

EnumToString

```
string EnumToString(  
    any_enum value    // значение из перечисления любого типа  
);
```

value
[in]

[GetLastError\(\)](#).

[__LastError](#)

- ERR_INTERNAL_ERROR -
- ERR_NOT_ENOUGH_MEMORY -
- ERR_INVALID_PARAMETER -

```

enum interval // перечисление именованных констант
{
    month=1,      // интервал в один месяц
    two_months,   // два месяца
    quarter,      // три месяца - квартал
    halfyear=6,   // полугодие
    year=12,      // год - 12 месяцев
};
//+-----+
//| Script program start function |
//+-----+
void OnStart()
{
    //--- установим временной интервал равным месяцу
    interval period=month;
    Print(EnumToString(period)+"="+IntegerToString(period));

    //--- установим временной интервал равным кварталу (три месяца)
    period=quarter;
    Print(EnumToString(period)+"="+IntegerToString(period));

    //--- установим временной интервал равным году (12 месяцев)
    period=year;
    Print(EnumToString(period)+"="+IntegerToString(period));

    //--- проверим как выводится тип ордера
    ENUM_ORDER_TYPE type=ORDER_TYPE_BUY;
    Print(EnumToString(type)+"="+IntegerToString(type));

    //--- проверим как выводится неверное значение
    type=WRONG_VALUE;
    Print(EnumToString(type)+"="+IntegerToString(type));

    // Результат выполнения:
    // month=1
    // quarter=3
    // year=12
    // ORDER_TYPE_BUY=0
    // ENUM_ORDER_TYPE::-1=-1
}

```

_____ , [Input](#)

IntegerToString

函数将整数类型值转变成特定长度的字符串值并返回包含的字符串。

```
string IntegerToString(  
    long    number,           // 数字  
    int     str_len=0,       // 字符串结果长度  
    ushort  fill_symbol=' '  // 填充物  
);
```

参量

number

[in] 转换数量

str_len=0

[in] 字符串长度。如果结果字符串长度超过指定长度，字符串不会切断，如果短，从左边添加交易品种。

fill_symbol=' '

[in] 填补交易品种，默认分开

返回值

字符串。

ShortToString

把交易品种代码（统一码）转变成字符串交易品种中的一中，返回结果字符串。

```
string ShortToString(  
    ushort symbol_code    // 交易品种  
);
```

参量

symbol_code

[in] 交易品种代码，可以使用包含交易品种或2字节十六进制代码的文本字符串代替交易品种代码，但在同一码平台上与交易品种代码相一致。

返回值

字符串。

ShortArrayToString

把数组的复制部分转到返回字符串。

```
string ShortArrayToString(  
    ushort array[],      // 数组  
    int start=0,         // 数组中的启动位置  
    int count=-1         // 交易品种数  
);
```

参量

array[]

[in] 无符号短整型数组（模拟wchar_t 型）

start=0

[in] 位置，从复制开始起，默认是0

count=-1

[in] 复制的数组元素数字，定义结果字符串的长度，默认值是-1，表示复制到数组末尾，否则客户端仍旧是0

返回值

字符串。

TimeToString

转变值包括从01.01.1970起已消耗的秒数，以字符串格式"yyyy.mm.dd hh:mi"

```
string TimeToString(  
    datetime value,                // 数字  
    int mode=TIME_DATE|TIME_MINUTES // 输出形式  
);
```

参量

value

[in] 时间从00:00 1970/01/01开始

mode=TIME_DATE|TIME_MINUTES

[in] 额外数据输入模式，可以是一个也可以包括以 "yyyy.mm.dd"为形式的标签TIME_DATE，以"hh:mi"为结果的 TIME_MINUTES，以 "hh:mi:ss"为结果的TIME_SECONDS。

返回值

字符串。

NormalizeDouble

把浮点类型转变成指定精确度

```
double NormalizeDouble(
    double value,      // 标准化号码
    int     digits     // 小数点后的数字数
);
```

参量

value

[in] 浮点类型值

digits

[in] 精确度模式，浮点数字 (0-8) .

返回值

预先设置的双精度类型值

注释

待办订单止损数值，目标数值和开仓值的计算值一定根据精确度规范吗，该值可以通过 [Digits\(\)](#) 获得。

示例：

```
double pi=M_PI;
Print("pi = ",DoubleToString(pi,16));

double pi_3=NormalizeDouble(M_PI,3);
Print("NormalizeDouble(pi,3) = ",DoubleToString(pi_3,16))
;
double pi_8=NormalizeDouble(M_PI,8);
Print("NormalizeDouble(pi,8) = ",DoubleToString(pi_8,16));

double pi_0=NormalizeDouble(M_PI,0);
Print("NormalizeDouble(pi,0) = ",DoubleToString(pi_0,16));
/*
Result:
pi= 3.1415926535897931
NormalizeDouble(pi,3)= 3.1419999999999999
NormalizeDouble(pi,8)= 3.1415926499999998
NormalizeDouble(pi,0)= 3.0000000000000000
*/
```

另见so

[双精度到字符串](#)， [真实型（双精度，浮点）](#)， [类型减少](#)

StringToCharArray

交易品种复制从一个Unicode型字符串转变成ANSI，双字符型数组的选择位置，返回复制元素的数量。

```
int StringToCharArray(  
    string text_string,           // 源字符串  
    uchar& array[],              // 数组  
    int start=0,                  // 数组中的启动位置  
    int count=-1                  // 交易品种数  
    uint codepage=CP_ACP         // 代码页  
);
```

参量

text_string

[in] 复制字符串型

array[]

[out] 无符字符型数组

start=0

[in] 从起始位置开始复制，默认值是0

count=-1

[in] 数组元素复制数量，定义字符串结果的长度，默认值是-1，会一直复制到数组的结尾，或者是终端0。终端0也会被接收数组复制，在此情形下，如果字符串大小需要，动态数组的大小是能增加的，如果动态数组增大到跟路线一样长，数组的体积不会减小。

codepage=CP_ACP

[in] 代码页的值，最多使用 [代码页](#) 提供适当常量

返回值

复制元素的数量

StringToColor

使用颜色名称将RGB字符串型转变成颜色类型值

```
color StringToColor(  
    string color_string    // 字符串颜色表示  
);
```

参量

color_string

[in] 代表颜色的RGB类型字符串类型或者其中一种预定义 [网页-颜色](#) 名称

返回值

颜色值

示例：

```
color str_color=StringToColor("0,127,0");  
Print(str_color);  
Print((string)str_color);  
//--- 变换颜色  
str_color=StringToColor("0,128,0");  
Print(str_color);  
Print((string)str_color);
```

StringToDouble

将字符串转变成数字的双精度型

```
double StringToDouble(  
    string value    // 字符串  
);
```

参量

value

[in] 数字的字符串换行格式

返回值

双精度型值

StringToInteger

将字符串型转换成整型结果。

```
long StringToInteger(  
    string value    // 字符串  
);
```

参量

value

[in] 数字的字符串转换格式

返回值

整型值

StringToShortArray

函数的交易品种复制一组字符串到长短型数组的规定地点，返回复制元素的数量。

```
int StringToShortArray(  
    string text_string,      // 源字符串  
    ushort& array[],        // 数组  
    int start=0,            // 数组启动位置  
    int count=-1            // 交易品种数  
);
```

参量

text_string

[in] 字符串复制

array[]

[out] [无符短型](#) 型数组 (wchar_t 模拟类型)

start=0

[in] 位置，从复制值开始，默认值是0

count=-1

[in] 复制的数组元素的数量，定义字符串结果的长度，默认值是-1，意味着根据数组的结尾来进行复制，直到0是终点。终点0也会复制到接收数组，在此情形下，如果字符串大小需要，动态数组的大小是能增加的，如果动态数组增大到跟路线一样长，数组的体积不会减小。

返回值

复制元素的数量

StringToTime

函数转变一组以时间日期"yyyy.mm.dd [hh:mi]" 为标本的字符串样式为日期时间型

```
datetime StringToTime(  
    string value      // 日 前 字 符 串  
);
```

参量

value

[in] 以" yyyy.mm.dd hh:mi " 为标本的字符串形式

返回值

[datetime](#) 型值自01.01.1970起包括秒的总数。

StringFormat

该函数格式包括参量和字符串的返回

```
string StringFormat(  
    string format,      // 带有格式描述的字符串  
    ...               // 参量  
);
```

Parameters

format

[in] 字符串包含格式化办法，格式化规则与 [PrintFormat](#) 函数相同。

...

[in] 参量，被逗号分开。

返回值

字符串。

另见

[打印格式](#), [双精度到字符串](#), [颜色到字符串](#), [时间到字符串](#)

数学函数

数学函数和三角函数

函数	功能
MathAbs	返回指定数值的绝对值（绝对值）
MathArccos	返回x弧度的反余弦
MathArcsin	返回x弧度的反正弦
MathArctan	返回x弧度的反正切
MathCeil	从上面返回最靠近的整数数值
MathCos	返回数字余弦
MathExp	返回数字指数
MathFloor	从下面返回最靠近的整数数值
MathLog	返回自然对数
MathMax	返回两个数值的最大值
MathMin	返回两个数值的最小值
MathMod	两个数值相除后返回实余数
MathPow	从基数升到额定输出
MathRand	0-32767范围返回随机值
MathRound	四舍五入到最近整数值
MathSin	返回正弦数
MathSqrt	返回平方根
MathSrand	设置开始点生成系列随机数
MathTan	返回正切数
MathIsValidNumber	检测实数真实型

MathAbs

函数返回特定数值的绝对值（绝对值）

```
double MathAbs(  
    double value    // 数值  
);
```

参量

value
[in] 数值

返回值

双精度类型值多于或等于0

注释

取代MathAbs() 函数，可以使用 fabs() 。

MathArccos

函数在0到 弧度返回x。

```
double MathArccos(  
    double val      // -1<val<1  
);
```

参量

val

[in] *val*值在-1到1之间，计算反余弦

返回值

编号幅度反余弦，如果*val*小于-1或者大于1，函数返回NaN（不确定值）。

注释

取代MathArccos() 函数可以使用 [acos\(\)](#) 。

另见

[真实型（双精度，浮点）](#)

MathArcsin

函数余弦x在- $\pi/2$ 到 $\pi/2$ 范围之内

```
double MathArcsin(  
    double val      // -1<值<1  
);
```

参量

val

[in] val值在-1到1之间，计算余弦值

返回值

val数的余弦在- $\pi/2$ 到 $\pi/2$ 之间，如果val小于-1，或者大于1，函数返回值是NaN（不确定值）

注释

取代MathArcsin() 函数，可以使用 [asin\(\)](#) 。

另见

[真实型（双精度，浮点）](#)

MathArctan

函数返回反正切 x ，如果 x 等于0，函数返回0

```
double MathArctan(  
    double value    // 正切  
);
```

参量

value

[in] 数字代表正切

返回值

MathArctan返回值在 $-\pi/2$ 到 $\pi/2$ 之间。

注释

代替MathArctan() 函数，可以使用 [atan\(\)](#) 。

MathCeil

函数从上返回最接近的整数值。

```
double MathCeil(  
    double val      // 数字  
);
```

参量

val
[in] 数字值

返回值

数字值代表最少总数超出或者等于val。

注释

取代MathCeil() 函数，可以使用 [ceil](#)() 。

MathCos

函数返回余弦角。

```
double MathCos(  
    double value    // 数字  
);
```

参量

value
[in] 角度

返回值

双精度值在-1到1之间。

注释

取代MathCos() 函数可以使用 `cos()` 。

MathExp

函数返回e到d的值。

```
double MathExp(  
    double value    // 数字 e 的功能  
);
```

参量

value

[in] 指定数字功率

返回值

许多双精度函数，函数返回INF（无限大），在缺少MathExp时返回0

注释

取代MathExp() 可以使用 [exp\(\)](#) .

另见

[真实型（双精度，浮点）](#)

MathFloor

函数返回下面最接近的整数值。

```
double MathFloor(  
    double val    // 数字  
);
```

参量

val
[in] 数值。

返回值

数字值代表最大整数少于或者等于val。

注释

取代MathFloor() 可以使用 [floor\(\)](#) 。

MathLog

函数返回一组自然对数。

```
double MathLog(  
    double val    // 取对数的数值  
);
```

参量

val

[in] 可以找到对数值。

返回值

如果成功出现自然对数 val ，如果 val 是负的，函数返回NaN(未定值) 。如果 val 等于0，函数返回INF(无限)。

注释

取代MathLog() 可以使用 [log\(\)](#) 。

另见

[真实型 \(双精度, 浮点\)](#)

MathLog

以10为底返回对数。

```
double MathLog10(  
    double val      // 取对数的数值  
);
```

参量

val

[in] 数字值是计算的常用对数。

返回值

成功了是常用对数，如果val是负的，函数返回NaN(待定值) ，如果val等于0，函数返回INF（无限大）

注释

取代MathLog10() 可以使用 [log10\(\)](#) 。

另见

[真实型（双精度，浮点）](#)

MathMax

函数返回最大的量个值。

```
double MathMax(  
    double value1,    // 第一值  
    double value2     // 第二值  
);
```

参量

value1

[in] 第一个数字值。

value2

[in] 第二个数字值。

返回值

两个值中的最大值。

注释

取代MathMax() 可以使用 [fmax\(\)](#)。函数 [fmax\(\)](#) , [fmin\(\)](#) , [MathMax\(\)](#) , [MathMin\(\)](#) 是整型的, 而非铸字的双精度类型。

如果不同类型参量通过函数传递, 副类型参量自动[转](#)到主类型。该类型返回值相当于主类型返回值。

如果相同数据类型传递, 不用执行铸字。

MathMin

函数返回两个值中的最小值。

```
double MathMin(  
    double value1,    // 第一值  
    double value2     // 第二值  
);
```

参量

value1

[in] 第一数值。

value2

[in] 第二数值

返回值

两值中最小的。

注释

取代MathMin() 可以使用 [fmin\(\)](#)。函数 fmax() , [fmin\(\)](#) MathMax() , [MathMin\(\)](#) 是整型 , 而非铸字的双精度类型。

如果是不同类型参量传递到函数中, 副类型参量自动 [转](#) 到主类型。返回值类型与主类型相同。

如果数据类型传递, 不需要执行铸字。

MathMod

函数返回两个数字相除的余数。

```
double MathMod(  
    double value,      // 股息值  
    double value2      // 除数值  
);
```

参量

value

[in] 被除数值。

value2

[in] 除数值。

返回值

MathMod函数计算真值 f of val / y such that $val = i * y + f$, f 是整数, f 有与 val 相同的符号, f 的绝对值小于 y 的绝对值。

注释

取代MathMod() 可以使用 [fmod\(\)](#) 。

MathPow

函数增加基数的整个功率。

```
double MathPow(  
    double base,           // 基数  
    double exponent        // 指数值  
);
```

参量

base

[in] 基数。

exponent

[in] 指数值

返回值

基数值提升到特定功率。

注释

取代MathPow() 可以使用 [pow\(\)](#) 。

MathRand

在0到32767之间返回随机整数。

```
int MathRand();
```

返回值

整数值在0到32767之间。

注释

在第一次调用前，调用 [MathSrand](#) 建立初始状态伪随机数是有必要的。

注释

取代MathRand() 可以使用 [rand](#)() 。

MathRound

函数四舍五入返回最近的整数值。

```
double MathRound(  
    double value    // 全值  
);
```

参量

value

[in] 四舍五入前的数值。

返回值

四舍五入到最近的整数值。

注释

取代MathRound() 可以使用 [round\(\)](#) .

MathSin

返回特定角的正弦。

```
double MathSin(  
    double value    // 弧度参数  
);
```

参量

value
[in] 角的弧度。

返回值

测量角的正弦，返回在-1到1之间的值。

注释

取代MathSin() 可以使用 [sin\(\)](#) 。

MathSqrt

返回数字的平方根。

```
double MathSqrt(  
    double value    // 正值  
);
```

参量

value
[in] 正值

返回值

平方根值，值是负的，MathSqrt返回NaN(待定值) 。

注释

取代MathSqrt() 可以使用 [sqrt\(\)](#) 。

另见

[真实型（双精度，浮点）](#)

MathSrand

建立伪随机整数产生序列出发点。

```
void MathSrand(  
    int seed    // 初始化数字  
);
```

参量

seed

[in] 随机数开始数字的排列。

返回值

没有返回值。

注释

再初始化的发生（例如，发生器到先前的初始化状态），有必要使用1值当做正在初始化的参量。初始数字任意值建立随机开始点的发生器。

[MathRand](#) 返回接连产生的伪随机数，在任意调用MathSrand之前的相同序列调用MathRand，参量为1。

取代MathSrand() 使用 [srand\(\)](#) .

MathTan

函数返回正切数。

```
double MathTan(  
    double rad    // 弧度参数  
);
```

参量

rad
[in] 弧度角

返回值

rad正切，如果rad大于或等于263，或者小于或等于-263，丢掉有意值，在此情况下，函数返回不确定。

注释

取代MathTan() 可以使用 [tan\(\)](#) .

另见

[真实型（双精度，浮点）](#)

MathIsValidNumber

检测实数的正确性。

```
bool MathIsValidNumber(  
    double number    // 检测数  
);
```

参量

number

[in] 检测数值。

返回值

返回真值，如果检测值是可以接受的实数。如果检测值是正无穷或者负无穷大，或者没有数字（NaN），函数返回错误值。

示例：

```
double abnormal=MathArcsin(2.0);  
if(!MathIsValidNumber(abnormal)) Print("Attention! MathArcsin(2.0) = ",abnormal);
```

另见

[真实型（双精度，浮点）](#)

字符串函数

用来与 [字符串](#) 型数据一起执行的函数。

函数	功能
StringAdd	在给出子连接上添加字符串
StringBufferLen	为字符串返回分配式缓冲器的大小
StringCompare	比较两个字符串并且如果第一个字符串大于第二个返回1；0-如果两个字符串相等；-1（负一）-如果第一个字符串小于第二个
StringConcatenate	实现一串参数传递
StringFill	通过挑选出来的交易品种填满指定字符串
StringFind	在字符串里搜索子串
StringGetCharacter	在规定字符串位置返回数字值
StringInit	通过指定交易品种初始化字符串并提供指定字符串长度
StringLen	在字符串里返回交易品种数字
StringReplace	
StringSetCharacter	在指定位置返回复制后的交易品种的改变值
StringSubstr	从指定位置提取文本串
StringToLower	通过存储单位把所有交易品种中已选的字符串转为小写
StringToUpper	通过存储单位把所有交易品种中已选的字符串转为资本
StringTrimLeft	在字符串的左边切断线路供应字符，空间和标号
StringTrimRight	在字符串的右边切断线路供应字符，空间和标号

StringAdd

在字符串末尾添加子串。

```
bool StringAdd(
    string& string_var,      // 将要添加内容的字符串
    string add_substring     // 被添加的字符串
);
```

参量

string_var

[in][out] 字符串，添加另一组。

add_substring

[in] 在源字符串末尾添加i

返回值

成功返回true, 否则是false, 为了获得 [错误代码](#)，可以调用 [GetLastError\(\)](#) 函数。

注释

与捆绑字符串所需的额外操作相比，StringAdd() add函数更快，并能节省空间。

示例：

```
void OnStart()
{
    string a="a",b="b",c;
//--- 第一函数
    int start=GetTickCount(),stop;
    long i;
    for(i=0;i<length;i++)
    {
        c=a+b;
    }
    stop=GetTickCount();
    Print("time for 'c = a + b' = ",(stop-start)," milliseconds, i = ",i);

//--- 第二函数
    start=GetTickCount();
    for(i=0;i<length;i++)
    {
        StringAdd(a,b);
    }
    stop=GetTickCount();
    Print("time for 'StringAdd(a,b)' = ",(stop-start)," milliseconds, i = ",i);

//--- 第三函数
    start=GetTickCount();
    a="a"; // 重新初始化变量 a
```

```
for(i=0;i<length;i++)
{
    int k=StringConcatenate(c,a,b);
}
stop=GetTickCount();
Print("time for 'StringConcatenate(c,a,b)' = ",(stop-start)," milliseconds, i = ",
}
```

另见

[StringConcatenate](#)

StringBufferLen

该函数为字符串返回分离式缓冲器的大小。

```
int StringBufferLen(  
    string string_var    // 字符串  
)
```

参量

string_var
[in] 字符串.

返回值

0值表示字符串是恒量，缓冲器大小不改变。-1表示字符串属于客户端，缓冲器内容修正拥有不确定因素。

注释

最小缓冲器大小是16。

示例：

```
void OnStart()  
{  
    long length=1000;  
    string a="a",b="b";  
    //---  
    long i;  
    Print("before: StringBufferLen(a) = ",StringBufferLen(a),  
        "   StringLen(a) = ",StringLen(a));  
    for(i=0;i<length;i++)  
    {  
        StringAdd(a,b);  
    }  
    Print("after: StringBufferLen(a) = ",StringBufferLen(a),  
        "   StringLen(a) = ",StringLen(a));  
}
```

另见

[StringAdd](#), [StringInit](#), [StringLen](#), [StringFill](#)

StringCompare

该函数是比较两个字符串并且以整数形式返回比较结果。

```
int StringCompare(
    const string& string1,           // 比较的第一个字符串
    const string& string2,           // 比较的第二个字符串
    bool case_sensitive=true         // 为比较选择大小写敏感模式
);
```

参量

string1

[in] 第一个字符串。

string2

[in] 第二个字符串。

case_sensitive=true

[in] 大小写敏感模式。如果是true，则 "A">"a"。如果是false，则"A"="a"。默认情况下值等于 true。

返回值

- -1 (负一)，如果 string1<string2
- 0 (零)，如果 string1=string2
- 1 (一)，如果 string1>string2

注释

字符串是字符与字符相比较，字符根据当前代码页以字母表的顺序比较。

示例：

```
void OnStart()
{
    //--- 哪个更大 - apple or home?
    string s1="Apple";
    string s2="home";

    //--- 区分大小写的比较
    int result1=StringCompare(s1,s2);
    if(result1>0) PrintFormat("Case sensitive comparison: %s > %s",s1,s2);
    else
    {
        if(result1<0) PrintFormat("Case sensitive comparison: %s < %s",s1,s2);
        else PrintFormat("Case sensitive comparison: %s = %s",s1,s2);
    }

    //--- 不区分大小写的比较
    int result2=StringCompare(s1,s2,false);
    if(result2>0) PrintFormat("Case insensitive comparison: %s > %s",s1,s2);
    else
    {
        if(result2<0) PrintFormat("Case insensitive comparison: %s < %s",s1,s2);
        else PrintFormat("Case insensitive comparison: %s = %s",s1,s2);
    }
}
/* 结果:
```

区分大小写的比较: Appl e < home
不区分大小写的比较: Appl e < home

```
*/  
}
```

另见

[字符串类型](#), [CharToString\(\)](#), [ShortToString\(\)](#), [StringToCharArray\(\)](#), [StringToShortArray\(\)](#), [StringGetCharacter\(\)](#), [代码页的使用](#)

StringConcatenate

数据的字符串形式通过并且返回常规字符串大小。 参量可以为任意类型。通过参量的总数不得少于2字符或超过64个字符。

```
int StringConcatenate(  
    string& string_var,    // 要形成的字符串  
    void argument1         // 任何简单类型的第一参量  
    void argument2         // 任何简单类型的第二参量  
    ...                    // 任何简单类型的下一个参量  
);
```

参量

string_var

[in][out] 字符串作为串联结果能够形成。

argumentN

[in] 逗号分离值，从2-63任意简单类型参量。

返回值

返回字符串长度，通过参量串联转变成字符串类型而形成，根据与 [Print\(\)](#) 和 [Comment\(\)](#) 相同的规则把参量转变成字符串。与打印不同，结果字符串串联参量并不能被空间分开。

注释

与捆绑字符串使用的额外操作相比，StringConcatenate() 更快，并节省内存，因为他们不需要使用 [字符串](#) 类型的临时变量。

另见

[StringAdd](#)

StringFill

通过指定交易品种填充所选字符串。

```
bool StringFill(  
    string&    string_var,      // 要添加的字符串  
    ushort    character        // 添加字符串的交易品种  
);
```

参量

string_var

[in][out] 字符串，通过所选交易品种填充。

character

[in] 交易品种，字符串填充。

返回值

如果成功返回true，否则是false，获得 [错误代码](#) 调用 [GetLastError\(\)](#)。

注释

正确填充字符串表示交易品种直接插入字符串而无需使用新字符串添加或复制的过渡性操作，这样缩短的操作时间。

示例：

```
void OnStart()  
{  
    string str;  
    StringInit(str,20,'_');  
    Print("str = ",str);  
    StringFill(str,0);  
    Print("str = ",str," : StringBufferLen(str) = ", StringBufferLen(str));  
}  
// 结果  
//   str = _____  
//   str =   : StringBufferLen(str) = 20  
//
```

另见

[StringBufferLen](#) , [StringLen](#) , [StringInit](#)

StringFind

在字符串中搜索子字符串。

```
int StringFind(  
    string  string_value,      // 进行搜索的字符串  
    string  match_substring,   // 搜索内容  
    int     start_pos=0       // 搜索开始位置  
);
```

参量

string_value

[in] 字符串，在产生搜索中使用。

match_substring

[in] 寻找子串。

start_pos=0

[in] 从搜索开始位置索引。

返回值

如果未找到子字符串，从搜索子字符串开始返回字符串中的位置，或是 -1。

StringGetCharacter

从字符串指定位置返回交易品种的值。

```
ushort StringGetCharacter(  
    string  string_value,    // 字符串  
    int     pos             // 字符串中交易品种位置  
);
```

参量

string_value

[in] 字符串

pos

[in] 字符串中交易品种的位置，能够从0到 [StringLen\(text\)](#) -1.

返回值

交易品种代码或错误为0，获得 [错误代码](#) 函数调用 [GetLastError\(\)](#).

StringInit

通过指定交易品种初始化一组字符串并提供指定字符串大小。

```
bool StringInit(
    string&    string_var,      // 要初始化的字符串
    int        new_len=0,       // 初始化后所需要的字符串长度
    ushort     character=0      // 填充字符串的交易品种
);
```

参量

string_var

[in][out] 字符串应该初始化和非初始化。

new_len=0

[in] 在初始化结束后的字符串长度，如果长度是0，就没有进行字符串初始化，例如，清除字符串的缓冲器，将缓冲器的地址调到0位置。

character=0

[in] 将交易品种填充字符串。

返回值

如果成功返回，显示true，否则是false，获得 [错误代码](#) 函数调用 [GetLastError\(\)](#)。

注释

如果字符是0，长度new_len>0，固定位置长度的字符串缓冲器会分散，并以0填满，字符串长度等于0，因为全部缓冲器通过字符串终端填满。

示例：

```
void OnStart()
{
    //---
    string str;
    StringInit(str,200,0);
    Print("str = ",str,": StringBufferLen(str) = ",
        StringBufferLen(str)," StringLen(str) = ",StringLen(str));
}
/* Result
str = : StringBufferLen(str) = 200   StringLen(str) = 0
*/
```

另见

[StringBufferLen](#) -[StringLen](#)

StringLen

在字符串中返回交易品种数字。

```
int StringLen(  
    string string_value    // 字符串  
);
```

参量

string_value
[in] 字符串计算长度。

返回值

在字符串中交易品种数字不能以0结尾。

StringReplace

```
int StringReplace(
    string&      str,          // строка, в которой будет осуществляться замена
    const string find,        // искомая подстрока
    const string replacement   // подстрока, которая будет вставлена в найденны
);
```

str

[in][out]

find

[in]

replacement

[in]

-1.

[GetLastError\(\)](#).

(

0.

) ,

str *find*

(

[StringInit\(\)](#)) .

:

```
string text="The quick brown fox jumped over the lazy dog.";
int replaced=StringReplace(text,"quick","slow");
replaced+=StringReplace(text,"brown","black");
replaced+=StringReplace(text,"fox","bear");
Print("Replaced: ", replaced,". Result=",text);

// Результат
// Replaced: 3. Result=The slow black bear jumped over the lazy dog.
//
```

[StringSetCharacter\(\)](#), [StringSubstr\(\)](#)

StringSetCharacter

在指定位置上返回字符串上复制的改变字节。

```
bool StringSetCharacter(
    string&    string_var,      // 字符串
    int        pos,             // 位置
    ushort     character        // 字符
);
```

参量

string_var
[in][out] String.

pos
[in] 在字符串上的位置特性，从0开始到 [StringLen](#)（文本）

character
[in] 交易品种代码Unicode。

注释

如果销售点少于 [字符串长度](#) 并且交易品种代码值是0，切断字符串（但是 [缓冲区大小](#)，为未改变的字符串类型分散），字符串长度与销售点相同。

如果pos与字符串长度相同，指定交易品种会添加到字符串末尾，长度会一个接一个扩大。

示例：

```
void OnStart()
{
    string str="0123456789";
    Print("before: str = ",str," ,StringBufferLen(str) = ",
        StringBufferLen(str)," StringLen(str) = ",StringLen(str));
    //--- 在中心添加零值
    StringSetCharacter(str,6,0);
    Print(" after: str = ",str," ,StringBufferLen(str) = ",
        StringBufferLen(str)," StringLen(str) = ",StringLen(str));
    //--- 在末尾添加交易品种
    int size=StringLen(str);
    StringSetCharacter(str,size,'+');
    Print("addition: str = ",str," ,StringBufferLen(str) = ",
        StringBufferLen(str)," StringLen(str) = ",StringLen(str));
}
/* Result
before: str = 0123456789 ,StringBufferLen(str) = 0 StringLen(str) = 10
after: str = 012345 ,StringBufferLen(str) = 16 StringLen(str) = 6
addition: str = 012345+ ,StringBufferLen(str) = 16 StringLen(str) = 7
*/
```

另见

[StringBufferLen](#), [StringLen](#), [StringFill](#), [StringInit](#)

StringSubstr

子串的摘要从指定位置的文本字符串开始。

```
string StringSubstr(  
    string  string_value,    // 字符串  
    int     start_pos,      // 初始位置  
    int     length=-1       // 提取的字符串的长度  
);
```

参量

string_value

[in] 字符串提取子串。

start_pos

[in] 子串的初始位置，从0到 [StringLen](#)(文本) -1。

length=-1

[in] 提取子串的长度，如果参量值是-1或者没有建立参量，子串会从指定位置提取直到字符串结束。

返回值

复制提取的子串，如果可能的话，否则返回空字符串

StringToLower

所选字符串的所有交易品种通过存储单元转变成小写字母。

```
bool StringToLower(  
    string& string_var    // 要处理的字符串  
);
```

参量

string_var
[in][out] 字符串。

返回值

如果成功，返回值是true, 否则是false，获得 [错误代码](#) 调用 [GetLastError\(\)](#)。

StringToUpper

所有选择字符串中的交易品种通过存储单元转换成大写字母。

```
bool StringToUpper(  
    string& string_var    // 要处理的字符串  
);
```

参量

string_var
[in][out] 字符串

返回值

成功返回真值，否则是失败值，获得 [错误代码](#) 调用 [GetLastError\(\)](#)。

StringTrimLeft

函数为字符串换行，在第一位的实意交易品种后，是字符串左边的字符和标号，字符串适当修改。

```
int StringTrimLeft(  
    string& string_var    // 要剪切的字符串  
);
```

参量

string_var

[in][out] 字符串从左边剪切。

返回值

返回剪切交易品种数量。

StringTrimRight

函数为字符串换行，在最后面的实意交易品种后，是字符串右边的字符和标号，字符串适当修改。

```
int StringTrimRight(  
    string& string_var    // 要剪切的字符串  
);  
);
```

参量

string_var

[in][out] 字符串从右边剪切。

返回值

返回剪切交易品种数量。

日期和时间

函数组与 [日期时间](#) 类型一起工作（整型代表已使用的秒数从1970.1.1.零时开始）。

函数	功能
TimeCurrent	日期时间格式返回服务器最后一个可知时间（最后一个报价收据时间）
TimeTradeServer	返回交易服务器当前计算时间
TimeLocal	以日期时间格式返回本地计算机时间
TimeGMT	通过客户端运行的计算机本地时间以日期时间格式返回GMT日节省时间。
TimeDaylightSavings	返回日节省时间切换符号
TimeGMTOffset	以秒计算返回当前GMT和计算机本地的不同，也包括夏令时切换
TimeToStruct	日期时间值转换到MqlDate Time结构类型变量
StructToTime	MqlDate Time结构类型变量转化到日期时间值

TimeCurrent

返回最后访问的服务器时间，最后在"市场报价"窗口中选择交易品种接收的引用时间。在 [OnTick\(\)](#) 处理器中，函数返回接收的处理器时间。在其他情况下（例如，调用 [handlers](#) [OnInit\(\)](#)，[OnDeinit\(\)](#)，[OnTimer\(\)](#) 等等），对于在"市场报价"窗口中的交易品种常量有 [最后一个报价收据时间](#)，时间在窗口标题上显示。时间值在交易服务器上形成并不能依靠时间设置电脑。有两种变量函数。

无参量调用

```
datetime TimeCurrent();
```

调用MqlDateTime类型参量

```
datetime TimeCurrent(  
    MqlDateTime& dt_struct    // 结构类型变量  
);
```

参量

`dt_struct`
[out] [MqlDateTime](#) 结构类型变量

返回值

[日期时间](#) 类型值

注释

如果 [MqlDateTime](#) 结构类型变量以参量传递，就会因此填满。

TimeTradeServer

返回交易服务器计算出的当前时间，不像[TimeCurrent\(\)](#)，时间值的计算在客户端中执行并依据电脑里的时间设置。有两种常量函数。

无参量调用

```
datetime TimeTradeServer();
```

调用MqlDateTime类型参量

```
datetime TimeTradeServer(  
    MqlDateTime& dt_struct // 结构类型变量  
);
```

参量

dt_struct
[out] [MqlDateTime](#) 结构类型变量

返回值

[日期时间](#) 类型值

注释

如果 MqlDateTime 如果结构类型变量以参量传递，会因此填满。

TimeLocal

返回客户端运行时电脑上的当地时间，有两种变量函数。

无参量调用

```
datetime TimeLocal();
```

调用MqlDateTime类型参量

```
datetime TimeLocal(  
    MqlDateTime& dt_struct    // 结构类型变量  
);
```

参量

dt_struct

[out] [MqlDateTime](#) 结构类型变量

返回值

[日期时间](#) 类型值

注释

如果 MqlDateTime 结构类型变量以参量传递，就会填满。

TimeGMT

返回GMT，该计算考虑到当客户端运行时电脑上当地时间的DST转换，有两种变量函数。

无参量调用

```
datetime TimeGMT();
```

调用MqlDateTime类型参量

```
datetime TimeGMT(  
    MqlDateTime& dt_struct    // 结构类型变量  
);
```

参量

dt_struct

[out] [MqlDateTime](#) 结构类型变量。

返回值

[日期时间](#) 类型值

注释

如果 MqlDateTime 结构类型变量以参量传递，就会填满。

TimeDaylightSaving

返回以秒为单位的正确时间，转变成夏季时间，取决于电脑上的时间设置。

```
int TimeDaylightSavings();
```

返回值

如果转换成冬季（标准的）时间，返回0。

TimeGMTOffset

返回在GMT时间和当前电脑之间的不同，考虑到转换成冬季或者夏季时间，取决于电脑上设置的时间。

```
int TimeGMTOffset();
```

返回值

整型值，代表在当前电脑和 [GMT时间](#) 之间的不同。

TimeToStr

转变时间类型值（数字从1970.01.01以秒开始）到结构变量 [MqlDateTime](#) 中。

```
void TimeToStr(  
    datetime      dt,           // 日期和时间  
    MqlDateTime&  dt_struct     // 采用值的结构  
);
```

参量

dt

[in] 转变数据值。

dt_struct

[out] 结构类型MqlDateTime变量

返回值

无返回值。

StructToTime

转变结构变量 `MqlDateTime` 到 [日期时间](#) 类型值，然后返回结果值。

```
datetime StructToTime(  
    MqlDateTime$ dt_struct    // 日期和时间结构  
);
```

参量

dt_struct

[in] 结构类型变量 `MqlDateTime`。

返回值

日期时间型的值包括从01.01.1970起的每秒钟数据。

账户信息

返回当前账户参量的函数。

函数	功能
AccountInfoDouble	返回相应账户属性的双精度值
AccountInfoInteger	返回相应账户属性的整数类型值（布尔，整型或者长整型）
AccountInfoString	返回相应账户属性的字符串类型值

AccountInfoDouble

返回对应账户财产值。

```
double AccountInfoDouble(  
    int property_id // 属性标识符  
);
```

参量

property_id

[in] 属性指标，值可以是[ENUM_ACCOUNT_INFO_DOUBLE](#)其中一个

返回值

[双精度](#) 类型值

示例：

```
void OnStart()  
{  
    // --- 显示 AccountInfoDouble() 函数中所有有效信息  
    printf("ACCOUNT_BALANCE = %G", AccountInfoDouble(ACCOUNT_BALANCE));  
    printf("ACCOUNT_CREDIT = %G", AccountInfoDouble(ACCOUNT_CREDIT));  
    printf("ACCOUNT_PROFIT = %G", AccountInfoDouble(ACCOUNT_PROFIT));  
    printf("ACCOUNT_EQUITY = %G", AccountInfoDouble(ACCOUNT_EQUITY));  
    printf("ACCOUNT_MARGIN = %G", AccountInfoDouble(ACCOUNT_MARGIN));  
    printf("ACCOUNT_FREEMARGIN = %G", AccountInfoDouble(ACCOUNT_FREEMARGIN));  
    printf("ACCOUNT_MARGIN_LEVEL = %G", AccountInfoDouble(ACCOUNT_MARGIN_LEVEL));  
    printf("ACCOUNT_MARGIN_SO_CALL = %G", AccountInfoDouble(ACCOUNT_MARGIN_SO_CALL));  
    printf("ACCOUNT_MARGIN_SO_SO = %G", AccountInfoDouble(ACCOUNT_MARGIN_SO_SO));  
}
```

另见

[SymbolInfoDouble](#) , [SymbolInfoString](#) , [SymbolInfoInteger](#) , [PrintFormat](#)

AccountInfoInteger

返回账户属性值。

```
long AccountInfoInteger(
    int property_id    // 属性标识符
);
```

参量

property_id

[in] 属性标识符，值可以是 [ENUM_ACCOUNT_INFO_INTEGER](#) 值中一个。

返回值

[长整型](#) 类型值

整数

属性可以是 [布尔](#)、[整型](#) 或者 [长整型](#) 类型中的一个。

示例：

```
void OnStart()
{
//--- 显示 AccountInfoInteger() 函数中所有有效信息
printf("ACCOUNT_LOGIN = %d", AccountInfoInteger(ACCOUNT_LOGIN));
printf("ACCOUNT_LEVERAGE = %d", AccountInfoInteger(ACCOUNT_LEVERAGE));
bool thisAccountTradeAllowed=AccountInfoInteger(ACCOUNT_TRADE_ALLOWED);
bool EATradeAllowed=AccountInfoInteger(ACCOUNT_TRADE_EXPERT);
ENUM_ACCOUNT_TRADE_MODE tradeMode=AccountInfoInteger(ACCOUNT_TRADE_MODE);
ENUM_ACCOUNT_STOPOUT_MODE stopOutMode=AccountInfoInteger(ACCOUNT_MARGIN_SO_MODE);

//--- 通知完成交易操作的可能性
if(thisAccountTradeAllowed)
    Print("Trade for this account is permitted");
else
    Print("Trade for this account is prohibited!");

//--- 找出是否可能通过EA交易进行这个账户的交易
if(EATradeAllowed)
    Print("Trade by Expert Advisors is permitted for this account");
else
    Print("Trade by Expert Advisors is prohibited for this account!");

//--- 找出账户类型
switch(tradeMode)
{
    case(ACCOUNT_TRADE_MODE_DEMO):
        Print("This is a demo account");
        break;
```

```
case (ACCOUNT_TRADE_MODE_CONTEST):
    Print("This is a competition account");
    break;
default: Print("This is a real account!");
}

//--- 找出止损离场水平设置模式
switch (stopOutMode)
{
case (ACCOUNT_STOPOUT_MODE_PERCENT):
    Print("The StopOut level is specified percentage");
    break;
default: Print("The StopOut level is specified in monetary terms");
}
}
```

另见

[账户信息](#)

AccountInfoString

返回对应账户属性值。

```
string AccountInfoString(  
    int property_id    // 属性标识符  
);
```

参数

property_id

[in] 属性标识符，值可以是 [ENUM_ACCOUNT_INFO_STRING](#) 值中一个。

返回值

[字符串](#) 类型值

示例：

```
void OnStart()  
{  
    // --- 显示 AccountInfoString() 函数中所有有效信息  
    Print("The name of the broker = ", AccountInfoString(ACCOUNT_COMPANY));  
    Print("Deposit currency = ", AccountInfoString(ACCOUNT_CURRENCY));  
    Print("Client name = ", AccountInfoString(ACCOUNT_NAME));  
    Print("The name of the trade server = ", AccountInfoString(ACCOUNT_SERVER));  
}
```

另见

[账户信息](#)

启动检测

函数返回当前客户端现行状态的参量

函数	功能
GetLastError	返回上一个错误
IsStopped	如果mql5程序被令停止操作，返回true
UninitializeReason	返回无法初始化原因代码
TerminalInfoInteger	返回mql5程序环境相关属性的整数值
TerminalInfoString	返回mql5程序环境相关属性的字符串值
MQL5InfoInteger	返回运行mql5程序相关属性的整数值
MQL5InfoString	返回运行mql5程序相关属性的字符串值
Symbol	返回当前图表的交易品种的名称
Period	返回当前图表时间表
Digits	返回决定当前图表交易品种价格值的精确性的小数位数
Point	返回报价货币当前交易品种的点大小

GetLastError

返回 [_LastError](#) 系统变量目录。

```
int GetLastError();
```

返回值

在MQL5程序执行时，返回最后发生的[错误值](#)。

注释

在函数调用后，不重设 [_LastError](#) 目录。重设变量，需要调用 [ResetLastError\(\)](#)。

IsStopped

检测MQL5程序的强制关机。

```
bool IsStopped();
```

返回值

如果系统变量包括非0值，返回true，非零值录入 [__StopFlag](#)，如果MQL5程序命令完成操作，在此情况下，必须立即终止程序，否则系统在3秒后从外部强制完成。

UninitializeReason

返回代码 [无法初始化原因](#)。

```
int UninitializeReason();
```

返回值

在 [__UninitReason](#) 调用之前返回 [OnDeinit\(\)](#) 值，该值依靠导致无法初始化的原因。

TerminalInfoInteger

在MQL5程序环境中该函数返回一定值。

```
int TerminalInfoInteger(  
    int property_id    // 属性标识符  
);
```

参量

property_id

[in] 属性标识符，或许是计算式值 [ENUM_TERMINAL_INFO_INTEGER](#) 中一个。

返回值

整型值。

TerminalInfoString

在MQL5程序环境中该函数返回一定值，该属性一定是字符串型。

```
string TerminalInfoString(  
    int property_id    // 属性标识符  
);
```

参量

property_id

[in] 属性标识符，或许是计算式值 [ENUM_TERMINAL_INFO_STRING](#) 中一个。

返回值

字符串类型值。

MQL5InfoInteger

返回运行的MQL5程序的类似属性值。

```
int MQL5InfoInteger(  
    int property_id    // 属性标识符  
);
```

参量

property_id

[in] 属性标识符，可以是 [ENUM_MQL5_INFO_INTEGER](#) 值中的一个。

返回值

整型值。

MQL5InfoString

返回运行中的MQL5程序的类似属性值。

```
string MQL5InfoString(  
    int property_id    // 属性标识符  
);
```

参量

property_id

[in] 属性标识符，可以是 [ENUM_MQL5_INFO_STRING](#) 值中的一个。

返回值

字符串类型值。

Symbol

返回当前图表的交易品种名称。

```
string Symbol();
```

返回值

Symbol 系统变量的值，存储当前图表交易品种的名称。

Period

返回当前图表的时间表。

```
ENUM_TIMEFRAMES Period();
```

返回值

[__Period](#) 变量内容包括当前图表时间表的值，该值可以是 [ENUM_TIMEFRAMES](#) 计算式中的一个。

另见

[PeriodSeconds](#) - [图表时间表](#)， [日期和时间](#)， [对象可见性](#)

Digits

返回当前图表交易品种价格精确度的小数位数数量。

```
int Digits();
```

返回值

[_Digits](#) 变量值存储在小数位数数量中，觉得当前图表交易品种价格的精确度。

Point

在引用货币中返回当前交易品种大小点。

```
double Point();
```

返回值

[_Point](#) 变量值，在引用货币中存储当前交易品种的确值。

获取市场信息

函数用来接收关于市场规则的信息。

函数	功能
SymbolsTotal	返回有效交易品种数量（选择市场报价或者全部）
SymbolName	返回指定交易品种名称
SymbolSelect	市场报价窗口选择或者去除交易品种
SymbolIsSynchronized	检测程序端选择的交易品种的数据与交易服务器的数据是否 同步 。
SymbolInfoDouble	为相关属性返回交易品种双精度值
SymbolInfoInteger	为相关属性返回指定交易品种的整数型值（长整型，日期时间，整型或者布尔型）
SymbolInfoString	为相关属性返回指定交易品种的字符串类型值
SymbolInfoTick	返回 MqlTick 型变量中指定交易品种当前价格。
SymbolInfoSessionQuote	平日允许为指定交易品种接收指定报价期的起止时间
SymbolInfoSessionTrade	平日允许为指定交易品种接收指定交易期的起止时间
MarketBookAdd	提供选定交易品种市场深度开盘，签署DOM更改的接收通告
MarketBookRelease	提供选定交易品种市场深度收盘，取消DOM更改的接收通告的签署。
MarketBookGet	返回包括指定交易品种市场深度记录的结构数组 MqlBookInfo

交易品种总计

返回可用交易品种数量（在市场调查中选择）。

```
int SymbolsTotal(  
    bool selected // True - 只是市场报价中的交易品种  
);
```

参量

selected

[in] 请求模式，可以是true或false。

返回值

如果“选择”参量是true,函数返回字符的数量就在市场窗口中选择，如果值是false，返回所有交易品种的数量。

SymbolName

返回交易品种名称。

```
string SymbolName(  
    int    pos,           // 列表中的数字  
    bool   selected       // true - 只是市场报价中的交易品种  
);
```

参量

pos

[in] 一个教育品种的订单编号。

selected

[in] 请求方式，如果值是true,该交易品种从其在市场观测的选择中找出，如果值是false，交易品种从总目录中找出。

返回值

包含交易品种名称的字符串的值。

SymbolSelect

在市场观测中选择一个交易品种，或者从窗口中移动一个交易品种。

```
bool SymbolSelect(  
    string name,          // 交易品种名称  
    bool select           // 添加或者移除  
);
```

参量

name

[in] 交易品种名称。

select

[in]转换，如果值是false，交易品种从市场观测中删除，然后在窗口中选择一个交易品种，如果交易品种共图表是开放式的，交易品种就不能被删除，或者该交易品种有开放定位。

返回值

以防返回失败。

同步交易品种

该函数检验在客户端中选出来的交易品种数据与交易服务器中的数据是同步的。

```
bool SymbolIsSynchronized(  
    string name,          // 交易品种名称  
);
```

参量

name

[in] 交易品种名称。

返回值

如果数据是 [同步的](#)，返回true, 否则返回false。

另见

[SymbolInfoInteger](#), [组织数据接入](#)

SymbolInfoDouble

返回一个规定交易样品的类似特性，函数有两个变量。

1. 快速返回规定值。

```
double SymbolInfoDouble(
    string name,          // 交易品种
    int prop_id           // 属性标识符
);
```

2. 返回真值或失败值取决于函数操作是否成功，如果成功，属性值通过引用从最后的参量传递到接受变量中。

```
bool SymbolInfoDouble(
    string name,          // 交易品种
    int prop_id,          // 属性标识符
    double& double_var    // 这里假设属性值
);
```

参量

name

[in] 交易品种名称

prop_id

[in] 交易样品属性标识符，值可能是计算式 [ENUM_SYMBOL_INFO_DOUBLE](#) 中的一个。

double_var

[out] 双精度型变量接收需求属性的值

返回值

双精度型值。

示例：

```
void OnTick()
{
    //--- 从交易品种属性获得点差
    bool spreadfloat=SymbolInfoInteger(Symbol(),SYMBOL_SPREAD_FLOAT);
    string comm=StringFormat("Spread %s = %I64d points\r\n",
                             spreadfloat?"floating":"fixed",
                             SymbolInfoInteger(Symbol(),SYMBOL_SPREAD));

    //--- 现在自己计算点差
    double ask=SymbolInfoDouble(Symbol(),SYMBOL_ASK);
    double bid=SymbolInfoDouble(Symbol(),SYMBOL_BID);
    double spread=ask-bid;
    int spread_points=(int)MathRound(spread/SymbolInfoDouble(Symbol(),SYMBOL_POINT));
    comm=comm+"Calculated spread = "+(string)spread_points+" points";
    Comment(comm);
}
```

SymbolInfoInteger

返回一个规定交易样品的类似特质，该函数有两种变体

1. 快速返回属性值

```
long SymbolInfoInteger(
    string name,           // 交易品种
    int prop_id            // 属性标识符

);
```

2. 返回真值或失败值取决于函数是否成功执行，如果成功，属性值通过引用从最后的参量传递到接受变量中

```
bool SymbolInfoInteger(
    string name,           // 交易品种
    int prop_id,           // 属性标识符
    long& long_var         // 这里假设属性值

);
```

参量

name

[in] 交易样品名称

prop_id

[in] 交易样品属性标识符，值可以是计算式中 [ENUM_SYMBOL_INFO_INTEGER](#) 的一个。

long_var

[out] 长型变量接收要求变量的值。

返回值

长整型值

示例：

```
void OnTick()
{
//--- 从交易品种属性获得点差
    bool spreadfloat=SymbolInfoInteger(Symbol(),SYMBOL_SPREAD_FLOAT);
    string comm=StringFormat("Spread %s = %I64d points\r\n",
                             spreadfloat?"floating":"fixed",
                             SymbolInfoInteger(Symbol(),SYMBOL_SPREAD));

//--- 现在自己计算点差
    double ask=SymbolInfoDouble(Symbol(),SYMBOL_ASK);
    double bid=SymbolInfoDouble(Symbol(),SYMBOL_BID);
    double spread=ask-bid;
    int spread_points=(int)MathRound(spread/SymbolInfoDouble(Symbol(),SYMBOL_POINT));
    comm=comm+"Calculated spread = "+(string)spread_points+" points";
    Comment(comm);
}
```

SymbolInfoString

一个规定交易品种返回相对应的特性，该函数中有两个变体。

1. 立即返回属性值

```
string SymbolInfoString(  
    string name,           // 交易品种  
    int prop_id            // 属性标识符  
);
```

2. 返回true或者false，由该函数运行成功与否来决定，如果成功，在培育变量通过引用传递到最后常量中函数属性固定。

```
bool SymbolInfoString(  
    string name,           // 交易品种  
    int prop_id,           // 属性标识符  
    string& string_var     // 这里假设属性值  
);
```

参量

name

[in] 交易品种名称。

prop_id

[in] 一个交易品种属性标识符，值是 [ENUM_SYMBOL_INFO_STRING](#) 其中一个。

string_var

[out] 字符串类型变量接收需求属性的值。

返回值

字符串的值。

SymbolInfoTick

该函数返回在MqlTick 类型常量中的规定交易品种的当前价值。

```
bool SymbolInfoTick(  
    string      symbol,      // 交易品种名称  
    MqlTick&    tick         // 结构参考  
);
```

参量

symbol

[in] 交易品种名称。

book[]

[out] 连接 [MqlTick](#) 类型结构，显示当前价格和最后价格的即时更新。

返回值

如果成功，函数返回真值，否则返回失败

SymbolInfoSessionQuote

允许为规定交易样品和周期接收引用登陆的开始和结束时间。

```
bool SymbolInfoSessionQuote(  
    string          name,           // 交易品种名称  
    ENUM_DAY_OF_WEEK day_of_week,  // 一周中的每天  
    uint           session_index,   // 期指  
    datetime&      from,           // 期始时间  
    datetime&      to              // 期末时间  
);
```

参量

name

[in] 交易品种名称

ENUM_DAY_OF_WEEK

[in] 每周日期，计算式 [ENUM_DAY_OF_WEEK](#) 值、

uint

[in] 登陆的序列号，接收开始和结束的时间，以0开始登陆

from

[out] 从00小时00分钟以短时间开始登陆，返回值的日期可以忽略不计

to

[out] 从00小时00分钟以短时间结束登陆，返回值的日期可以忽略不计

返回值

如果接收到规定登陆数据的交易品种和每周日期，返回真值，否则返回false

另见

[交易品种属性](#)，[TimeToStruct](#)，[数据结构](#)

SymbolInfoSessionTrade

平日指定交易品种允许接收规定交易的起始和结束时间

```
bool SymbolInfoSessionTrade(  
    string          name,           // 交易品种名称  
    ENUM_DAY_OF_WEEK day_of_week,  // 一周中的每天  
    uint            session_index,  // 期指  
    datetime&       from,          // 期始时间  
    datetime&       to             // 期末时间  
);
```

参量

name

[in] 交易品种名称。

ENUM_DAY_OF_WEEK

[in] 每周日期，计算 [ENUM_DAY_OF_WEEK](#) 值

uint

[in] 一段登陆的数字，能够收到起始和结束时间，从0开始

from

[out] 从00小时00分钟以短时间开始登陆，返回值的日期可以忽略不计。

to

[out] 从00小时00分钟以短时间结束登陆，返回值的日期可以忽略不计。

返回值

如果接收到规定登陆数据的交易品种和每周日期，返回真值，否则返回false。

另见

[交易品种属性](#) - [TimeToStruct](#) - [数据结构](#)

MarketBookAdd

提供所选的交易品种的开盘市场深度信息，预定接收DOM转变的通知。

```
bool MarketBookAdd(  
    string symbol    // 交易品种  
);
```

参量

symbol

[in] 交易品种的名称，市场深度用于EA交易或脚本中。

返回值

如果成功开仓，返回true，否则是false。

整数

一般来说，该函数一定从 [OnInit\(\)](#) 函数中调用或者在分类构造函数中。为处理引用警报，在EA交易程序中必须包括 [OnBookEvent](#)函数(字符串& 交易品种) 。

另见

[市场深度结构](#)， [结构和类](#)

MarketBookRelease

提供所选交易品种的收盘市场报价信息，并取消预定接收DOM转变的通知。

```
bool MarketBookRelease(  
    string symbol    // 交易品种  
);
```

参量

symbol

[in] 交易品种名称。

返回值

如果成功关闭是真值，否则是错误值。

Note

通常，该函数需要调用[OnDeinit\(\)](#) 函数才能启用，如果在[OnInit\(\)](#) 函数中调用类似于 [MarketBookAdd\(\)](#) 函数，就会调用分类解构函数，从分类解构函数中调用类似于MarketBookAdd() 的函数。

另见

[市场深度结构](#)， [结构和类](#)

MarketBookGet

返回结构数组 [MqlBookInfo](#) 包括指定交易品种的市场报价记录。

```
bool MarketBookGet(  
    string      symbol,      // 交易品种  
    MqlBookInfo& book[]      // 参考数组  
);
```

参量

symbol

[in] 交易品种名称。

book[]

[in] 关于市场报价的记录。数组再分配给记录数量。如果 [动态数组](#) 没有再分配给操作内存，客户端会自行分配数组。

返回值

如果成功，返回true，否则是false。

整数

市场深度通过 [MarketBookAdd\(\)](#) 函数重启。

示例：

```
MqlBookInfo priceArray[];  
bool getBook=MarketBookGet(NULL,priceArray);  
if(getBook)  
{  
    int size=ArraySize(priceArray);  
    Print("MarketBookInfo for ",Symbol());  
    for(int i=0;i<size;i++)  
    {  
        Print(i+":",priceArray[i].price  
            +"    Volume = "+priceArray[i].volume,  
            " type = ",priceArray[i].type);  
    }  
}  
else  
{  
    Print("Could not get contents of the symbol DOM ",Symbol());  
}
```

另见

[市场深度结构](#)，[结构和类](#)

访问时间序列和指标数据

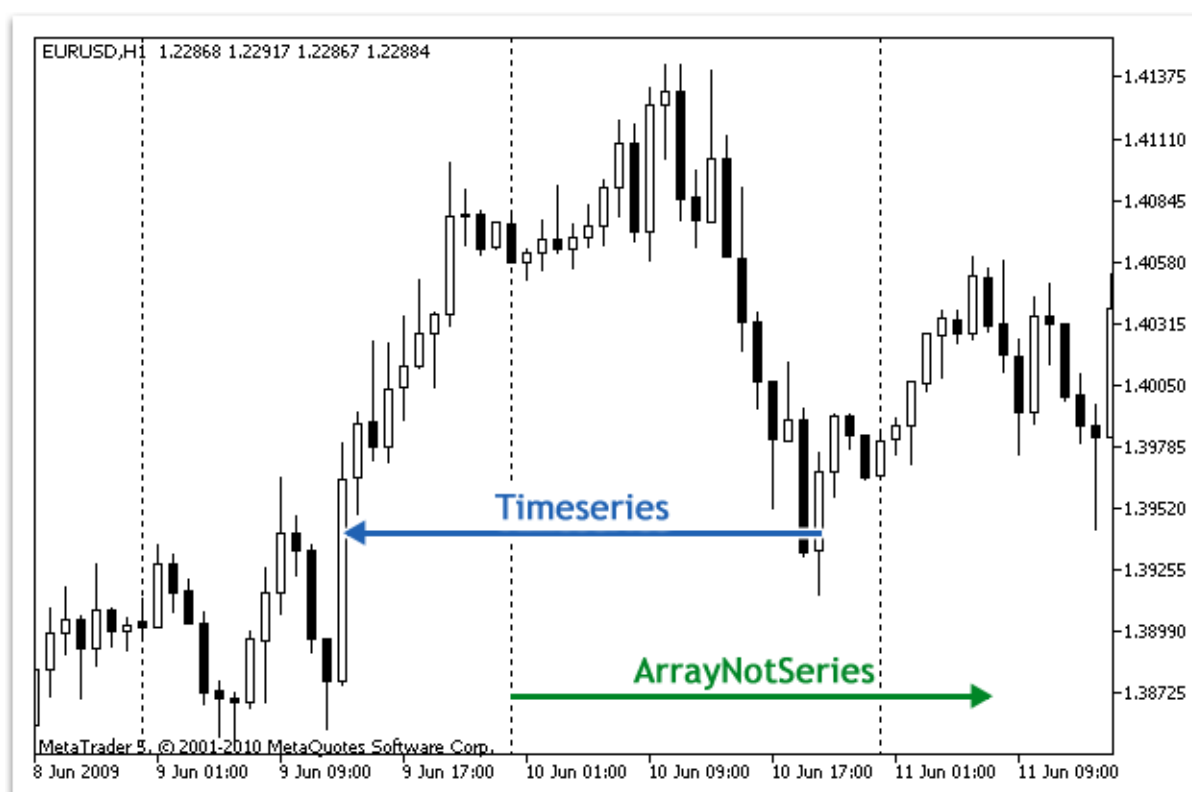
该组函数使用时序列和指标，时序列与通常的倒序的数据数组不同——时序列的元素从数组末尾到开头进行索引（大多从当前数据到最早数据），为了复制时序列值和指标数据，仅使用 [动态数组](#) 即可完成，因为复制函数为收到的值分配数组的必要大小。

这项规则有一个重要的例外情况：如果时序列和指标的值需要经常复制吗，例如，每次在EA交易中调用 [OnTick\(\)](#) 或者在指标中调用 [OnCalculate\(\)](#)，最好使用[静态分配的数组](#)，因为动态数组的内存分配操作需要额外时间，在测试和最佳化中会起到作用。

当使用函数访问时序列和指标的值时，要考虑索引方向，在 [数组和时间序列中索引方向](#) 章节中有所描述。

指标和时序列数据的访问应用不考虑要求数据是否准备完好（也就是所谓的 [异步访问](#)），这对于自定义指标的计算来说非常重要，因此，如果没有数据，函数 Copy...() 类型会立即返回错误值。然而，当形成EA交易和脚本的访问时，接收数据的尝试会短暂停顿，只是为时序列要求的指标计算值的下载提供必要的时间。

[组织访问数据](#) 章节在MT5客户端中描述接收、存储和需求价格数据的细节。



历史上在数组中接收价格数据从末尾开始，物理上，新的数据通常从数组末尾开始编写，但数组索引通常是0，时序列数组中的0索引表示当前字节中的数据，也就是，在时间表中未完成的时间间隔。

时间表中单一价格字节可以形成，在 [标准时间表](#) 中预先定义21个。

函数	功能
SeriesInfoInteger	返回历史资料信息

Bars	返回规定交易品种和周期的历史字节数量
BarsCalculated	错误情况下（还没有计算数据）返回指标缓冲区计算的数据量或者-1
IndicatorRelease	如果没有其他，移除指标处理器并释放指标计算模块。
IndicatorCreate	返回通过 MqlParam 参量类型数据创建的指定技术指标处理器。
CopyBuffer	从指定指标获得特定缓冲区数据到数组中。
CopyRates	将指定交易品种 比率 结构的历史数据和周期放在数组中。
CopyTime	将指定交易品种 开盘时间柱的历史数据和周期放在数组中。
CopyOpen	将指定交易品种 开盘价格柱的历史数据和周期放在数组中。
CopyHigh	将指定交易品种最大价格柱的历史数据和周期放在数组中。
CopyLow	将指定交易品种最小价格柱的历史数据和周期放在数组中。
CopyClose	将指定交易品种收盘价格柱的历史数据和周期放在数组中。
CopyTickVolume	指定交易品种订单交易量历史数据和周期放在数组中。
CopyRealVolume	指定交易品种交易量历史数据和周期放在数组中。
CopySpread	指定交易品种点差历史数据和周期放在数组中。

尽管通过 [ArraySetAsSeries\(\)](#) 函数可能像在时间序列一样建立 [数组](#) 接入元素，数组元素存储的可能是同一个——只有索引方向不同，为证明这个事实，提供如下示例：

```

datetime TimeAsSeries[];
//--- 如时间序列访问一样设置数组访问
ArraySetAsSeries(TimeAsSeries,true);
ResetLastError();
int copied=CopyTime(NULL,0,0,10,TimeAsSeries);
if(copied<=0)
{
    Print("The copy operation of the open time values for last 10 bars has failed")
    return;
}
Print("TimeCurrent =",TimeCurrent());
Print("ArraySize(Time) =",ArraySize(TimeAsSeries));
int size=ArraySize(TimeAsSeries);
for(int i=0;i<size;i++)
{
    Print("TimeAsSeries["+i+"] =",TimeAsSeries[i]);
}

datetime ArrayNotSeries[];
ArraySetAsSeries(ArrayNotSeries,false);
ResetLastError();
copied=CopyTime(NULL,0,0,10,ArrayNotSeries);
if(copied<=0)
{

```

```
Print("The copy operation of the open time values for last 10 bars has failed")
return;
}
size=ArraySize(ArrayNotSeries);
for(int i=size-1;i>=0;i--)
{
    Print("ArrayNotSeries["+i+"] =",ArrayNotSeries[i]);
}
```

结果我们可以如下输出：

```
TimeCurrent = 2009.06.11 14:16:23
ArraySize(Time) = 10
TimeAsSeries[0] = 2009.06.11 14:00:00
TimeAsSeries[1] = 2009.06.11 13:00:00
TimeAsSeries[2] = 2009.06.11 12:00:00
TimeAsSeries[3] = 2009.06.11 11:00:00
TimeAsSeries[4] = 2009.06.11 10:00:00
TimeAsSeries[5] = 2009.06.11 09:00:00
TimeAsSeries[6] = 2009.06.11 08:00:00
TimeAsSeries[7] = 2009.06.11 07:00:00
TimeAsSeries[8] = 2009.06.11 06:00:00
TimeAsSeries[9] = 2009.06.11 05:00:00

ArrayNotSeries[9] = 2009.06.11 14:00:00
ArrayNotSeries[8] = 2009.06.11 13:00:00
ArrayNotSeries[7] = 2009.06.11 12:00:00
ArrayNotSeries[6] = 2009.06.11 11:00:00
ArrayNotSeries[5] = 2009.06.11 10:00:00
ArrayNotSeries[4] = 2009.06.11 09:00:00
ArrayNotSeries[3] = 2009.06.11 08:00:00
ArrayNotSeries[2] = 2009.06.11 07:00:00
ArrayNotSeries[1] = 2009.06.11 06:00:00
ArrayNotSeries[0] = 2009.06.11 05:00:00
```

从结果中可以看到，TimeAsSeries数组增加的索引，索引减少的时间值，也就是说，从当前移到过去，对于普通数组ArrayNotSeries来说，结果是不同的——一旦索引增加，从过去移动到当前。

另见

[数组动态](#)，[ArrayGetAsSeries](#)，[ArraySetAsSeries](#)，[数据系列](#)

数组，缓冲器和时序列中索引方向

所有数组和指标缓冲区的默认索引是从左至右的，第一元素的索引等于0，因此，数组或指标缓冲区的第一元素默认从最左边的0开始，而最后一个元素在最右边。

指标缓冲区是双精度类型 [动态数组](#)，由客户端进行管理，因此相当于指标的字节数量。一个普通的双精度类型动态数组可以使用 [SetIndexBuffer\(\)](#) 函数分派到指标缓冲区。指标缓冲区不要求使用 [ArrayResize\(\)](#) 函数来建立大小-它会被终端的执行系统来执行。

[时序列](#)是颠倒搜索的数组，例如，时序列的第一个元素在最右边，最后一个元素在最左边。时序列是用来存储历史价格数据并包括时间信息，我们可以看见时序列最右边是最新数据，而最左边是最旧的数据。

因此时序列以0索引，包括最近引用的交易品种的信息。如果时序列包括每日时间列表的数据，当前尚未完成的日期会显示在0位置，而1索引位置会显示昨天的数据。

改变索引方向

函数 [ArraySetAsSeries\(\)](#) 允许改变方位动态数组元素的方法，数据的自动命令在电脑内存中并未改变。该函数简单改变了安置数组元素的方法，因此当使用[ArrayCopy\(\)](#)，函数复制一组数组到另一组中时，接收数组的命令会依据源数组的索引方向执行。

索引方向不能为分散的静态数组改变，即使数组以参量传递函数，想要改变内置的索引方向也是不可以的。

对于指标缓冲区来说，像普通数组一样，索引方向向后设置（像时序列），例如，关于指标缓冲区的零位置代表类似指标缓冲区的最后值，也相当于指标最新字节的值。因此，指标的实际字节位置并未改变。

指标接收价格数据

每个[自定义指标](#)都必要地建立 [OnCalculate\(\)](#) 函数，价格数据需要传递指标缓冲区的计算值，搜索方向可以在传递数组中使用 [ArrayGetAsSeries\(\)](#) 函数找到。

[传递到函数](#)的数组反应价格数据，例如，当检测数组时，这些数组都有时序列标志和函数[ArrayIsSeries\(\)](#) 返回真值，然后，在许多情况下，索引方向应该只能通过 [ArrayGetAsSeries\(\)](#) 函数检测。

为了不依赖默认值，[ArraySetAsSeries\(\)](#) 应该为数组无条件调用，并建立必要方向。

接收价格数据和指标值

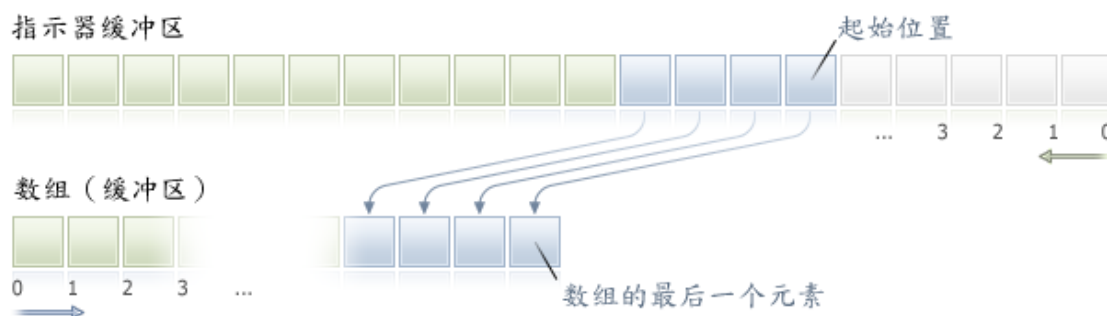
在EA交易，指标和脚本中的所有默认检索方向都是从左至右。如果必要的话，在MQL5程序中，可以要求时序列价值是任意交易品种和时序列，指标的计算值是任意交易品种和时序列。

使用Copy...() 函数完成这些操作：

- [复制缓冲器](#) - 复制指标缓冲区到双精度类型数组；
- [复制价格](#) - 复制价格记录到结构 [MqlRates](#) 类型数组；
- [复制时间](#) - 复制时间值到日期时间类型数组；
- [复制开盘价](#) - 复制开仓价格到双精度类型数组；
- [复制最高价](#) - 复制最高价值到双精度类型数组；
- [复制最低价](#) - 复制最低价值到双精度类型数组；
- [复制收盘价](#) - 复制收盘价格到双精度类型数组；

- [复制点成交量](#) – 复制点成交量到长类型数组；
- [复制真实成交量](#) – 复制空成交量到长类型数组；
- [复制点差](#) – 复制点差到整型数组；

所有这些函数都以相同方式工作，考虑到CopyBuffer() 在示例中包含的原理，暗示在需求数据的索引方向是时间序列，0索引位置存储当前未完成字节的数据。为了房屋这些数据，需要复制必要的数据成交量容器数组，例如，数组 buffer。



当复制时，我们需要在源数组中指定起始位置，从数据复制的容器数组开始。在成功的条件下，指定元素数量从源数组中复制容器数组（假设从指标缓冲区中）。不考虑设置在容器数组中的索引值，复制执行总是显示在上述计算中。

IsStopped() :

```
int copied=CopyBuffer(ma_handle, // хэндл индикатора
    0, // индекс индикаторного буфера
    0, // стартовая позиция для копирования
    number, // количество значений для копирования
    Buffer // массив-получатель значений
);

if(copied<0) return;
int k=0;
while(k<copied && !IsStopped())
{
    //--- получаем значение для индекса k
    double value=Buffer[k];
    // ...
    // работа со значением value
    k++;
}
```

示例：

```
input int per=10; // 指数周期
int ma_handle; // 指标处理
//+-----+
```



```
//| 专家初始化函数 |
//+-----+
int OnInit()
{
//---
    ma_handle=iMA(_Symbol,0,per,0,MODE_EMA,PRICE_CLOSE);
//---
    return(0);
}
//+-----+
//| 专家订单号函数 |
//+-----+
void OnTick()
{
//---
    double ema[10];
    int copied=CopyBuffer(ma_handle,// 指标处理
                        0,          // 指标缓冲区指数
                        0,          // 复制的初始位置
                        10,        // 用于复制的值数
                        ema         // 接收数组的值
                        );

    if(copied<0) return;
// .... 深层代码
}
```

另见

[组织数据存储](#)

组织数据存储

这一章节是讲述获得、存储和索取价格数据的（[时间序列](#)）。

从交易服务器中获得数据

在MT5客户端的价格数据执行之前，必须先收到然后经过处理，为了接收数据，要确定MT5交易服务器已经确定连接，数据根据终端要求以微小字节形式接收。

服务器的机械装置是参考数据的，并不依靠最初的要求-使用者通过操作图表或者MQL5语言程序方式。

储存媒介数据

从服务器重接收的数据自动打开或存储在HCC媒介信息中，每个象征数据都以分类文件夹 `terminal_directory\bases\server_name\history\symbol_name` 的形式记录。例如，关于EURUSD的数据从MetaQuotes-Demo服务器中接收，但会存储在 `terminal_directory\bases\MetaQuotes-Demo\history\EURUSD` 中。

数据以.hcc的扩展名方式书写到文件中，每个文件存储分钟字节型数据一年，例如，2009年的数据文件包含2009年所有的分钟字节型数据，这些文件为所有时间架构提供价格数据的，他们并不能直接接入。

在媒介数据外获得必要的时间架构

媒介HCC文件以源数据类型使用，它在HC板式中以建立价格数据的要求来使用的，HC板式的数据是事件序列为快速接入的最大化准备，他们根据图表或MQL5程序的要求来构建，数据的成交量不能超过图表字节的最大值，数据长期存储使用hc扩展文件。

为了节约资源，在时间架构里的数据如果需要的话以ram格式存储，如果不能长时间调用，会从RAM中释放并保存到文件夹中，对于每个时间序列来说，数据的存贮无论是否已经有为另一时间架构准备的备用数据，形式规则和接入数据对于所有时间架构来说都是一样的。那就意味着，尽管在HCC中存储了一分钟数据，HCC数据的可行性并不意味着在同成交量中M1时间架构的可行性。

从服务器中接收新数据会调用所有时间架构HC板式的价格数据自动更新，也能导致所有指标的重新计算，那样可以暗中输出数据来计算。

参数“图表中的最大字节”

“图表中的最大字节”常量限制可行性图表、指标和MQL5程序中HC板式的字节数，对于所有可行性时间序列和服务器来说是有效的，首先，为了节约电脑资源。

当建立有关该常量一个较大值时，需要记住的是，如果深度价格数据对于小的时间架构来说是可行的，用来存储时间序列和指标缓冲区的内存可以是上百个字节，并且达到RAM对客户端程序的限制（2Gb、MS视窗32字节）。

在客户端重启之后，“图表中最大字节”的改变开始生效，该参量的转变原因并不是递交给服务器额外的数据，也不是形成额外的时序列字节，额外价格数据从服务器上要求，时序列的更新于新的限制有关，以防无数据图表卷轴或者数据被MQL5程序调用。

服务器要求的数据成交量与时间架构的“图表中最大字节”的字节要求数量有关，该参数限制并不严格，多数情况下，可利用的字节数字比当前参量值还多。

数据有效性

如果这些数据在MQL5程序中以图表或者使用的形式出现，HCC版式中的数据或者为使用HC版式而准备的数据并

不能完全可用。

当MQL5程序访问价格数据或者指示值时，他们的可行性是一个确切的时间值或者有保留地从一个确定的时间点开始，事实上，为了节约资源，必要数据的复制并不全部存储在MT5中，只是给出了终端数据库。

所有时间架构的价格历史都从普通的HCC版本数据建立，然后从服务器更新系统更新，重新计算指标，由于这种数据接入能够关闭，甚至这些数据在之前都能使用。

终端数据和服务器数据的同步性

自从MQL5程序可以以符号和时间架构的形式调用数据信息，就存在在终端必要的时序列数据没有建立的可能性，或者必要的价格数据没有与交易服务器同步化，因此很难预测潜在时间。

使用潜在循环计算式并不是最好的解决方法，这一事件的唯一例外是脚本，因为有事件要处理，他们没有交替运算方法的选择。对于此种算法的常规指标，和其他潜在循环动作并没有被推荐使用，因为导致所有指标的计算终止，和其他价格数据操作失败。

对于EA交易和指标来说，使用 [相等模式](#) 来处理最好不过，在处理OnTick() 或 OnCalculate() 事件时，必要时序列的接受数据是失败的，此时需要离开事件处理器，在下一个处理器调用的时候适当接入。

脚本添加历史的范例

示例是，从交易服务器中接收脚本处理预选对象的请求，脚本已经为运行预选对象图表做了准备；时间架构不能操作，因为如上所述，价格数据是从交易服务器中以一分钟数据形式大包而来的，当时任何预选时序列都在构造中。

记录下所有数据脚本的操作来当成一个分离函数CheckLoadHistory(symbol, timeframe, start_date)：

```
int CheckLoadHistory(string symbol,ENUM_TIMEFRAMES period,datetime start_date)
{
}
```

CheckLoadHistory() 函数是一种能被任何操作调用的通用函数（EA交易、脚本或指标）；此前它需要三种输入参量：符号名称，周期和开始日期来表明你所需的价格历史开始运行。

在调用缺失的历史之前，在函数代码中插入格，首先，应该确保符号名称和周期值是正确的：

```
if(symbol==NULL || symbol=="") symbol=Symbol();
if(period==PERIOD_CURRENT) period=Period();
```

确定该符号在市场观测窗口的可见性，如下，当向交易服务器发出请求时，符号的历史就可用，如果在市场观测中没有这样的符号，使用 [SymbolSelect\(\)](#) 函数可以添加。

```
if(!SymbolInfoInteger(symbol,SYMBOL_SELECT))
{
    if(GetLastError()==ERR_MARKET_UNKNOWN_SYMBOL) return(-1);
    SymbolSelect(symbol,true);
}
```

现在我们能够收到可用历史交易的起始时间，也许，起始输入函数值，传递到CheckLoadHistory()，是可行性历史，而后交易服务器的请求就不需要了。为了获得第一时间的符号序列，可用使用 [SeriesInfoInteger\(\)](#) 函数的 [SERIES_FIRSTDATE](#) 修饰语。

```
SeriesInfoInteger(symbol,period,SERIES_FIRSTDATE,first_date);
if(first_date>0 && first_date<=start_date) return(1);
```

下一项重要的检测就是从函数调用检验程序类型，注释，与指标同时发送时序列的更新请求，该种调用更新是不符合要求的。指标中与符号序列相同的要求数据受历史数据约束，因此，固定发生的概率偏高，检验该项目可以使用 `MQL5InfoInteger()` 函数中的 `MQL5_PROGRAM_TYPE` 修饰语。

```
if(MQL5InfoInteger(MQL5_PROGRAM_TYPE)==PROGRAM_INDICATOR && Period()==period && Symbol()!="")
    return(-4);
```

如果所有的检测都成功通过，最后要涉及的就是交易服务器，首先找到起始日期，在HCC版式下的分钟日期就可。使用功能函数 `SeriesInfoInteger()` 中 `SERIES_TERMINAL_FIRSTDATE` 修饰语可以调用该值，也可与 `start_date` 常数值做比较。

```
if(SeriesInfoInteger(symbol,PERIOD_M1,SERIES_TERMINAL_FIRSTDATE,first_date))
{
    //--- 有建立时间序列的加载数据
    if(first_date>0)
    {
        //--- 强制创建时间序列
        CopyTime(symbol,period,first_date+PeriodSeconds(period),1,times);
        //--- 检测日期
        if(SeriesInfoInteger(symbol,period,SERIES_FIRSTDATE,first_date))
            if(first_date>0 && first_date<=start_date) return(2);
    }
}
```

如果所有的检测都执行通过，却仍然有 `CheckLoadHistory()` 函数，意味着有必要要求交易服务器调回丢失价格代码，首先，返回“图表中最大字节”值，使用 `TerminalInfoInteger()` 函数：

```
int maxBars=TerminalInfoInteger(TERMINAL_MAXBARS);
```

需要阻止额外调回数据，在服务器中找回字符历史的第一个日期（忽略周期），使用已知函数 `SeriesInfoInteger()` 的修饰语 `SERIES_SERVER_FIRSTDATE`。

```
datetime first_server_date=0;
while(!SeriesInfoInteger(symbol,PERIOD_M1,SERIES_SERVER_FIRSTDATE,first_server_date))
    Sleep(5);
```

该请求是异步操作，函数在回收时调用时间晚了5毫秒，直到 `first_server_date` 变量接收到一个值，或者该执行周期被用户终止（`IsStopped()` 在此情形下会返回真值），在起始日期指明正确值，就从向交易服务器要求价格数据时开始。

```
if(first_server_date>start_date) start_date=first_server_date;
if(first_date>0 && first_date<first_server_date)
    Print("Warning: first server date ",first_server_date," for ",
symbol," does not match to first series date ",first_date);
```

如果服务器的起始日期是 `first_server_date` 比HCC版式的符号 `first_date` 起始日期早，在分类账户中，类似进入将会被输出。

现在向交易服务器发送一个丢失价格数据的请求，该请求以循环的模式开始填满全部函数。

```

while(!IsStopped())
{
    //1. 等候如同HHC一样重新建立时间序列与中间历史记录的同步化
    //2. 接收这个时间序列的当前n个柱数
    //    如果柱大于图表中的最大柱，可以退出，工作结束
    //3. 获得重建时间序列开始日期的初始日期并且与开始日期相比较
    //    如果初始日期低于开始日期，则退出，工作结束
    //4. 从服务器请求历史记录的新部分-从最近有效标有'bars'的柱中启动的100柱
}

```

使用已知手段可以实行前三种方法。

```

while(!IsStopped())
{
    //--- 1. 等待直到时间序列重建程序结束
    while(!SeriesInfoInteger(symbol,period,SERIES_SYNCRONIZED) && !IsStopped())
        Sleep(5);
    //--- 2. 请求多少柱
    int bars=Bars(symbol,period);
    if(bars>0)
    {
        //--- 超过图表中所画的柱，退出
        if(bars>=max_bars) return(-2);
        //--- 3. 返回时间序列的当前开始日期
        if(SeriesInfoInteger(symbol,period,SERIES_FIRSTDATE,first_date))
            // 开始日期早于要求日期，任务完成
            if(first_date>0 && first_date<=start_date) return(0);
    }
    //4. 从服务器请求历史记录的新部分-从最近有效标有'bars'的柱中启动的100柱
}

```

剩下第四步-索取历史，不能直接查阅服务器，如果在HCC板式中的历史不足，任何[Copy-function](#)的启动要求都发往服务器，first_date 变量的第一个起始日期是简明的，也有对请求等级的衡量标准，最快的方法是使用[CopyTime\(\)](#) 函数。

当调用函数从时序复制数据时，应该标记起始参量（字节的数字，复制价格数据的起始），这些都应该在服务器历史中找到，如果只有100字节，从500个指标中复制300字节是无效的，该请求会被视为错误出现不能执行，例如，交易服务器中没有额外历史记录加载。

这就是我们从起始字节指标复制100字节的原因，它会从交易服务器中顺利下载丢失的历史数据，事实上，下载的往往比要求的100字节多一点，服务器也会发送大一些的历史数据。

```
int copied=CopyTime(symbol,period,bars,100,times);
```

复制操作之后，需要分析复制元素号码，如果尝试失败，copied值等于空值并且计数值fail_cnt增加到1，在100次尝试失败后，函数操作就会停止。

```
int fail_cnt=0;
```

```

...
int copied=CopyTime(symbol,period,bars,100,times);
if(copied>0)
{
    ///--- 检测数据
    if(times[0]<=start_date) return(0); // 复制值更小, 准备完毕
    if(bars+copied>=max_bars) return(-2); // 柱多于图表中所画的柱, 准备完毕
    fail_cnt=0;
}
else
{
    ///--- 成功的话失败尝试少于100
    fail_cnt++;
    if(fail_cnt>=100) return(-5);
    Sleep(10);
}

```

因此, 不仅当前情况的恰当处理在函数中可以实施, 而且还能返回结束代码, 在调用CheckLoadHistory() 函数后可以直接解决获得额外信息, 例如, 这种方法:

```

int res=CheckLoadHistory(InpLoadedSymbol, InpLoadedPeriod, InpStartDate);
switch(res)
{
    case -1 : Print("Unknown symbol ", InpLoadedSymbol); break;
    case -2 : Print("More requested bars than can be drawn in the chart"); break;
    case -3 : Print("Execution stopped by user"); break;
    case -4 : Print("Indicator mustn't load its own data"); break;
    case -5 : Print("Loading failed"); break;
    case 0 : Print("All data loaded"); break;
    case 1 : Print("Already available data in timeseries are enough"); break;
    case 2 : Print("Timeseries is built from available terminal data"); break;
    default : Print("Execution result undefined");
}

```

函数的全部代码能在脚本示例中找到, 显示出正确的接入操作与请求结果正在处理。

代码:

```

//+-----+
//|                                     TestLoadHistory.mq5 |
//|                                     Copyright 2009, MetaQuotes Software Corp. |
//|                                     http://www.mql5.com |
//+-----+
#property copyright "2009, MetaQuotes Software Corp."
#property link      "http://www.mql5.com"
#property version   "1.02"
#property script_show_inputs
///--- 输入参数
input string        InpLoadedSymbol="NZDUSD"; // 被加载的交易品种
input ENUM_TIMEFRAMES InpLoadedPeriod=PERIOD_H1; // 被加载的周期
input datetime      InpStartDate=D'2006.01.01'; // 开始日期
//+-----+
//| 脚本程序启动函数 |
//+-----+
void OnStart()

```

```

{
    Print("Start load", InpLoadedSymbol+", "+GetPeriodName(InpLoadedPeriod), "from", InpSt
//---
    int res=CheckLoadHistory(InpLoadedSymbol, InpLoadedPeriod, InpStartDate);
    switch(res)
    {
        case -1 : Print("Unknown symbol ", InpLoadedSymbol); break;
        case -2 : Print("Requested bars more than max bars in chart"); break;
        case -3 : Print("Program was stopped"); break;
        case -4 : Print("Indicator shouldn't load its own data"); break;
        case -5 : Print("Load failed"); break;
        case 0 : Print("Loaded OK"); break;
        case 1 : Print("Loaded previously"); break;
        case 2 : Print("Loaded previously and built"); break;
        default : Print("Unknown result");
    }
//---
    datetime first_date;
    SeriesInfoInteger(InpLoadedSymbol, InpLoadedPeriod, SERIES_FIRSTDATE, first_date);
    int bars=Bars(InpLoadedSymbol, InpLoadedPeriod);
    Print("First date ", first_date, " - ", bars, " bars");
//---
}
//+-----+
//|
//+-----+
int CheckLoadHistory(string symbol, ENUM_TIMEFRAMES period, datetime start_date)
{
    datetime first_date=0;
    datetime times[100];
//--- 检测交易品种 & 周期
    if(symbol==NULL || symbol=="") symbol=Symbol();
    if(period==PERIOD_CURRENT) period=Period();
//--- 检测市场报价是否选定了交易品种
    if(!SymbolInfoInteger(symbol, SYMBOL_SELECT))
    {
        if(GetLastError()==ERR_MARKET_UNKNOWN_SYMBOL) return(-1);
        SymbolSelect(symbol, true);
    }
//--- 检测数据是否存在
    SeriesInfoInteger(symbol, period, SERIES_FIRSTDATE, first_date);
    if(first_date>0 && first_date<=start_date) return(1);
//--- 如果是指标的话不需要加载自己的数据
    if(MQL5InfoInteger(MQL5_PROGRAM_TYPE)==PROGRAM_INDICATOR && Period()==period && Sy
        return(-4);
//--- 第二尝试
    if(SeriesInfoInteger(symbol, PERIOD_M1, SERIES_TERMINAL_FIRSTDATE, first_date))
    {

```

```

//--- 有建立时间序列加载的数据
if(first_date>0)
{
    //--- 强制创建时间序列
    CopyTime(symbol,period,first_date+PeriodSeconds(period),1,times);
    //--- 检测日期
    if(SeriesInfoInteger(symbol,period,SERIES_FIRSTDATE,first_date))
        if(first_date>0 && first_date<=start_date) return(2);
}
}

//--- 图表中来自程序端选项的最大柱
int max_bars=TerminalInfoInteger(TERMINAL_MAXBARS);
//--- 加载交易品种历史记录信息
datetime first_server_date=0;
while(!SeriesInfoInteger(symbol,PERIOD_M1,SERIES_SERVER_FIRSTDATE,first_server_date))
    Sleep(5);
//--- 为加载确定开始日期
if(first_server_date>start_date) start_date=first_server_date;
if(first_date>0 && first_date<first_server_date)
    Print("Warning: first server date ",first_server_date," for ",symbol,
        " does not match to first series date ",first_date);
//--- 逐步地加载数据
int fail_cnt=0;
while(!IsStopped())
{
    //--- 等待建立时间序列
    while(!SeriesInfoInteger(symbol,period,SERIES_SYNCRONIZED) && !IsStopped())
        Sleep(5);
    //--- 要求建成的柱
    int bars=Bars(symbol,period);
    if(bars>0)
    {
        if(bars>=max_bars) return(-2);
        //--- 请求初始日期
        if(SeriesInfoInteger(symbol,period,SERIES_FIRSTDATE,first_date))
            if(first_date>0 && first_date<=start_date) return(0);
    }
    //--- 复制部分强制数据加载
    int copied=CopyTime(symbol,period,bars,100,times);
    if(copied>0)
    {
        //--- 数据检测
        if(times[0]<=start_date) return(0);
        if(bars+copied>=max_bars) return(-2);
        fail_cnt=0;
    }
    else
    {

```



```

        //--- 失败尝试少于100
        fail_cnt++;
        if(fail_cnt>=100) return(-5);
        Sleep(10);
    }
}

//--- 停止
return(-3);
}

//+-----+
//| 返回该周期的字符串值 |
//+-----+

string GetPeriodName(ENUM_TIMEFRAMES period)
{
    if(period==PERIOD_CURRENT) period=Period();
//---
    switch(period)
    {
        case PERIOD_M1: return("M1");
        case PERIOD_M2: return("M2");
        case PERIOD_M3: return("M3");
        case PERIOD_M4: return("M4");
        case PERIOD_M5: return("M5");
        case PERIOD_M6: return("M6");
        case PERIOD_M10: return("M10");
        case PERIOD_M12: return("M12");
        case PERIOD_M15: return("M15");
        case PERIOD_M20: return("M20");
        case PERIOD_M30: return("M30");
        case PERIOD_H1: return("H1");
        case PERIOD_H2: return("H2");
        case PERIOD_H3: return("H3");
        case PERIOD_H4: return("H4");
        case PERIOD_H6: return("H6");
        case PERIOD_H8: return("H8");
        case PERIOD_H12: return("H12");
        case PERIOD_D1: return("Daily");
        case PERIOD_W1: return("Weekly");
        case PERIOD_MN1: return("Monthly");
    }
//---
    return("unknown period");
}

```

SeriesInfoInteger

关于历史数据的返回信息，有2个变量可供调用。

直接返回属性值。

```
long SeriesInfoInteger(
    string          symbol_name,    // 交易品种名称
    ENUM_TIMEFRAMES timeframe,     // 周期
    ENUM_SERIES_INFO_INTEGER prop_id, // 属性标识符
);
```

依靠函数的成果运行返回真值或错误值。

```
bool SeriesInfoInteger(
    string          symbol_name,    // 交易品种名称
    ENUM_TIMEFRAMES timeframe,     // 周期
    ENUM_SERIES_INFO_INTEGER prop_id, // 属性ID
    long&          long_var       // 用于获得信息的变量
);
```

参量

symbol_name

[in] 交易品种名称。

timeframe

[in] 周期。

prop_id

[in] 要求性质标识符，计算 [ENUM_SERIES_INFO_INTEGER](#) 值。

long_var

[out] 计算安置后的要求性质变量值。

返回值

第一，返回长型值。

第二，如果指定性质可用并且该值添加到 *long_var* 常量中，返回true,否则返回false.更多细节参照 [error](#)，调用 [GetLastError\(\)](#)。

示例：

```
void OnStart()
{
    //---
    Print("Total number of bars for the symbol-period at this moment = ",
        SeriesInfoInteger(Symbol(), 0, SERIES_BARS_COUNT));

    Print("The first date for the symbol-period at this moment = ",
        (datetime)SeriesInfoInteger(Symbol(), 0, SERIES_FIRSTDATE));
```

```
Print("The first date in the history for the symbol-period on the server = ",  
      (datetime)SeriesInfoInteger(Symbol(), 0, SERIES_SERVER_FIRSTDATE));  
  
Print("Symbol data are synchronized = ",  
      (bool)SeriesInfoInteger(Symbol(), 0, SERIES_SYNCRONIZED));  
}
```

Bars

为指定交易品种和周期返回历史计算中的字节数量。有两种变量函数可以调用。

要求所有历史字节

```
int Bars(  
    string          symbol_name,      // 交易品种名称  
    ENUM_TIMEFRAMES timeframe        // 周期  
);
```

为所选时间间隔要求历史字节

```
int Bars(  
    string          symbol_name,      // 交易品种名称  
    ENUM_TIMEFRAMES timeframe,        // 周期  
    datetime        start_time,       // 启动日期和时间  
    datetime        stop_time        // 结束日期和时间  
);
```

参量

symbol_name
[in] 交易品种名称。

timeframe
[in] 周期。

start_time
[in] 与第一元素相一致的字节时间。

stop_time
[in] 与最后一个元素相一致的字节时间。

返回值

如果定义start_time 和 stop_time参量，函数返回指定时间间隔的自己数量，否则返回全部字节数量。

注释

如果指定参量的时序列数据通过Bars() 函数调用在终端形成，或者时序列数据在函数调用时不在[synchronized](#)交易服务器中，函数返回0值。

示例：

```

int bars=Bars(_Symbol,_Period);
if(bars>0)
{
    Print("Number of bars in the terminal history for the symbol-period at the moment");
}
else //无有效柱
{
    //--- 交易品种数据不能和服务端数据同步
    bool synchronized=false;
    //--- 循环计数器
    int attempts=0;
    // 进行5次尝试等候同步进行
    while(attempts<5)
    {
        if(SeriesInfoInteger(Symbol(),0,SERIES_SYNCRONIZED))
        {
            //--- 同步化完成,退出
            synchronized=true;
            break;
        }
        //--- 增加计数器
        attempts++;
        //--- 等候10毫秒直至嵌套反复
        Sleep(10);
    }
    //--- 同步化后退出循环
    if(synchronized)
    {
        Print("Number of bars in the terminal history for the symbol-period at the moment");
        Print("The first date in the terminal history for the symbol-period at the moment = ",
            (datetime)SeriesInfoInteger(Symbol(),0,SERIES_FIRSTDATE));
        Print("The first date in the history for the symbol on the server = ",
            (datetime)SeriesInfoInteger(Symbol(),0,SERIES_SERVER_FIRSTDATE));
    }
    //--- 不发生数据同步
    else
    {
        Print("Failed to get number of bars for ",_Symbol);
    }
}

```

另见

[事件处理函数](#)

BarsCalculated

返回指定指标计算数组的数量。

```
int BarsCalculated(  
    int indicator_handle, // 指标处理  
);
```

参量

indicator_handle

[in] 指标处理，通过类似指标函数返回。

返回值

在指标缓冲区返回计算数据的数量或者错误中的-1（数据不能进行计算）。

注释

当需要在创建后立即获得指标数据时，该函数是很有用的（指标处理可行）。

示例：

```
void OnStart()  
{  
    double Ups[];  
    //--- 设置整理数组的时间序列  
    ArraySetAsSeries(Ups,true);  
    //--- 为分形指标创建处理程序  
    int FractalsHandle=iFractals(NULL,0);  
    //--- 重置错误代码  
    ResetLastError();  
    //--- 复制指标值  
    int i,copied=CopyBuffer(FractalsHandle,0,0,1000,Ups);  
    if(copied<=0)  
    {  
        Sleep(50);  
        for(i=0;i<100;i++)  
        {  
            if(BarsCalculated(FractalsHandle)>0)  
                break;  
            Sleep(50);  
        }  
        copied=CopyBuffer(FractalsHandle,0,0,1000,Ups);  
        if(copied<=0)  
        {  
            Print("Failed to copy upper fractals. Error = ",GetLastError(),  
                "i = ",i," copied = ",copied);  
            return;  
        }  
    }  
    else
```

```
Print("Upper fractals copied",  
      "i = ",i,"      copied = ",copied);  
}  
else Print("Upper fractals copied. ArraySize = ",ArraySize(Ups));  
}
```

IndicatorCreate

函数返回以 [MqlParam](#) 类型参量数组为基础建立的特定技术计时器处理。

```
int IndicatorCreate(
    string          symbol,           // 交易品种名称
    ENUM_TIMEFRAMES period,          // 时间表
    ENUM_INDICATOR  indicator_id,    // ENUM_INDICATOR枚举中的指标类
    int             parameters_cnt=0, // 参量号
    const MqlParam& parameters_array[]=NULL, // 参量数组
);
```

参量

symbol

[in] 交易品种名称，指标计算数据，[NULL](#) 表示当前交易品种。

period

[in] 时间表的值可以是 [ENUM_TIMEFRAMES](#) 值中的一个，0代表当前时间表。

indicator_id

[in] 指标类型，可以是 [ENUM_INDICATOR](#) 值中的一个。

parameters_cnt

[in] 参量的数量在 `_array[]` 参量数组中传递，数组元素类型有特殊的结构类型 [MqlParam](#)。默认，0-参量不传递。如果指定参量的非零数量，参量 `parameters__array` 是强制的，可以传递超过256个参量。

parameters_array[]=NULL

[in] [MqlParam](#) 类型数组，元素包括类型和每个[技术指标](#)输入参数的类型值。

返回值

返回特定技术指标的处理，错误返回 [INVALID_HANDLE](#)。

注释

如果IND_CUSTOM类型指标处理建立，数组中第一元素的 type域，输入参量 `parameters__array` 一定是 [ENUM_DATATYPE](#) 中的TYPE_STRING值，第一元素类型 `string_value` 域一定包括自定义指标名称。自定义指标必须在MQL5/Indicators客户端或者子目录中编辑。

在[第一调用形式](#)自定义指标中，当传递输入参量时可以加上数据参量。如果"Apply to"参量没有明确规定，默认计算是以 [PRICE_CLOSE](#) 值为基础的。

示例：

```
void OnStart()
{
    MqlParam params[];
    int      h_MA, h_MACD;
    // --- 创建 iMA("EURUSD", PERIOD_M15, 8, 0, MODE_EMA, PRICE_CLOSE);
    ArrayResize(params, 4);
    // --- 设置 ma_period
```



```
params[0].type          =TYPE_INT;
params[0].integer_value=8;
//--- 设置 ma_shift
params[1].type          =TYPE_INT;
params[1].integer_value=0;
//--- 设置 ma_method
params[2].type          =TYPE_INT;
params[2].integer_value=MODE_EMA;
//--- 数组 applied_price
params[3].type          =TYPE_INT;
params[3].integer_value=PRICE_CLOSE;
//--- 创建 MA
h_MA=IndicatorCreate("EURUSD",PERIOD_M15,IND_MA,4,params);
//--- 创建 iMACD("EURUSD",PERIOD_M15,12,26,9,h_MA);
ArrayResize(params,4);
//--- 设置快速 ma_period
params[0].type          =TYPE_INT;
params[0].integer_value=12;
//--- 设置慢速 ma_period
params[1].type          =TYPE_INT;
params[1].integer_value=26;
//--- 为不同性设置平滑周期
params[2].type          =TYPE_INT;
params[2].integer_value=9;
//--- 设置指标处理程序为 applied_price
params[3].type          =TYPE_INT;
params[3].integer_value=h_MA;
//--- 创建基于移动平均数的MACD
h_MACD=IndicatorCreate("EURUSD",PERIOD_M15,IND_MACD,4,params);
//--- 使用指标
//--- . . .
//--- 发布指标 (首先 h_MACD)
IndicatorRelease(h_MACD);
IndicatorRelease(h_MA);
}
```

IndicatorRelease

如果不使用，函数删除指标缓冲区并释放指标计算空间。

```
bool IndicatorRelease(
    int      indicator_handle,    // 指标处理
);
```

返回值

如果成功返回真值，否则返回错误值。

注释

如果不再需要，函数允许删除指标缓冲区来节省内存。处理立即删除，计算块会在一定时间内删除（如果不再调用）。

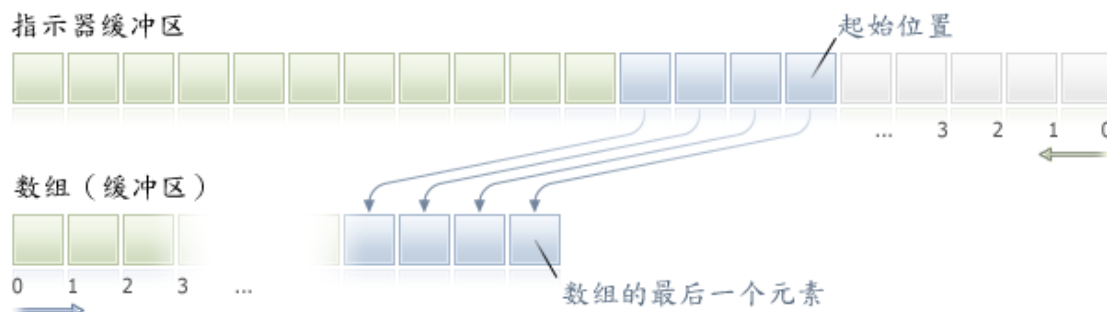
示例：

```
//+-----+
//|                                     Test_IndicatorRelease.mq5 |
//|                                     Copyright 2010, MetaQuotes Software Corp. |
//|                                     http://www.mql5.com |
//+-----+
#property copyright "2010, MetaQuotes Software Corp."
#property link      "http://www.mql5.com"
#property version   "1.00"
//--- 输入参数
input int           MA_Period=15;
input int           MA_shift=0;
input ENUM_MA_METHOD MA_smooth=MODE_SMA;
input ENUM_APPLIED_PRICE price=PRICE_CLOSE;
//--- 存储指标处理程序
int MA_handle=INVALID_HANDLE;
//+-----+
//| 专家初始化函数                                     |
//+-----+
int OnInit()
{
//--- 创建指标处理程序
    MA_handle=iMA(Symbol(),0,MA_Period,MA_shift,MA_smooth,PRICE_CLOSE);
//--- 删除全局变量
    if(GlobalVariableCheck("MA_value"))
        GlobalVariableDel("MA_value");
//---
    return(0);
}
//+-----+
//| 专家订单号函数                                     |
//+-----+
void OnTick()
```

```
{
//--- 如果全局变量值不存在
if(!GlobalVariableCheck("MA_value"))
{
//--- 在最近2柱获得指标值
if(MA_handle!=INVALID_HANDLE)
{
//--- 指标值的动态数组
double values[];
if(CopyBuffer(MA_handle,0,0,2,values)==2 && values[0]!=EMPTY_VALUE)
{
//--- 记住倒数第二柱的全局变量值
if(GlobalVariableSet("MA_value",values[0]))
{
//--- 释放指标处理程序
if(!IndicatorRelease(MA_handle))
Print("IndicatorRelease() failed. Error ",GetLastError());
else MA_handle=INVALID_HANDLE;
}
else
Print("GlobalVariableSet failed. Error ",GetLastError());
}
}
}
//---
}
```

CopyBuffer

在需要数量中，获得确定指标指定缓冲区的数据。



从一开始执行位置计算复制数据元素（指标缓冲区索引buffer_num）到当前传递位置。例如，起始位置是0代表当前字节（当前字节指标值）。

当复制位置数量的数据时，推荐使用[动态数组](#)和buffer[]容器缓冲区，因为函数试图分配复制数据的接收数组的大小。如果指标缓冲区（通过使用[SetIndexBuffer\(\)](#)函数将数组再分配给存储指标值）以buffer[]容量数组来使用，就允许局部复制。示例可以在标准客户端包自定义指标Awesome_Oscillator.mql5中找到。

如果需要局部复指标值到另一数组（没有指标缓冲区），应该使用媒介数组，这样可以复制预期数量。如果从中间接收数组，可以管理复制所需元素值的管理之后加到要求的位置。

如果知道所需复制数据的数量，为了阻止过多分配内存，最好操作[静态分配缓冲区](#)。

无论目标数组- as_series=true or as_series=false的属性是什么。数据都需要复制，因为最旧的元素放置在数组的内存存储位置，有3种变体可以实现函数调用。

第一位置调用和所需元素数量

```
int CopyBuffer(
    int      indicator_handle,    // 指标处理
    int      buffer_num,         // 指标缓冲区数
    int      start_pos,          // 启动位置
    int      count,              // 复制总额
    double   buffer[])           // 复制的目标数组
);
```

开始日期调用和所需元素的数量

```
int CopyBuffer(
    int      indicator_handle,    // 指标处理
    int      buffer_num,         // 指标缓冲区数
    datetime start_time,         // 启动日期和时间
    int      count,              // 复制总额
    double   buffer[])           // 复制的目标数组
);
```

开始调用和要求时间间隔的最后日期

```

int CopyBuffer(
    int      indicator_handle,    // 指标处理
    int      buffer_num,         // 指标缓冲区数
    datetime start_time,         // 启动日期和时间
    datetime stop_time,          // 结束日期和时间
    double   buffer[]            // 复制的目标数组
);

```

参量

indicator_handle

[in] 指标处理，通过类似指标函数返回。

buffer_num

[in] 指标缓冲区数量。

start_pos

[in] 第一复制元素位置。

count

[in] 复制结算数据。

start_time

[in] 字节时间，与第一元素相一致。

stop_time

[in] 字节时间，与最后元素相一致。

buffer[]

[out] [双精度](#) 类型数组。

返回值

返回计算的复制数据或者[错误](#)时是-1。

注释

当向指标要求数据，如果要求的时序列没有建立或者需要从服务器上下载，函数很快会返回-1，但是下载/建立的过程将会开始。

当EA交易或脚本要求数据时，[来自服务器的下载](#) 会开始，如果终端本地没有这些数据，要求建立的时序列就会开始，如果数据可以从本地历史建立但尚未准备好。函数在超时期满时，会返回一定数量的准备好的数据，但是历史下载会继续，在下一个类似要求函数中会返回更多数据。

示例：

```

//+-----+
//|                                     TestCopyBuffer3.mq5 |
//|                                     Copyright 2009, MetaQuotes Software Corp. |
//|                                     http://www.mql5.com |
//+-----+
#property copyright "2009, MetaQuotes Software Corp."
#property link      "http://www.mql5.com"

```

```

#property version      "1.00"

#property indicator_separate_window
#property indicator_buffers 1
#property indicator_plots  1
//---- 图 MA
#property indicator_label1  "MA"
#property indicator_type1   DRAW_LINE
#property indicator_color1  clrRed
#property indicator_style1  STYLE_SOLID
#property indicator_width1  1
//--- 输入参数
input bool                AsSeries=true;
input int                 period=15;
input ENUM_MA_METHOD      smootMode=MODE_EMA;
input ENUM_APPLIED_PRICE  price=PRICE_CLOSE;
input int                 shift=0;
//--- 指标缓冲区
double                    MABuffer[];
int                        ma_handle;
//+-----+
//| 自定义指标初始化函数 |
//+-----+
int OnInit()
{
//--- 指标缓冲区绘图
    SetIndexBuffer(0,MABuffer,INDICATOR_DATA);
    Print("Parameter AsSeries = ",AsSeries);
    Print("Indicator buffer after SetIndexBuffer() is a timeseries = ",
        ArrayGetAsSeries(MABuffer));
//--- 设置短小的指标名称
    IndicatorSetString(INDICATOR_SHORTNAME,"MA("+period+")"+AsSeries);
//--- 设置 AsSeries(依据输入参数)
    ArraySetAsSeries(MABuffer,AsSeries);
    Print("Indicator buffer after ArraySetAsSeries(MABuffer,true); is a timeseries = "
        ArrayGetAsSeries(MABuffer));
//---
    ma_handle=iMA(Symbol(),0,period,shift,smootMode,price);
    return(0);
}
//+-----+
//| 自定义指标重复函数 |
//+-----+
int OnCalculate(const int rates_total,
                const int prev_calculated,
                const datetime &time[],
                const double &open[],
                const double &high[],

```

```

        const double &low[],
        const double &close[],
        const long &tick_volume[],
        const long &volume[],
        const int &spread[])
    {
//--- 检测是否计算了所有数据
        if(BarsCalculated(ma_handle)<rates_total) return(0);
//--- 不复制所有数据
        int to_copy;
        if(prev_calculated>rates_total || prev_calculated<=0) to_copy=rates_total;
        else
        {
            to_copy=rates_total-prev_calculated;
            //--- 常常复制最后的值
            to_copy++;
        }
//--- 复制
        if(CopyBuffer(ma_handle,0,0,to_copy,MABuffer)<=0) return(0);
//--- 为下次调用返回prev_calculated值
        return(rates_total);
    }
//+-----+

```

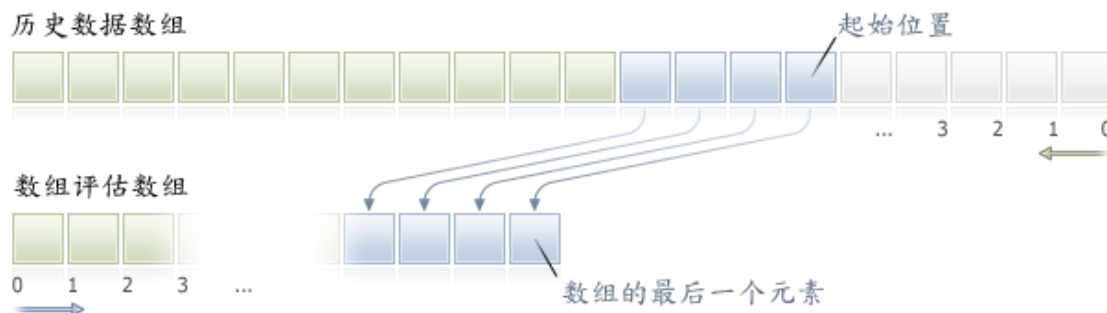
以上示例表明在相同交易品种周期中，指标缓冲区填满另一指标缓冲区的值。

另见

[自定义指标属性](#)， [SetIndexBuffer](#)

CopyRates

获得指定交易品种周期的特定MqlRates结构到rates_array数组中。元素复制数据的命令是从现在到过去，例如，从当前位置0开始表示当前字节。



当复制未知数量数据时，推荐使用[动态数组](#)为目标数组，因为如果要求数据账户少于（或多于）目标数值的长度，函数试图重新分配内存以完全适应要求数据。

如果知道所需复制的数据数量，为了阻止内存过多分配，最好使用 [静态分配缓冲区](#)。

无论目标数组- as_series=true or as_series=false的属性是什么，数据就会复制，以至于最旧的数组元素会在记忆内存开始中放置。有3个变量函数可供调用。

调用第一位置和要求元素的数量

```
int CopyRates(
    string          symbol_name,      // 交易品种名称
    ENUM_TIMEFRAMES timeframe,       // 周期
    int             start_pos,        // 启动位置
    int             count,            // 复制数据计算
    MqlRates        rates_array[]    // 复制目标数组
);
```

调用起始位置日期和要求元素数量

```
int CopyRates(
    string          symbol_name,      // 交易品种名称
    ENUM_TIMEFRAMES timeframe,       // 周期
    datetime        start_time,       // 开始日期和时间
    int             count,            // 复制数据计算
    MqlRates        rates_array[]    // 复制目标数组
);
```

调用起始位置和要求时间间隔的日期末尾

```
int CopyRates(
    string          symbol_name,      // 交易品种名称
    ENUM_TIMEFRAMES timeframe,       // 周期
```



```

datetime      start_time,      // 开始日期和时间
datetime      stop_time,       // 结束日期和时间
MqlRates      rates_array[]    // 复制目标数组
);

```

参量

symbol_name

[in] 交易品种名称。

timeframe

[in] 周期。

start_time

[in] 复制的第一元素字节时间。

start_pos

[in] 复制的第一元素起始位置。

count

[in] 复制的数据点。

stop_time

[in] 字节时间，与最后一个复制元素相一致。

rates_array[]

[out] [MqlRates](#) 类型数组

返回值

返回复制数据数量或者**错误**时是-1。

注释

如果要求数据的全部间隔不在服务器的可用数据内，函数返回-1。如果外部数据[TERMINAL_MAXBARS](#)（图表中的最大字节量）是要求的，函数也会返回-1。

当向指标要求数据，如果要求的时序列没有建立或者需要从服务器上下载，函数很快会返回-1，但是下载/建立的过程将会开始。

当EA交易或脚本要求数据时，[来自服务器的下载](#) 会开始，如果终端本地没有这些数据，要求建立的时序列就会开始，如果数据可以从本地历史建立但尚未准备好。函数在超时期满时，会返回一定数量的准备好的数据，但是历史下载会继续，在下一个类似要求函数中会返回更多数据。

当在日期指定范围内要求数据，只有间隔中的数据才能返回，间隔建立并指到秒。这表示，任意字节的开始时间，返回的值（成交量，传播，指标缓冲区的值，开仓价，最高价，最低价，收盘价或者开仓时间）是间隔要求范围内的。

因此，如果当前日期是星期六，想要复制指定的一周的时间表start_time=Last_Tuesday 和 stop_time=Last_Friday，函数的返回值是0，因为每周开盘时间是星期日，但是一周字节不能分成特殊的间隔。

如果需要返回与当前未完成字节相类似的值，可以调用指定的 start_pos=0 和 count=1 第一种形式。

示例：

```
void OnStart()
{
    //---
    MqlRates rates[];
    ArraySetAsSeries(rates,true);
    int copied=CopyRates(Symbol(),0,0,100,rates);
    if(copied>0)
    {
        Print("Bars copied: "+copied);
        string format="open = %G, high = %G, low = %G, close = %G, volume = %d";
        string out;
        int size=fmin(copied,10);
        for(int i=0;i<size;i++)
        {
            out=i+": "+TimeToString(rates[i].time);
            out=out+" "+StringFormat(format,
                                     rates[i].open,
                                     rates[i].high,
                                     rates[i].low,
                                     rates[i].close,
                                     rates[i].tick_volume);

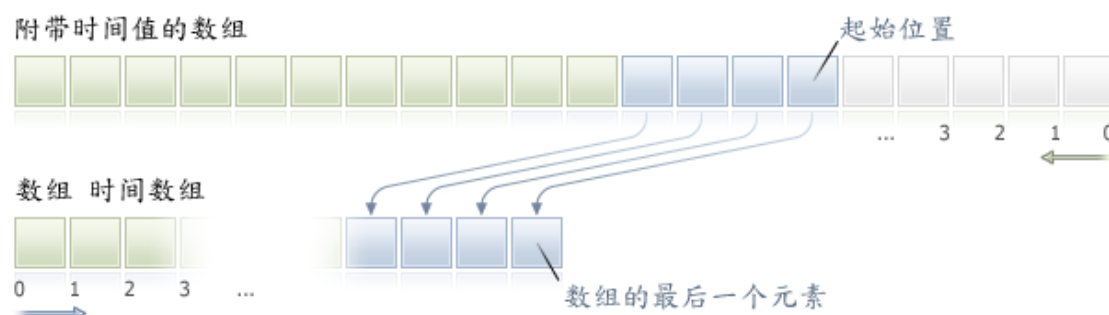
            Print(out);
        }
    }
    else Print("Failed to get history data for the symbol ",Symbol());
}
```

另见

[结构和类](#)， [时间到字符串](#)， [字符串格式](#)

CopyTime

函数在特定量中获得指定交易品种周期的特定开盘时间数据到time_array数据中。元素注释的命令是从现在到过去，例如，从当前位置0开始表示当前字节。



当复制未知数量数据时，推荐使用[动态数组](#)为目标数组，因为如果要求数据账户少于（或多于）目标数值的长度，函数试图重新分配内存以完全适应要求数据。

如果知道所需复制的数据数量，为了阻止内存过多分配，最好使用[静态分配缓冲区](#)。

无论目标数组- as_series=true 或者 as_series=false的属性是什么，数据就会复制，以至于最旧的数组元素会在记忆内存开始中放置。有3个变量函数可供调用。

调用第一位置和要求元素的数量

```
int CopyTime(
    string          symbol_name,      // 交易品种名称
    ENUM_TIMEFRAMES timeframe,       // 周期
    int             start_pos,        // 启动位置
    int             count,            // 复制的数据计算
    datetime        time_array[]      // 复制开仓时间的目标数组
);
```

调用起始位置日期和要求元素数量

```
int CopyTime(
    string          symbol_name,      // 交易品种名称
    ENUM_TIMEFRAMES timeframe,       // 周期
    datetime        start_time,       // 启动日期和时间
    int             count,            // 复制的数据计算
    datetime        time_array[]      // 复制开仓时间的目标数组
);
```

调用起始位置和要求时间间隔的日期末尾

```
int CopyTime(
    string          symbol_name,      // 交易品种名称
    ENUM_TIMEFRAMES timeframe,       // 周期
    datetime        start_time,       // 启动日期和时间
    datetime        stop_time,        // 结束日期和时间
```

```
datetime      time_array[]    // 复制开仓时间的目标数组
);
```

参量

symbol_name

[in] 交易品种名称。

timeframe

[in] 周期。

start_pos

[in] 复制第一元素的起始位置。

count

[in] 复制的数据点。

start_time

[in] 复制的第一元素起始时间。

stop_time

[in] 复制最后元素相一致的字节时间。

time_array[]

[out] [日期时间](#) 类型数组

返回值

返回复制数据点或者[错误](#)情况下-1。

注释

如果要求的数据的全部间隔没有在可行数据的服务器中，函数返回-1，如果外部数据[TERMINAL_MAXBARS](#)（图表中的最大字节量）是要求的，函数也会返回-1。

当向指标要求数据，如果要求的时序列没有建立或者需要从服务器上下载，函数很快会返回-1，但是下载/建立的过程将会开始。

当EA交易或脚本要求数据时，[来自服务器的下载](#)会开始，如果终端本地没有这些数据，要求建立的时序列就会开始，如果数据可以从本地历史建立但尚未准备好。函数在超时期满时，会返回一定数量的准备好的数据，但是历史下载会继续，在下一个类似要求函数中会返回更多数据。

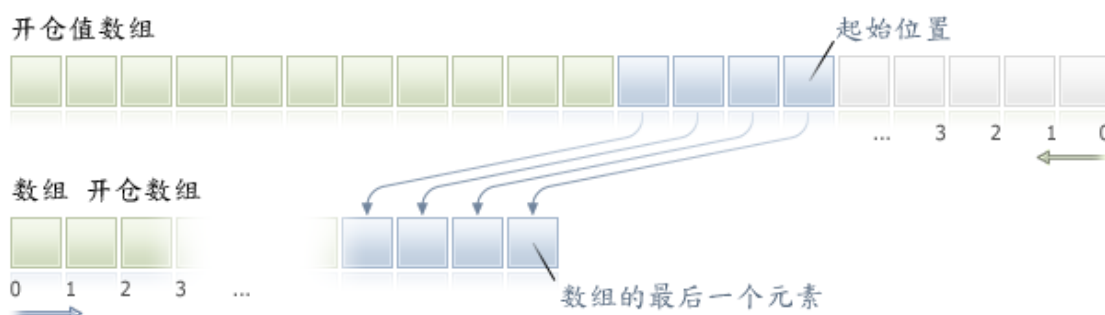
当在日期指定范围内要求数据，只有间隔中的数据才能返回，间隔建立并指到秒。这表示，任意字节的开始时间，返回的值（成交量，传播，指标缓冲区的值，开仓价，最高价，最低价，收盘价或者开仓时间）是间隔要求范围内的。

因此，如果当前日期是星期六，想要复制指定的一周的时间表 `start_time=Last_Tuesday` 和 `stop_time=Last_Friday`，函数的返回值是0，因为每周开盘时间是星期日，但是一周字节不能分成特殊的间隔。

如果需要返回与当前未完成字节相类似的值，可以调用指定的 `start_pos=0` 和 `count=1` 第一种形式。

CopyOpen

函数在特定量中获得指定交易品种周期的特定开盘价格数据到open_array数据中。元素注释的命令是从现在到过去，例如，从当前位置0开始表示当前字节。



当复制未知数量数据时，推荐使用 [动态数组](#) 为目标数组，因为如果要求数据账户少于（或多于）目标数值的长度，函数试图重新分配内存以完全适应要求数据。

如果知道所需复制的数据数量，为了阻止内存过多分配，最好使用 [静态分配缓冲区](#)。

无论目标数组- as_series=true 或者 as_series=false的属性是什么，数据就会复制，以至于最旧的数组元素会在记忆内存开始中放置。有3个变量函数可供调用。

调用第一位置和要求元素的数量

```
int CopyOpen(
    string          symbol_name,      // 交易品种名称
    ENUM_TIMEFRAMES timeframe,       // 周期
    int             start_pos,        // 启动位置
    int             count,            // 复制的数据计算
    double          open_array[]      // 复制开仓时间的目标数组
);
```

调用起始位置日期和要求元素数量

```
int CopyOpen(
    string          symbol_name,      // 交易品种名称
    ENUM_TIMEFRAMES timeframe,       // 周期
    datetime        start_time,       // 启动日期和时间
    int             count,            // 复制的数据计算
    double          open_array[]      // 用于开盘柱的目标数组
);
```

调用起始位置和要求时间间隔的日期末尾

```
int CopyOpen(
    string          symbol_name,      // 交易品种名称
    ENUM_TIMEFRAMES timeframe,       // 周期
    datetime        start_time,       // 启动日期和时间
    datetime        stop_time,        // 结束日期和时间
```

```
double      open_array[]    // 用于开盘柱的目标数组
);
```

参量

symbol_name

[in] 交易品种名称。

timeframe

[in] 周期。

start_pos

[in] 复制的第一元素起始位置。

count

[in] 复制的数据点。

start_time

[in] 复制的第一元素起始时间。

stop_time

[in] 复制的最后元素的起始时间。

open_array[]

[out] [双精度](#) 类型数组

返回值

返回数组中元素数量或者[错误](#)情况下-1。

注释

如果要求的数据的全部间隔没有在可行数据的服务器中，函数返回-1，如果外部数据[TERMINAL_MAXBARS](#)（图表中的最大字节量）是要求的，函数也会返回-1。

当向指标要求数据，如果要求的时序列没有建立或者需要从服务器上下载，函数很快会返回-1，但是下载/建立的过程将会开始。

当EA交易或脚本要求数据时，[来自服务器的下载](#)会开始，如果终端本地没有这些数据，要求建立的时序列就会开始，如果数据可以从本地历史建立但尚未准备好。函数在超时期满时，会返回一定数量的准备好的数据，但是历史下载会继续，在下一个类似要求函数中会返回更多数据。

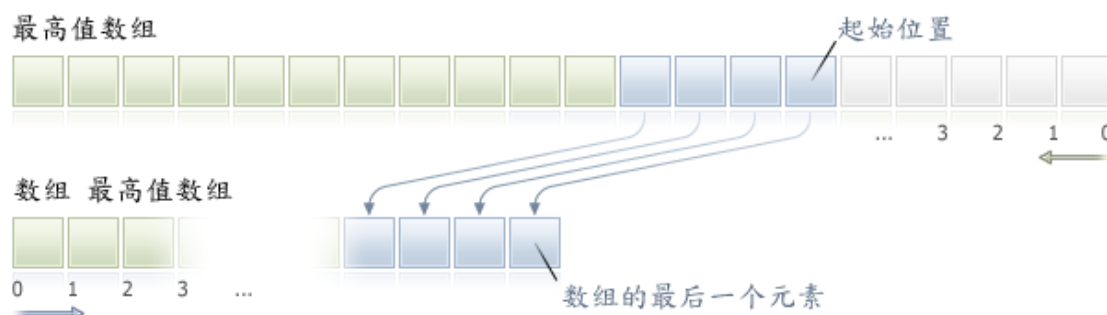
当在日期指定范围内要求数据，只有间隔中的数据才能返回，间隔建立并指到秒。这表示，任意字节的开始时间，返回的值（成交量，传播，指标缓冲区的值，开仓价，最高价，最低价，收盘价或者开仓时间）是间隔要求范围内的。

因此，如果当前日期是星期六，想要复制指定的一周的时间表 `start_time=Last_Tuesday` 和 `stop_time=Last_Friday`，函数的返回值是0，因为每周开盘时间是星期日，但是一周字节不能分成特殊的间隔。

如果需要返回与当前未完成字节相类似的值，可以调用指定的`start_pos=0`和`count=1`第一种形式。

CopyHigh

函数在特定量中获得指定交易品种周期的特定最高字节价格数据到high_array数据中。元素注释的命令是从现在到过去，例如，从当前位置0开始表示当前字节。



当复制未知数量数据时，推荐使用[动态数组](#)为目标数组，因为如果要求数据账户少于（或多于）目标数值的长度，函数试图重新分配内存以完全适应要求数据。

如果知道所需复制的数据数量，为了阻止内存过多分配，最好使用 [静态分配缓冲区](#)。

无论目标数组- as_series=true 或者 as_series=false的属性是什么，数据就会复制，以至于最旧的数组元素会在记忆内存开始中放置。有3个变量函数可供调用。

调用第一位置和要求元素的数量

```
int CopyHigh(
    string          symbol_name,      // 交易品种名称
    ENUM_TIMEFRAMES timeframe,        // 周期
    int             start_pos,        // 启动位置
    int             count,            // 复制的数据计算
    double          high_array[]      // 复制的目标数组
);
```

调用起始位置日期和要求元素数量

```
int CopyHigh(
    string          symbol_name,      // 交易品种名称
    ENUM_TIMEFRAMES timeframe,        // 周期
    datetime        start_time,       // 启动日期和时间
    int             count,            // 复制的数据计算
    double          high_array[]      // 复制的目标数组
);
```

调用起始位置和要求时间间隔的日期末尾

```
int CopyHigh(
    string          symbol_name,      // 交易品种名称
    ENUM_TIMEFRAMES timeframe,        // 周期
    datetime        start_time,       // 启动日期和时间
    datetime        stop_time,        // 结束日期和时间
```

```
double          high_array[]    // 复制的目标数组
);
```

参量

symbol_name

[in] 交易品种名称。

timeframe

[in] 周期。

start_pos

[in] 复制的第一元素起始位置。

count

[in] 复制数据点。

start_time

[in] 复制第一个元素的起始时间。

stop_time

[in] 字节时间，与复制最后元素相一致。

high_array[]

[out] [双精度](#) 类型数组。

返回值

返回数组中元素数量或者[错误](#)情况下-1。

注释

如果要求的数据的全部间隔没有在可行数据的服务器中，函数返回-1，如果外部数据 [TERMINAL_MAXBARS](#)（图表中的最大字节量）是要求的，函数也会返回-1。

当向指标要求数据，如果要求的时序列没有建立或者需要从服务器上下载，函数很快会返回-1，但是下载/建立的过程将会开始。

当EA交易或脚本要求数据时，[来自服务器的下载](#)会开始，如果终端本地没有这些数据，要求建立的时序列就会开始，如果数据可以从本地历史建立但尚未准备好。函数在超时期满时，会返回一定数量的准备好的数据，但是历史下载会继续，在下一个类似要求函数中会返回更多数据。

当在日期指定范围内要求数据，只有间隔中的数据才能返回，间隔建立并指到秒。这表示，任意字节的开始时间，返回的值（成交量，传播，指标缓冲区的值，开仓价，最高价，最低价，收盘价或者开仓时间）是间隔要求范围内的。

因此，如果当前日期是星期六，想要复制指定的一周的时间表start_time=Last_Tuesday 和 stop_time=Last_Friday，函数的返回值是0，因为每周开盘时间是星期日，但是一周字节不能分成特殊的间隔。

如果需要返回与当前未完成字节相类似的值，可以调用指定的 start_pos=0 和 count=1第一种形式。

示例：

```
#property copyright "2009, MetaQuotes Software Corp."
#property link      "http://www.mql5.com"
```



```

#property version      "1.00"

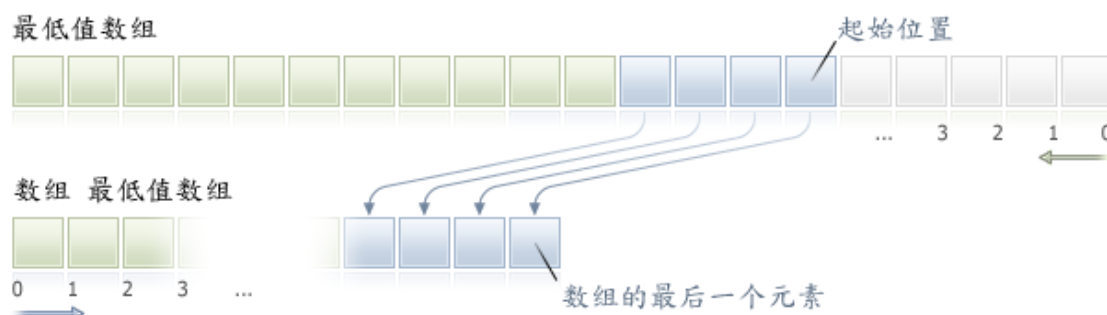
#property description "An example for output of the High[i] and Low[i]"
#property description "for a random chosen bars"

double High[],Low[];
//+-----+
//| 为指定柱指数获得低价                                |
//+-----+
double iLow(string symbol,ENUM_TIMEFRAMES timeframe,int index)
{
    double low=0;
    ArraySetAsSeries(Low,true);
    int copied=CopyLow(symbol,timeframe,0,Bars(symbol,timeframe),Low);
    if(copied>0 && index<copied) low=Low[index];
    return(low);
}
//+-----+
//| 为指定柱指数获得高价                                |
//+-----+
double iHigh(string symbol,ENUM_TIMEFRAMES timeframe,int index)
{
    double high=0;
    ArraySetAsSeries(High,true);
    int copied=CopyHigh(symbol,timeframe,0,Bars(symbol,timeframe),High);
    if(copied>0 && index<copied) high=High[index];
    return(high);
}
//+-----+
//| 专家订单号函数                                    |
//+-----+
void OnTick()
{
    //--- 每个订单号为带有指数的柱输出高低值 ,
    //--- 等同第二订单号
    datetime t=TimeCurrent();
    int sec=t%60;
    printf("High[%d] = %G  Low[%d] = %G",
        sec,iHigh(Symbol(),0,sec),
        sec,iLow(Symbol(),0,sec));
}

```

CopyLow

函数在特定量中获得指定交易品种周期的特定最低字节价格数据到low_array数据中。元素注释的命令是从现在到过去，例如，从当前位置0开始表示当前字节。



当复制未知数量数据时，推荐使用[动态数组](#)为目标数组，因为如果要求数据账户少于（或多于）目标数值的长度，函数试图重新分配内存以完全适应要求数据。

如果知道所需复制的数据数量，为了阻止内存过多分配，最好使用 [静态分配缓冲区](#)。

无论目标数组- as_series=true 或者 as_series=false的属性是什么，数据就会复制，以至于最旧的数组元素会在记忆内存开始中放置。有3个变量函数可供调用。

调用第一位置和要求元素的数量

```
int CopyLow(
    string          symbol_name,      // 交易品种名称
    ENUM_TIMEFRAMES timeframe,       // 周期
    int             start_pos,        // 启动位置
    int             count,            // 复制的数据计算
    double          low_array[]       // 复制的目标数组
);
```

调用起始位置日期和要求元素数量

```
int CopyLow(
    string          symbol_name,      // 交易品种名称
    ENUM_TIMEFRAMES timeframe,       // 周期
    datetime        start_time,       // 启动日期和时间
    int             count,            // 复制的数据计算
    double          low_array[]       // 复制的目标数组
);
```

调用起始位置和要求时间间隔的日期末尾

```
int CopyLow(
    string          symbol_name,      // 交易品种名称
    ENUM_TIMEFRAMES timeframe,       // 周期
    datetime        start_time,       // 启动日期和时间
    datetime        stop_time,        // 结束日期和时间
```

```
double          low_array[]    // 复制的目标数组
);
```

参量

symbol_name

[in] 交易品种。

timeframe

[in] 周期。

start_pos

[in] 复制的第一元素起始位置。

count

[in] 复制数据点。

start_time

[in] 字节时间，复制第一元素相一致。

stop_time

[in] 字节时间，复制的最后元素相一致。

low_array[]

[out] [双精度](#) 类型数组。

返回值

返回数组中元素数量或者 [错误](#)情况下-1。

注释

如果要求的数据的全部间隔没有在可行数据的服务器中，函数返回-1，如果外部数据 [TERMINAL_MAXBARS](#)（图表中的最大字节量）是要求的，函数也会返回-1。

当向指标要求数据，如果要求的时序列没有建立或者需要从服务器上下载，函数很快会返回-1，但是下载/建立的过程将会开始。

当EA交易或脚本要求数据时，[来自服务器的下载](#)会开始，如果终端本地没有这些数据，要求建立的时序列就会开始，如果数据可以从本地历史建立但尚未准备好。函数在超时期满时，会返回一定数量的准备好的数据，但是历史下载会继续，在下一个类似要求函数中会返回更多数据。

当在日期指定范围内要求数据，只有间隔中的数据才能返回，间隔建立并指到秒。这表示，任意字节的开始时间，返回的值（成交量，传播，指标缓冲区的值，开仓价，最高价，最低价，收盘价或者开仓时间）是间隔要求范围内的。

因此，如果当前日期是星期六，想要复制指定的一周的时间表start_time=Last_Tuesday 和 stop_time=Last_Friday，函数的返回值是0，因为每周开盘时间是星期日，但是一周字节不能分成特殊的间隔。

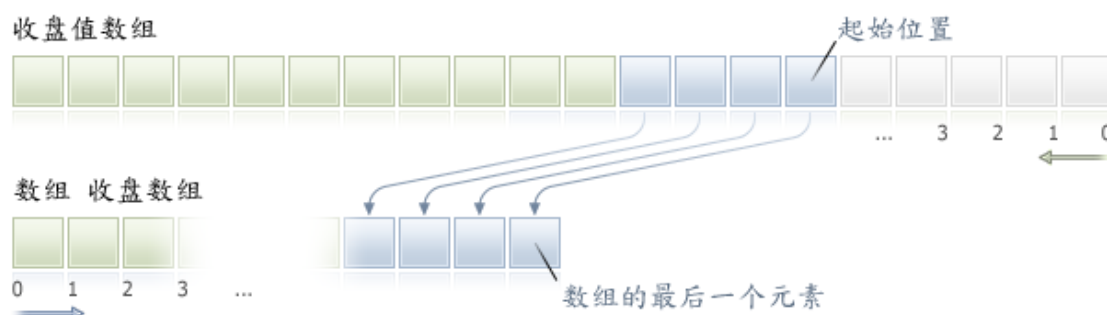
如果需要返回与当前未完成字节相类似的值，可以调用指定的 start_pos=0 和 count=1第一种形式。

另见

[复制最高价](#)

CopyClose

函数在特定量中获得指定交易品种周期的特定收盘价格数据到close_array数据中。元素注释的命令是从现在到过去，例如，从当前位置0开始表示当前字节。



当复制未知数量数据时，推荐使用[动态数组](#)为目标数组，因为如果要求数据账户少于（或多于）目标数值的长度，函数试图重新分配内存以完全适应要求数据。

如果知道所需复制的数据数量，为了阻止内存过多分配，最好使用 [静态分配缓冲区](#)。

无论目标数组- as_series=true 或者 as_series=false的属性是什么，数据就会复制，以至于最旧的数组元素会在记忆内存开始中放置。有3个变量函数可供调用。

调用第一位置和要求元素的数量

```
int CopyClose(
    string          symbol_name,      // 交易品种名称
    ENUM_TIMEFRAMES timeframe,       // 周期
    int             start_pos,        // 启动位置
    int             count,            // 复制的数据计算
    double          close_array[]     // 复制的目标数组
);
```

调用起始位置日期和要求元素数量

```
int CopyClose(
    string          symbol_name,      // 交易品种名称
    ENUM_TIMEFRAMES timeframe,       // 周期
    datetime        start_time,       // 启动日期和时间
    int             count,            // 复制的数据计算
    double          close_array[]     // 复制的目标数组
);
```

调用起始位置和要求时间间隔的日期末尾

```
int CopyClose(
    string          symbol_name,      // 交易品种名称
    ENUM_TIMEFRAMES timeframe,       // 周期
    datetime        start_time,       // 启动日期和时间
    datetime        stop_time,        // 结束日期和时间
```

```
double      close_array[]    // 复制的目标数组
);
```

参量

symbol_name

[in] 交易品种名称。

timeframe

[in] 周期。

start_pos

[in] 复制第一元素起始位置。

count

[in] 复制日期点。

start_time

[in] 复制第一元素起始位置。

stop_time

[in] 字节时间，复制最后元素相一致。

close_array[]

[out] [双精度](#) 类型数组。

返回值

返回数组中元素数量或者[错误](#)情况下-1。

注释

如果要求的数据的全部间隔没有在可行数据的服务器中，函数返回-1，如果外部数据 [TERMINAL_MAXBARS](#)（图表中的最大字节量）是要求的，函数也会返回-1。

当向指标要求数据，如果要求的时序列没有建立或者需要从服务器上下载，函数很快会返回-1，但是下载/建立的过程将会开始。

当EA交易或脚本要求数据时，[来自服务器的下载](#)会开始，如果终端本地没有这些数据，要求建立的时序列就会开始，如果数据可以从本地历史建立但尚未准备好。函数在超时期满时，会返回一定数量的准备好的数据，但是历史下载会继续，在下一个类似要求函数中会返回更多数据。

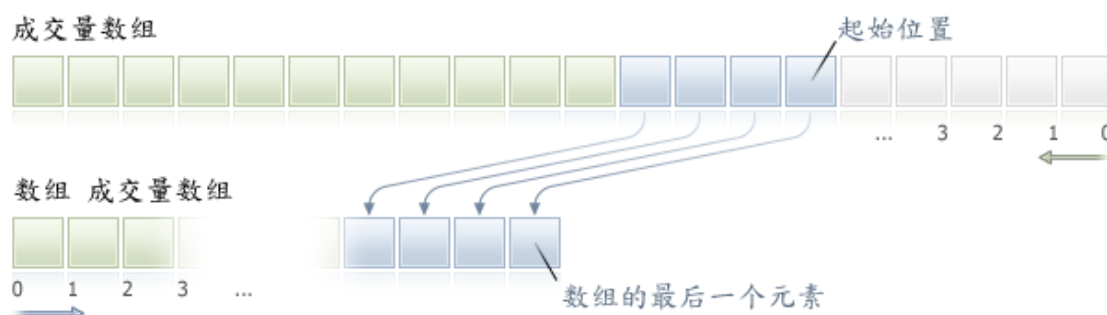
当在日期指定范围内要求数据，只有间隔中的数据才能返回，间隔建立并指到秒。这表示，任意字节的开始时间，返回的值（成交量，传播，指标缓冲区的值，开仓价，最高价，最低价，收盘价或者开仓时间）是间隔要求范围内的。

因此，如果当前日期是星期六，想要复制指定的一周的时间表start_time=Last_Tuesday 和 stop_time=Last_Friday，函数的返回值是0，因为每周开盘时间是星期日，但是一周字节不能分成特殊的间隔。

如果需要返回与当前未完成字节相类似的值，可以调用指定的 start_pos=0 和 count=1第一种形式。

CopyTickVolume

函数在特定量中获得指定交易品种周期的特定点成交量数据到volume_array数据中。元素注释的命令是从现在到过去，例如，从当前位置0开始表示当前字节。



当复制未知数量数据时，推荐使用[动态数组](#)为目标数组，因为如果要求数据账户少于（或多于）目标数值的长度，函数试图重新分配内存以完全适应要求数据。

如果知道所需复制的数据数量，为了阻止内存过多分配，最好使用 [静态分配缓冲区](#)。

无论目标数组- as_series=true 或者 as_series=false的属性是什么，数据就会复制，以至于最旧的数组元素会在记忆内存开始中放置。有3个变量函数可供调用。

调用第一位置和要求元素的数量

```
int CopyTickVolume(
    string      symbol_name,      // 交易品种名称
    ENUM_TIMEFRAMES timeframe,    // 周期
    int         start_pos,        // 启动位置
    int         count,            // 复制的数据计算
    long        volume_array[]    // 订单号交易量的目标数组
);
```

调用起始位置日期和要求元素数量

```
int CopyTickVolume(
    string      symbol_name,      // 交易品种名称
    ENUM_TIMEFRAMES timeframe,    // 周期
    datetime    start_time,       // 启动日期和时间
    int         count,            // 复制的数据计算
    long        volume_array[]    // 订单号交易量的目标数组
);
```

调用起始位置和要求时间间隔的日期末尾

```
int CopyTickVolume(
    string      symbol_name,      // 交易品种名称
    ENUM_TIMEFRAMES timeframe,    // 周期
    datetime    start_time,       // 开始日期和时间
    datetime    stop_time,        // 结束日期和时间
);
```

```
long          volume_array[]    // 订单号交易量的目标数组
);
```

参量

symbol_name

[in] 交易品种名称。

timeframe

[in] 周期。

start_pos

[in] 复制的第一元素开始位置。

count

[in] 复制数据点。

start_time

[in] 复制第一元素起始时间。

stop_time

[in] 字节时间，复制最后元素相一致。

volume_array[]

[out] [长整型](#) 类型数组

返回值

返回数组中元素数量或者[错误](#)情况下-1。

注释

如果要求的数据的全部间隔没有在可行数据的服务器中，函数返回-1，如果外部数据 [TERMINAL_MAXBARS](#)（图表中的最大字节量）是要求的，函数也会返回-1。

当向指标要求数据，如果要求的时序列没有建立或者需要从服务器上下载，函数很快会返回-1，但是下载/建立的过程将会开始。

当EA交易或脚本要求数据时，[来自服务器的下载](#)会开始，如果终端本地没有这些数据，要求建立的时序列就会开始，如果数据可以从本地历史建立但尚未准备好。函数在超时期满时，会返回一定数量的准备好的数据，但是历史下载会继续，在下一个类似要求函数中会返回更多数据。

当在日期指定范围内要求数据，只有间隔中的数据才能返回，间隔建立并指到秒。这表示，任意字节的开始时间，返回的值（成交量，传播，指标缓冲区的值，开仓价，最高价，最低价，收盘价或者开仓时间）是间隔要求范围内的。

因此，如果当前日期是星期六，想要复制指定的一周的时间表start_time=Last_Tuesday 和 stop_time=Last_Friday，函数的返回值是0，因为每周开盘时间是星期日，但是一周字节不能分成特殊的间隔。

如果需要返回与当前未完成字节相类似的值，可以调用指定的 start_pos=0 和 count=1 第一种形式。

示例：

```
#property indicator_separate_window
#property indicator_buffers 1
```

```

#property indicator_plots 1
//--- 订单号交易量图
#property indicator_label1 "TickVolume"
#property indicator_type1 DRAW_HISTOGRAM
#property indicator_color1 C'143,188,139'
#property indicator_style1 STYLE_SOLID
#property indicator_width1 1
//--- 输入参数
input int bars=3000;
//--- 指标缓冲区
double TickVolumeBuffer[];
//+-----+
//| 自定义指标初始化函数 |
//+-----+
void OnInit()
{
//--- 指标缓冲区绘图
SetIndexBuffer(0,TickVolumeBuffer,INDICATOR_DATA);
IndicatorSetInteger(INDICATOR_DIGITS,0);
//---
}
//+-----+
//| 自定义指标重复函数 |
//+-----+
int OnCalculate(const int rates_total,
               const int prev_calculated,
               const datetime &time[],
               const double &open[],
               const double &high[],
               const double &low[],
               const double &close[],
               const long &tick_volume[],
               const long &volume[],
               const int &spread[])
{
//---
if(prev_calculated==0)
{
long timeseries[];
ArraySetAsSeries(timeseries,true);
int prices=CopyTickVolume(Symbol(),0,0,bars,timeseries);
for(int i=0;i<rates_total-prices;i++) TickVolumeBuffer[i]=0.0;
for(int i=0;i<prices;i++) TickVolumeBuffer[rates_total-1-i]=timeseries[prices-1-i];
Print("We have received the following number of TickVolume values: "+prices);
}
else
{
long timeseries[];

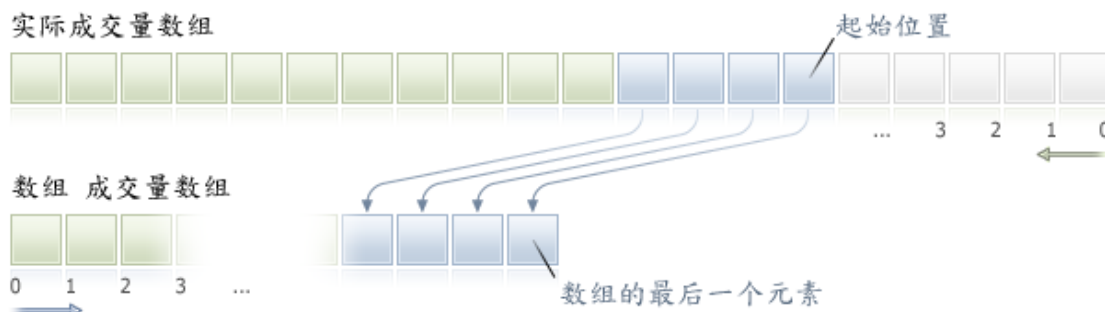
```



```
int prices=CopyTickVolume(Symbol(),0,0,1,timeseries);
TickVolumeBuffer[rates_total-1]=timeseries[0];
}
//--- 为下次调用返回prev_calculated值
return(rates_total);
}
```

CopyRealVolumn

函数在特定量中获得指定交易品种周期的特定交易成交量数据到volume_array数据中。元素注释的命令是从现在到过去，例如，从当前位置0开始表示当前字节。



当复制未知数量数据时，推荐使用[动态数组](#)为目标数组，因为如果要求数据账户少于（或多于）目标数值的长度，函数试图重新分配内存以完全适应要求数据。

如果知道所需复制的数据数量，为了阻止内存过多分配，最好使用 [静态分配缓冲区](#)。

无论目标数组- as_series=true 或者 as_series=false的属性是什么，数据就会复制，以至于最旧的数组元素会在记忆内存开始中放置。有3个变量函数可供调用。

调用第一位置和要求元素的数量

```
int CopyRealVolume(
    string      symbol_name,      // 交易品种名称
    ENUM_TIMEFRAMES timeframe,    // 周期
    int         start_pos,        // 启动位置
    int         count,            // 复制的数据计算
    long        volume_array[]    // 交易量值的目标数组
);
```

调用起始位置日期和要求元素数量

```
int CopyRealVolume(
    string      symbol_name,      // 交易品种名称
    ENUM_TIMEFRAMES timeframe,    // 周期
    datetime    start_time,       // 启动日期和时间
    int         count,            // 复制的数据计算
    long        volume_array[]    // 交易量值的目标数组
);
```

调用起始位置和要求时间间隔的日期末尾

```
int CopyRealVolume(
    string      symbol_name,      // 交易品种名称
    ENUM_TIMEFRAMES timeframe,    // 周期
    datetime    start_time,       // 启动日期和时间
    datetime    stop_time,        // 结束日期和时间
```

```
long          volume_array[]    // 交易量值的目标数组
);
```

参量

symbol_name

[in] 交易品种名称。

timeframe

[in] 周期。

start_pos

[in] 复制第一元素起始位置。

count

[in] 复制数据点。

start_time

[in] 复制第一元素起始时间。

stop_time

[in] 字节时间，复制最后元素相一致。

volume_array[]

[out] [长整型](#) 类型数组

返回值

返回数组中元素数量或者[错误](#)情况下-1。

注释

如果要求的数据的全部间隔没有在可行数据的服务器中，函数返回-1，如果外部数据[TERMINAL_MAXBARS](#)（图表中的最大字节量）是要求的，函数也会返回-1。

当向指标要求数据，如果要求的时序列没有建立或者需要从服务器上下载，函数很快会返回-1，但是下载/建立的过程将会开始。

当EA交易或脚本要求数据时，[来自服务器的下载](#)会开始，如果终端本地没有这些数据，要求建立的时序列就会开始，如果数据可以从本地历史建立但尚未准备好。函数在超时期满时，会返回一定数量的准备好的数据，但是历史下载会继续，在下一个类似要求函数中会返回更多数据。

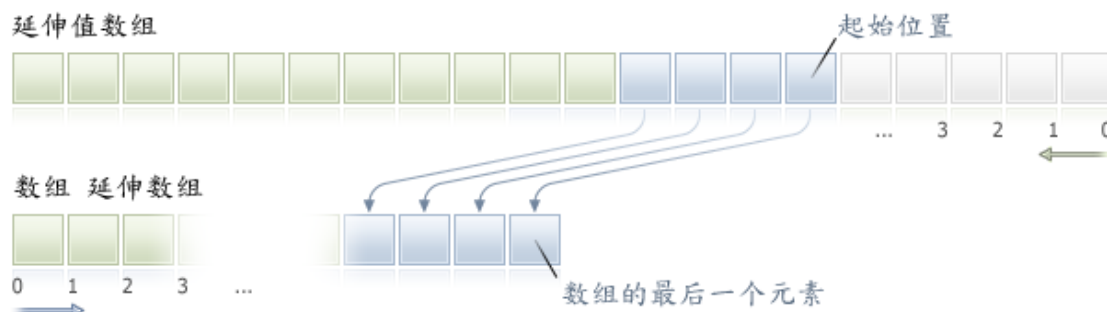
当在日期指定范围内要求数据，只有间隔中的数据才能返回，间隔建立并指到秒。这表示，任意字节的开始时间，返回的值（成交量，传播，指标缓冲区的值，开仓价，最高价，最低价，收盘价或者开仓时间）是间隔要求范围内的。

因此，如果当前日期是星期六，想要复制指定的一周的时间表start_time=Last_Tuesday 和 stop_time=Last_Friday，函数的返回值是0，因为每周开盘时间是星期日，但是一周字节不能分成特殊的间隔。

如果需要返回与当前未完成字节相类似的值，可以调用指定的 start_pos=0 和 count=1第一种形式。

CopySpread

函数在特定量中获得指定交易品种周期的特定点差值数据到spread_array数据中。元素注释的命令是从现在到过去，例如，从当前位置0开始表示当前字节。



当复制未知数量数据时，推荐使用[动态数组](#)为目标数组，因为如果要求数据账户少于（或多于）目标数值的长度，函数试图重新分配内存以完全适应要求数据。

如果知道所需复制的数据数量，为了阻止内存过多分配，最好使用[静态分配缓冲区](#)。

无论目标数组- as_series=true 或者 as_series=false的属性是什么，数据就会复制，以至于最旧的数组元素会在记忆内存开始中放置。有3个变量函数可供调用。

调用第一位置和要求元素的数量

```
int CopySpread(
    string          symbol_name,      // 交易品种名称
    ENUM_TIMEFRAMES timeframe,       // 周期
    int             start_pos,        // 启动位置
    int             count,            // 复制的数据计算
    int             spread_array[]    // 点差值的目标数组
);
```

调用起始位置日期和要求元素数量

```
int CopySpread(
    string          symbol_name,      // 交易品种名称
    ENUM_TIMEFRAMES timeframe,       // 周期
    datetime        start_time,       // 启动日期和时间
    int             count,            // 复制的数据计算
    int             spread_array[]    // 点差值的目标数组
);
```

调用起始位置和要求时间间隔的日期末尾

```
int CopySpread(
    string          symbol_name,      // 交易品种名称
    ENUM_TIMEFRAMES timeframe,       // 周期
    datetime        start_time,       // 启动日期和时间
    datetime        stop_time,        // 结束日期和时间
```

```
int          spread_array[]    // 点差值的目标数组
);
```

参量

symbol_name

[in] 交易品种名称。

timeframe

[in] 周期。

start_pos

[in] 复制第一元素起始位置。

count

[in] 复制数据点。

start_time

[in] 复制第一元素起始位置。

stop_time

[in] 字节时间，复制最后元素相一致。

spread_array[]

[out] [整型](#) 类型数组

返回值

返回数组中元素数量或者[错误](#)情况下-1。

注释

如果要求的数据的全部间隔没有在可行数据的服务器中，函数返回-1，如果外部数据[TERMINAL_MAXBARS](#)（图表中的最大字节量）是要求的，函数也会返回-1。

当向指标要求数据，如果要求的时序列没有建立或者需要从服务器上下载，函数很快会返回-1，但是下载/建立的过程将会开始。

当EA交易或脚本要求数据时，[来自服务器的下载](#)会开始，如果终端本地没有这些数据，要求建立的时序列就会开始，如果数据可以从本地历史建立但尚未准备好。函数在超时期满时，会返回一定数量的准备好的数据，但是历史下载会继续，在下一个类似要求函数中会返回更多数据。

当在日期指定范围内要求数据，只有间隔中的数据才能返回，间隔建立并指到秒。这表示，任意字节的开始时间，返回的值（成交量，传播，指标缓冲区的值，开仓价，最高价，最低价，收盘价或者开仓时间）是间隔要求范围内的。

因此，如果当前日期是星期六，想要复制指定的一周的时间表 `start_time=Last_Tuesday` 和 `stop_time=Last_Friday`，函数的返回值是0，因为每周开盘时间是星期日，但是一周字节不能分成特殊的间隔。

如果需要返回与当前未完成字节相类似的值，可以调用指定的 `start_pos=0` 和 `count=1` 第一种形式。

示例：

```
#property indicator_separate_window
#property indicator_buffers 1
```

```

#property indicator_plots 1
//--- 点差图
#property indicator_label1 "Spread"
#property indicator_type1 DRAW_HISTOGRAM
#property indicator_color1 clrRed
#property indicator_style1 STYLE_SOLID
#property indicator_width1 1
//--- 输入参数
input int bars=3000;
//--- 指标缓冲区
double SpreadBuffer[];
//+-----+
//| 自定义指标初始化函数 |
//+-----+
void OnInit()
{
//--- 指标缓冲区绘图
SetIndexBuffer(0,SpreadBuffer,INDICATOR_DATA);
IndicatorSetInteger(INDICATOR_DIGITS,0);
//---
}
//+-----+
//| 自定义指标重复函数 |
//+-----+
int OnCalculate(const int rates_total,
               const int prev_calculated,
               const datetime &time[],
               const double &open[],
               const double &high[],
               const double &low[],
               const double &close[],
               const long &tick_volume[],
               const long &volume[],
               const int &spread[])
{
//---
if(prev_calculated==0)
{
int spread_int[];
ArraySetAsSeries(spread_int,true);
int spreads=CopySpread(Symbol(),0,0,bars,spread_int);
Print("We have received the following number of Spread values: ",spreads);
for (int i=0;i<spreads;i++)
{
SpreadBuffer[rates_total-1-i]=spread_int[i];
if(i<=30) Print("spread["+i+"] = ",spread_int[i]);
}
}
}

```

```
else
{
    double Ask,Bid;
    Ask=SymbolInfoDouble(Symbol(),SYMBOL_ASK);
    Bid=SymbolInfoDouble(Symbol(),SYMBOL_BID);
    Comment("Ask = ",Ask," Bid = ",Bid);
    SpreadBuffer[rates_total-1]=(Ask-Bid)/Point();
}
//--- 为下次调用返回prev_calculated值
return(rates_total);
}
```

图表操作

这些是工作图表函数。只在EA交易和脚本中允许操作所有图表。

函数	功能
ChartApplyTemplate	在图表中应用指定文件中的特定模板。
ChartSaveTemplate	
ChartWindowFind	返回绘画指标的子窗口数量。
ChartOpen	打开指定交易品种和周期的新图表。
ChartClose	关闭指定图表。
ChartFirst	返回客户端第一图表的ID。
ChartNext	返回指定图表旁边图表的ID。
ChartSymbol	返回指定图表的交易品种名称。
ChartPeriod	返回指定图表的周期值。
ChartRedraw	调用指定图表的被迫重画。
ChartSetDouble	设置指定图表相关属性的双精度值。
ChartSetInteger	设置指定图表相关属性的整数值（日期时间，整型，颜色，布尔或者字符型）。
ChartSetString	设置指定图表相关属性的字符串值。
ChartGetDouble	返回指定图表的双精度值属性。
ChartGetInteger	返回指定图表的整数值属性。
ChartGetString	返回指定图表的字符串属性。
ChartNavigate	通过指定的关系柱完成指定图表到图表中的指定位置的转换。
ChartID	返回当前图表的ID。
ChartIndicatorAdd	用指定处理程序在指定图表窗口添加指标。
ChartIndicatorDelete	
ChartIndicatorName	
ChartIndicatorsTotal	
ChartWindowOnDropped	返回图表子窗口EA交易或者脚本下跌到的数（指数）。
ChartPriceOnDropped	返回图表点EA交易或者脚本下跌到的价格坐标。

<u>ChartTimeOnDropped</u>	返回图表点EA交易或者脚本下跌到的时间坐标。
<u>ChartXOnDropped</u>	返回图表点EA交易或者脚本下跌到的X坐标。
<u>ChartYOnDropped</u>	返回图表点EA交易或者脚本下跌到的Y坐标。
<u>ChartSetSymbolPeriod</u>	改变交易品种值和指定图表周期。
<u>ChartScreenShot</u>	以gif格式提供图表当前状态屏幕截图。

ChartApplyTemplate

在图表中应用指定文件中的特定模板。

```
bool ChartApplyTemplate(
    long          chart_id,      // 图表 ID
    const string  filename      // 模板文件名
);
```

参量

chart_id

[in] 图表 ID. 0 意味着当前图表。

filename

[in] 包括模板的文件名。

返回值

如果模板成功应用，函数返回true，否则返回false。若要获得错误信息，调用[GetLastError\(\)](#)函数。

注释

成功加载新模板时EA交易会被卸载且不能再继续操作。

```

MQL5

ChartSetInteger( ) :

OHLC;

Bid-
Ask-

( Last ) ;

( Stop Loss Take Profit ) .

```

ChartApplyTemplate()

mql5-
ChartApplyTemplate()

- `"\" (" \") ,`
`— —`
`— \MQL5,`
- `EX5-`
`ChartApplyTemplate() ;`
- `—`
`— \Profiles\Templates\.`
`—`
`— MetaTrader 5,`
`— —`

TerminalInfoString()

```
//--- каталог из которой запущен терминал
string terminal_path=TerminalInfoString(TERMINAL_PATH);
Print("Каталог терминала:",terminal_path);
//--- каталог данных терминала, в котором находится папка MQL5 с советниками и индикаторами
string terminal_data_path=TerminalInfoString(TERMINAL_DATA_PATH);
Print("Каталог данных терминала:",terminal_data_path);
```

```
//--- шаблон ищем в папке каталог_данных_терминала\MQL5\
ChartApplyTemplate(0,"\\first_template.tpl"))

//--- шаблон ищем в папке каталог_исполняемого_EX5_файла\, затем в папке каталог_данных_терминала\MQL5\
ChartApplyTemplate(0,"second_template.tpl"))

//--- шаблон ищем в папке каталог_исполняемого_EX5_файла\My_templates\, затем в папке каталог_данных_терминала\MQL5\
ChartApplyTemplate(0,"My_templates\\third_template.tpl"))
```

EX5.

示例：

```
//+-----+
//| 脚本程序启动函数 |
//+-----+
void OnStart()
{
//--- 打开并且应用 'my_template.tpl' 模板
if(!ChartApplyTemplate(0,"my_template"))
{
//--- 若无法应用模板，那么
Print("Unable to apply 'my_template', error ",GetLastError());
ResetLastError();
//--- 只搜索 'my_template' 文件，无 'tpl' 扩展名
if(FileExists("my_template"))
Print("File 'my_template' exists, but 'my_template.tpl' file is needed");
else
Print("'my_template.tpl' file is not found in "
+TerminalInfoString(TERMINAL_PATH)
+"\\MQL5\\Files");

//--- 现在返回带有 'tpl' 扩展名的准确文件名
if(ChartApplyTemplate(0,"my_template.tpl"))
Print("'my_template.tpl' successfully applied ");
else
Print("Unable to apply 'my_template.tpl', error ",GetLastError());

}
else
{
//--- 模板存储于无扩展名的文件中
Print("'my_template' successfully applied - file without extension");
}
}
```

另见

ChartSaveTemplate

```
bool ChartSaveTemplate(
    long      chart_id,      // идентификатор графика
    const string filename    // имя файла для сохранения шаблона
);
```

```
chart_id
[in]                                . 0

filename
[in]                                ".tpl"

                                \Profiles\Templates\

                                true,
                                false.
```

[GetLastError\(\)](#).

```
//+-----+
//|                                Test_ChartSaveTemplate.mq5 |
//|                                Copyright 2011, MetaQuotes Software Corp. |
//|                                http://www.mql5.com |
//+-----+
#property copyright "Copyright 2011, MetaQuotes Software Corp."
#property link      "http://www.mql5.com"
#property version   "1.00"
#property script_show_inputs
//--- input parameters
input string        symbol="GBPUSD"; // символ нового графика
input ENUM_TIMEFRAMES period=PERIOD_H3; // таймфрейм нового графика
//+-----+
```

```

//| Script program start function |
//+-----+
void OnStart()
{
//--- сначала набросим на график индикаторы
    int handle;
//--- подготовим индикатор к использованию
    if(!PrepareZigzag(NULL,0,handle)) return; // сразу обломались и выходим
//--- набросим индикатор на текущий график, но... в отдельное окно :)
    if(!ChartIndicatorAdd(0,1,handle))
    {
        PrintFormat("Не удалось добавить на график %s/%s индикатор с хэндлом=%d. Код оши
            _Symbol,
            EnumToString(_Period),
            handle,
            GetLastError());
        //--- досрочно прекращаем работу программы
        return;
    }
//--- прикажем графику обновиться, чтобы сразу увидеть наброшенный индикатор
    ChartRedraw();
//--- найдем два последних перелома зигзага
    double two_values[];
    datetime two_times[];
    if(!GetLastTwoFractures(two_values,two_times,handle))
    {
        PrintFormat("Не удалось найти два последних перелома в индикторе Zigzag!");
        //--- досрочно прекращаем работу программы
        return;
    }
//--- теперь набросим канал стандартного отклонения
    string channel="StdDeviation Channel";
    if(!ObjectCreate(0,channel,OBJ_STDDEVCHANNEL,0,two_times[1],0))
    {
        PrintFormat("Не удалось создать объект %s. Код ошибки %d",
            EnumToString(OBJ_STDDEVCHANNEL),GetLastError());
        return;
    }
    else
    {
        //--- канал создан, доопределим вторую опорную точку
        ObjectSetInteger(0,channel,OBJPROP_TIME,1,two_times[0]);
        //--- зададим каналу текст всплывающей подсказки
        ObjectSetString(0,channel,OBJPROP_TOOLTIP,"Demo from MQL5 Help");
        //--- обновим график
        ChartRedraw();
    }
//--- сохраним полученную красоту в шаблон

```

```

ChartSaveTemplate(0,"StdDevChannelOnZigzag");
//--- откроем новый график и набросим на него сохраненный шаблон
long new_chart=ChartOpen(symbol,period);
//--- включим показ всплывающих подсказок для графических объектов
ChartSetInteger(new_chart,CHART_SHOW_OBJECT_DESCR,true);
if(new_chart!=0)
{
    //--- набросим на новый график сохраненный шаблон
    ChartApplyTemplate(new_chart,"StdDevChannelOnZigzag");
}
sleep(10000);
}
//+-----+
//| создает хэндл зигзага и обеспечивает готовность его данных |
//+-----+
bool PrepareZigzag(string sym,ENUM_TIMEFRAMES tf,int &h)
{
    ResetLastError();
    h=iCustom(sym,tf,"Examples\\Zigzag");
    if(h==INVALID_HANDLE)
    {
        PrintFormat("%s: Не удалось создать хэндл индикатора Zigzag. Код ошибки %d",
            __FUNCTION__,GetLastError());
        return false;
    }
    //--- при создании хэндла индикатора ему требуется время на расчет значений
    int k=0; // количество попыток чтобы дождаться расчета индикатора
    //--- ждем расчета в цикле, делаем паузу в 50 миллисекунд, если расчет еще не готов
    while(BarsCalculated(h)<=0)
    {
        k++;
        //--- выведем количество попыток
        PrintFormat("%s: k=%d",__FUNCTION__,k);
        //--- подождем 50 миллисекунд, пока индикатор рассчитается
        Sleep(50);
        //--- если сделано больше 100 попыток, то что-то не так
        if(k>100)
        {
            //--- сообщим о проблеме
            PrintFormat("Не удалось рассчитать индикатор за %d попыток!");
            //--- досрочно прекращаем работу программы
            return false;
        }
    }
    //--- все готово, индикатор создан и значения рассчитаны
    return true;
}
//+-----+

```

```

//|  ищет два последних перелома зигзага и помещает в массивы |
//+-----+
bool GetLastTwoFractures(double &get_values[],datetime &get_times[],int handle)
{
    double values[];          // массив для получения значений зигзага
    datetime times[];         // массив для получения времени
    int size=100;             // размер массивов
    ResetLastError();
//--- копируем 100 последних значений индикатора
    int copied=CopyBuffer(handle,0,0,size,values);
//--- проверим сколько скопировалось
    if(copied<100)
    {
        PrintFormat("%s: Не удалось скопировать %d значений индикатора с хэндлом=%d. Ко
            __FUNCTION__,size,handle,GetLastError());
        return false;
    }
//--- определим порядок доступа к массиву как в таймсерии
    ArraySetAsSeries(values,true);
//--- сюда запишем номера баров, на которых найдены переломы
    int positions[];
//--- зададим размеры массивов
    ArrayResize(get_values,3); ArrayResize(get_times,3); ArrayResize(positions,3);
//--- счетчики
    int i=0,k=0;
//--- начинаем поиск переломов
    while(i<100)
    {
        double v=values[i];
        //--- пустые значения нас не интересуют
        if(v!=0.0)
        {
            //--- запомним номер бара
            positions[k]=i;
            //--- запомним значение зигзага на переломе
            get_values[k]=values[i];
            PrintFormat("%s: Zigzag[%d]=%G",__FUNCTION__,i,values[i]);
            //--- увеличим счетчик
            k++;
            //--- если нашли два перелома, то ломаем цикл
            if(k>2) break;
        }
        i++;
    }
//--- определим порядок доступа к массивам как в таймсерии
    ArraySetAsSeries(times,true); ArraySetAsSeries(get_times,true);
    if(CopyTime(_Symbol,_Period,0,size,times)<=0)
    {

```



```

        PrintFormat("%s: Не удалось скопировать %d значений из CopyTime(). Код ошибки %d",
                    __FUNCTION__, size, GetLastError());
        return false;
    }

    ///--- найдем время открытия баров, на которых были 2 последних перелома
    get_times[0]=times[positions[1]]; // предпоследнее значение будет записано первым
    get_times[1]=times[positions[2]]; // третье с конца значение будет вторым переломом
    PrintFormat("%s: первый=%s, второй=%s", __FUNCTION__, TimeToString(get_times[1]), TimeToString(get_times[0]));
    ///--- все получилось
    return true;
}

```

[ChartApplyTemplate\(\)](#), _____

ChartWindowFind

函数返回画指标的子窗口数。该函数有两个变体。

1. 函数在指定图表中搜索带有指定的指标“缩写名”子窗口（缩写名在子窗口的左上部显示），成功的话返回子窗口数。

```
int ChartWindowFind(
    long      chart_id,           // 图表标识符
    string     indicator_shortcode // 短指标名，见 INDICATOR\_SHORTNAME
```

2. 函数必须从自定义指标中调用。返回指标工作的子窗口数。

```
int ChartWindowFind();
```

参量

chart_id

[in] 图表 ID. 0 表示当前图表。

indicator_shortcode

[in] 指标缩写名。

返回值

成功的话返回子窗口数。失败，函数返回-1。

注释

如果函数第二变体（无参量）是从脚本或者EA交易调用，则返回-1。

示例：

```
#property script_show_inputs
//--- 输入参量
input string  shortname="MACD(12,26,9)";
//+-----+
//|  返回带有这个指标的图表窗口号      |
//+-----+
int GetIndicatorSubWindowNumber(long chartID=0,string short_name="")
{
    int window=-1;
//---
    if((ENUM_PROGRAM_TYPE)MQL5InfoInteger(MQL5_PROGRAM_TYPE)==PROGRAM_INDICATOR)
    {
        //--- 函数从指标调用，名称不需要
        window=ChartWindowFind();
    }
    else
    {
        //--- 函数从EA交易或者脚本调用
        window=ChartWindowFind(0,short_name);
        if(window==-1) Print(__FUNCTION__+"(): Error = ",GetLastError());
    }
}
```

```
    }  
    //---  
    return(window);  
    }  
    //+-----+  
    //| 脚本程序启动函数 |  
    //+-----+  
    void OnStart()  
    {  
    //---  
        int window=GetIndicatorSubWindowNumber(0,shortname);  
        if(window!=-1)  
            Print("Indicator "+shortname+" is in the window #"+(string)window);  
        else  
            Print("Indicator "+shortname+" is not found. window = "+(string)window);  
    }
```

另见

[ObjectCreate\(\)](#), [ObjectFind\(\)](#)

ChartOpen

打开指定交易品种和周期的新图表。

```
long ChartOpen(  
    string      symbol,      // 交易品种名称  
    ENUM_TIMEFRAMES period    // 周期  
);
```

参量

symbol

[in] 图表交易品种，[NULL](#) 意味着当前图表交易品种（附上EA交易）。

period

[in] 图表周期（时间表）。可以是[ENUM_TIMEFRAMES](#)值之一。0代表当前图表周期。

返回值

若成功，返回打开图表ID，否则返回0。

注释

程序端同时打开图表的最大值不能超过[CHARTS_MAX](#)值。

ChartFirst

返回客户端第一图表ID。

```
long ChartFirst();
```

返回值

图表 ID。

ChartNext

返回指定图表旁边的图表ID。

```
long ChartNext(  
    long chart_id    // 图表 ID  
);
```

参量

chart_id

[in] 图表 ID. 0 不表示当前图表。0表示“返回第一图表”。

返回值

图表 ID。如果在图表列表末端，返回-1。

示例：

```
//--- 用于图表 ID的变量  
long currChart,prevChart=ChartFirst();  
int i=0,limit=100;  
Print("ChartFirst =",ChartSymbol(prevChart)," ID =",prevChart);  
while(i<limit)// 不允许超过100个打开的窗口  
{  
    currChart=ChartNext(prevChart); // 通过使用之前图表ID获得新图表ID  
    if(currChart<0) break;           // 到达了图表列表末端  
    Print(i,ChartSymbol(currChart)," ID =",currChart);  
    prevChart=currChart;// 为ChartNext()保存当前图表ID  
    i++;// 不要忘记增加计数器  
}
```

ChartClose

关闭指定图表。

```
bool ChartClose(  
    long chart_id=0    // 图表 ID  
);
```

参量

chart_id=0

[in] 图表 ID. 0 意味着当前图表。

返回值

若成功，返回true，否则false。

ChartSymbol

返回指定图表的交易品种名称。

```
string ChartSymbol(  
    long chart_id=0    // 图表 ID  
);
```

参量

chart_id=0

[in] 图表 ID. 0 意味着当前图表。

返回值

如果图表不存在，结果是空字符串。

另见

[ChartSetSymbolPeriod](#)

ChartPeriod

返回指定图表的时间表 [周期](#)。

```
ENUM_TIMEFRAMES ChartPeriod(  
    long chart_id=0    // 图表 ID  
);
```

参量

chart_id=0

[in] 图表 ID. 0 意味着当前图表。

返回值

函数返回[ENUM_TIMEFRAMES](#)值之一，如果图表不存在，返回0。

ChartRedraw

指定图表被迫重画调用此函数。

```
void ChartRedraw(  
    long chart_id=0    // 图表 ID  
);
```

参量

chart_id=0

[in] 图表 ID. 0 意味着当前图表。.

注释

通常更改[物件属性](#)之后使用该函数。

另见

[对象函数](#)

ChartSetDouble

设置指定图表相关属性值。图表属性是[双精度](#)类型。

```
bool ChartSetDouble(  
    long    chart_id,    // 图表 ID  
    int     prop_id,     // 属性 ID  
    double  value        // 值  
);
```

参量

chart_id

[in] 图表 ID. 0 意味着当前图表。

prop_id

[in] 图表属性 ID. 可以是 [ENUM_CHART_PROPERTY_DOUBLE](#) 值中的一个 (除了只读属性)。

value

[in] 属性值。

返回值

成功, 返回 true, 否则返回 false。

ChartSetInteger

设置指定图表相关属性值。图表属性必须是[日期时间](#)，[整型](#)，[颜色](#)，[布尔型](#)或者[字符型](#)。

```
bool ChartSetInteger(  
    long   chart_id,      // 图表 ID  
    int    prop_id,       // 属性 ID  
    long   value          // 值  
);
```

参量

chart_id

[in] 图表 ID. 0 意味着当前图表。

prop_id

[in] 图表属性ID. 可以是[ENUM_CHART_PROPERTY_INTEGER](#)值中的一个（除了只读属性）。

value

[in] 属性值。

返回值

若成功则返回true，否则返回 false。

ChartSetString

设置指定图表的相关属性值。图表属性必须是字符串类型。

```
bool ChartSetString(  
    long   chart_id,      // 图表 ID  
    int    prop_id,       // 属性 ID  
    string str_value      // 值  
);
```

参量

chart_id

[in] 图表 ID. 0 意味着当前图表。

prop_id

[in] 图表属性 ID. 可以是[ENUM_CHART_PROPERTY_STRING](#)值中的一个 (除了只读属性)。

str_value

[in] 属性值字符串。字符串长度不能超过 2045 个字符 (多出的字符会被删除)。

返回值

如果成功返回true, 否则false。

注释

ChartSetString 可以代替[注释](#)函数用作图标中的注释输出。

示例：

```
void OnTick()  
{  
    //---  
    double Ask,Bid;  
    int Spread;  
    Ask=SymbolInfoDouble(Symbol(),SYMBOL_ASK);  
    Bid=SymbolInfoDouble(Symbol(),SYMBOL_BID);  
    Spread=SymbolInfoInteger(Symbol(),SYMBOL_SPREAD);  
    string comment=StringFormat("Printing prices:\nAsk = %G\nBid = %G\nSpread = %d",  
                                Ask,Bid,Spread);  
    ChartSetString(0,CHART_COMMENT,comment);  
}
```

另见

[注释](#), [ChartGetString](#)

ChartGetDouble

返回指定图表的相关属性值。图表属性必须是双精度类型。有两个函数调用变体。

1. 直接返回属性值。

```
double ChartGetDouble(
    long   chart_id,      // 图表 ID
    int    prop_id,       // 属性 ID
    int    sub_window=0   // 子窗口号，如果需要的话
);
```

2. 根据函数成功与否，返回 true 或 false。如果成功，属性值置于引用传递的目标变量double_ var。

```
bool ChartGetDouble(
    long   chart_id,      // 图表 ID
    int    prop_id,       // 属性 ID
    int    sub_window,    // 子窗口号
    double& double_var    // 用于图表属性的目标变量
);
```

参量

chart_id

[in] 图表 ID. 0 意味着当前图表。

prop_id

[in] 图表属性 ID. 该值可以是[ENUM_CHART_PROPERTY_DOUBLE](#)值中之一。

sub_window

[in] 图表子窗口数。对于第一种情况，默认值为0（图表主窗口）。属性大部分情况不需要子窗口数。

double_var

[out] 需求属性双精度类型的目标变量。

返回值

双精度值。

对于第二种调用情况，如果指定属性有效且其值置于double_ var变量，返回true，否则返回 false。若获得额外[错误](#)信息，需要调用函数[GetLastError\(\)](#)。

示例：

```
void OnStart()
{
    double priceMin=ChartGetDouble(0,CHART_PRICE_MIN,0);
    double priceMax=ChartGetDouble(0,CHART_PRICE_MAX,0);
    Print("CHART_PRICE_MIN =",priceMin);
    Print("CHART_PRICE_MAX =",priceMax);
}
```

ChartGetInteger

返回指定图表相关属性值。图表属性必须是日期时间，整型，布尔型类型。有两种函数调用变体。

1. 直接返回属性值。

```
long ChartGetInteger(
    long chart_id,      // 图表 ID
    int prop_id,        // 属性 ID
    int sub_window=0    // 子窗口号，如果需要的话
);
```

2. 按照函数成功与否，返回 true 或者 false。如果成功，属性值置于引用传递的目标变量long_ var。

```
bool ChartGetInteger(
    long chart_id,      // 图表 ID
    int prop_id,        // 属性 ID
    int sub_window,     // 子窗口号
    long& long_var      // 属性目标值
);
```

参量

chart_id

[in] 图表 ID. 0 意味着当前图表。

prop_id

[in] 图表属性 ID. 该值可以是 [ENUM_CHART_PROPERTY_INTEGER](#) 值中的一个。

sub_window

[in] 图表子窗口数。对于第一种情况，默认值为0（图表主窗口）。属性大部分情况都不需要子窗口数。

long_var

[out] 请求属性的长整型目标变量。

返回值

长整型值。

对于第二种情况，如果指定属性有效并且其值存储在long_ var变量中，则返回true，否则返回false。若获得额外错误信息，需要调用 [GetLastError\(\)](#) 函数。

示例：

```
void OnStart()
{
    int height=ChartGetInteger(0,CHART_HEIGHT_IN_PIXELS,0);
    int width=ChartGetInteger(0,CHART_WIDTH_IN_PIXELS,0);
    Print("CHART_HEIGHT_IN_PIXELS =",height,"pixels");
    Print("CHART_WIDTH_IN_PIXELS =",width,"pixels");
}
```

ChartGetString

返回指定图表的相关属性值。图表属性必须是字符串型。有两个函数调用变体。

1. 直接返回属性值。

```
string ChartGetString(
    long   chart_id,      // 图表 ID
    int    prop_id        // 属性 ID
);
```

2. 按照函数成功与否，返回true或者false。如果成功，属性值置于引用传递的目标变量string_ var。

```
bool ChartGetString(
    long   chart_id,      // 图表 ID
    int    prop_id,      // 属性 ID
    string& string_var    // 属性目标值
);
```

参数

chart_id

[in] 图表 ID. 0 意味着当前图表。

prop_id

[in] 图表属性 ID. 该值可以是 [ENUM_CHART_PROPERTY_STRING](#) 值中的一个。

string_var

[out] 请求属性字符串类型的目标变量。

返回值

字符串类型值。

对于第二种调用情况，如果指定属性有效并且其值存储在string_ var变量中返回true，否则返回 false。若要获得额外[错误](#)信息，需要调用[GetLastError\(\)](#) 函数。

注释

ChartGetString 可以使用[注释](#)或者[ChartSetString](#)函数用来读取图表中标绘的注释。

示例：

```
void OnStart()
{
    ChartSetString(0, CHART_COMMENT, "Test comment.\nSecond line.\nThird!");
    ChartRedraw();
    Sleep(1000);
    string comm=ChartGetString(0, CHART_COMMENT);
    Print(comm);
}
```

另见

[注释](#) [-ChartSetString](#)

ChartNavigate

通过指定的关系柱完成指定图表到图表中的指定位置的转换。

```
bool ChartNavigate(
    long  chart_id,    // 图表 ID
    int   position,    // 位置
    int   shift=0      // 移动值
);
```

参量

chart_id

[in] 图表 ID. 0 意味着当前图表。

position

[in] 完成转换的图表位置。可以是[ENUM_CHART_POSITION](#)值中的一个。

shift=0

[in] 转换图表的柱数。正值意味着向右移动（图表末端），负值意味着左移（到图表起点）。0值用来控制图表的起止。

返回值

若成功，返回true，否则返回false。

示例：

```
//+-----+
//| Script program start function |
//+-----+
void OnStart()
{
    //-- получим handle текущего графика
    long handle=ChartID();
    string comm="";
    if(handle>0) // если получилось, дополнительно настроим
    {
        //-- отключим автопрокрутку
        ChartSetInteger(handle,CHART_AUTOSCROLL,false);
        //-- установим отступ правого края графика
        ChartSetInteger(handle,CHART_SHIFT,true);
        //-- отобразим в виде свечей
        ChartSetInteger(handle,CHART_MODE,CHART_CANDLES);
        //-- установить режим отображения тиковых объемов
        ChartSetInteger(handle,CHART_SHOW_VOLUMES,CHART_VOLUME_TICK);

        //-- подготовим текст для вывода в Comment()
        comm="Прокрутим на 10 баров вправо от начала истории";
        //-- выведем комментарий
        Comment(comm);
    }
```

```

//--- прокрутим на 10 баров вправо от начала истории
ChartNavigate(handle, CHART_BEGIN, 10);
//--- получим номер самого первого видимого на графике бара (нумерация как в та
long first_bar=ChartGetInteger(0, CHART_FIRST_VISIBLE_BAR, 0);
//--- добавим символ переноса строки
comm=comm+"\r\n";
//--- дополним комментарий
comm=comm+"Первый бар на графике имеет номер "+IntegerToString(first_bar)+"\r\n";
//--- выведем комментарий
Comment(comm);
//--- подождем 5 секунд, чтобы успеть увидеть, как двигается график
Sleep(5000);

//--- допишем текст комментария
comm=comm+"\r\n"+"Прокрутим на 10 баров влево от правого края графика";
Comment(comm);
//--- прокрутим на 10 баров влево от правого края графика
ChartNavigate(handle, CHART_END, -10);
//--- получим номер самого первого видимого на графике бара (нумерация как в та
first_bar=ChartGetInteger(0, CHART_FIRST_VISIBLE_BAR, 0);
comm=comm+"\r\n";
comm=comm+"Первый бар на графике имеет номер "+IntegerToString(first_bar)+"\r\n";
Comment(comm);
//--- подождем 5 секунд, чтобы успеть увидеть, как двигается график
Sleep(5000);

//--- новый блок прокрутки графика
comm=comm+"\r\n"+"Прокрутим на 300 баров вправо от начала истории";
Comment(comm);
//--- прокрутим на 300 баров вправо от начала истории
ChartNavigate(handle, CHART_BEGIN, 300);
first_bar=ChartGetInteger(0, CHART_FIRST_VISIBLE_BAR, 0);
comm=comm+"\r\n";
comm=comm+"Первый бар на графике имеет номер "+IntegerToString(first_bar)+"\r\n";
Comment(comm);
//--- подождем 5 секунд, чтобы успеть увидеть, как двигается график
Sleep(5000);

//--- новый блок прокрутки графика
comm=comm+"\r\n"+"Прокрутим на 300 баров влево от правого края графика";
Comment(comm);
//--- прокрутим на 300 баров влево от правого края графика
ChartNavigate(handle, CHART_END, -300);
first_bar=ChartGetInteger(0, CHART_FIRST_VISIBLE_BAR, 0);
comm=comm+"\r\n";
comm=comm+"Первый бар на графике имеет номер "+IntegerToString(first_bar)+"\r\n";
Comment(comm);
}

```

```
}
```

ChartID

返回当前图表ID。

```
long ChartID();
```

返回值

[长整型](#) 值。

ChartIndicatorAdd

用指定处理程序在指定图表窗口填加指标。

```
bool ChartIndicatorAdd(  
    long chart_id,           // 图表 ID  
    int sub_window           // 子窗口号  
    int indicator_handle     // 指标处理  
);
```

参量

chart_id

[in] 图表 ID. 0 意味着当前图表。

sub_window

[in] 图表子窗口数。0代表图表主窗口。如果指定不存在的窗口，将会创建新窗口。

indicator_handle

[in] 指标处理程序。

返回值

若成功函数返回true，否则返回false。若要获得额外[错误](#)信息，需要调用[GetLastError\(\)](#) 函数。

[ChartIndicatorDelete\(\)](#), [ChartIndicatorName\(\)](#), [ChartIndicatorsTotal\(\)](#), [iCustom\(\)](#),
[IndicatorCreate\(\)](#)

ChartIndicatorDelete

```
bool ChartIndicatorDelete (
    long      chart_id,           // идентификатор графика
    int       sub_window         // номер подокна
    const string ind_shortcode    // короткое имя индикатора
);
```

```
chart_id
[in] . 0

sub_window
[in] . 0

const ind_shortcode
[in]
    INDICATOR\_SHORTNAME IndicatorSetString\( \).

    ChartIndicatorName\( \).

    true
false. GetLastError\( \).

    iCustom\( \) IndicatorCreate\( \).

    IndicatorRelease\( \).
```

```
IndicatorSetString( )  
INDICATOR__SHORTNAME.  
  
ChartIndicatorDelete  
( )  
  
ChartIndicatorAdd( ), ChartIndicatorName( ), ChartIndicatorsTotal( ), iCustom( ),  
IndicatorCreate( ), IndicatorSetString( )
```


ChartIndicatorName

```
string ChartIndicatorName (
    long  chart_id,      // идентификатор графика
    int   sub_window     // номер подокна
    int   index          // индекс индикатора в списке индикаторов, добавленных к данн
);
```

chart_id

[in] . 0

sub_window

[in] . 0

index

[in]

[ChartIndicatorsTotal\(\)](#).

[INDICATOR_SHORTNAME](#)

[IndicatorSetString\(\)](#).

[ChartIndicatorName\(\)](#).

[GetLastError\(\)](#).

[iCustom\(\)](#) [IndicatorCreate\(\)](#).

[IndicatorRelease\(\)](#).

[IndicatorSetString\(\)](#)
[INDICATOR_SHORTNAME](#).

()

ChartIndicatorDelete

ChartIndicatorAdd(), ChartIndicatorDelete(), ChartIndicatorsTotal(), iCustom(),
IndicatorCreate(), IndicatorSetString()

ChartIndicatorsTotal

```
int ChartIndicatorsTotal(  
    long chart_id,      // идентификатор графика  
    int sub_window      // номер подокна  
);
```

chart_id
[in] . 0

sub_window
[in] . 0

[GetLastError\(\)](#).

[CHART_WINDOWS_TOTAL](#)

[ChartGetInteger\(\)](#).

[ChartIndicatorAdd\(\)](#), [ChartIndicatorDelete\(\)](#), [ChartIndicatorsTotal\(\)](#), [iCustom\(\)](#),
[IndicatorCreate\(\)](#), [IndicatorSetString\(\)](#)

ChartWindowOnDropped

返回图表子窗口EA交易或者脚本下跌到的数（指数）。0代表图表主窗口。

```
int ChartWindowOnDropped();
```

返回值

整型 值。

示例：

```
int myWindow=ChartWindowOnDropped();  
int windowsTotal=ChartGetInteger(0,CHART_WINDOWS_TOTAL);  
Print("Script is running on the window #"+myWindow+  
      ". Total windows on the chart "+ChartSymbol()+":",windowsTotal);
```

另见

[ChartPriceOnDropped](#), [ChartTimeOnDropped](#), [ChartXOnDropped](#), [ChartYOnDropped](#)

ChartPriceOnDropped

返回图表点一致的EA交易或者脚本下跌到的价格坐标。

```
double ChartPriceOnDropped();
```

返回值

[双精度值](#)

示例：

```
double p=ChartPriceOnDropped();  
Print("ChartPriceOnDropped() = ",p);
```

另见

[ChartXOnDropped](#), [ChartYOnDropped](#)

ChartTimeOnDropped

返回图表点一致的EA交易或者脚本下跌到的时间坐标。

```
datetime ChartTimeOnDropped();
```

返回值

[日期时间](#) 类型值。

示例：

```
datetime t=ChartTimeOnDropped();  
Print("Script was dropped on the "+t);
```

另见

[ChartXOnDropped](#), [ChartYOnDropped](#)

ChartXOnDropped

返回EA交易或者脚本下跌到的图表点x坐标。

```
int ChartXOnDropped();
```

返回值

X 坐标值

注释

自左向右的X 轴方向。

示例：

```
int X=ChartXOnDropped();  
int Y=ChartYOnDropped();  
Print(" (X,Y) = (" +X+", "+Y+" )");
```

另见

[ChartWindowOnDropped](#), [ChartPriceOnDropped](#), [ChartTimeOnDropped](#)

ChartYOnDropped

返回EA交易或者脚本下跌到的图表点Y坐标。

```
int ChartYOnDropped();
```

返回值

Y 坐标值。

注释

自上向下的Y 轴方向。

另见

[ChartWindowOnDropped](#), [ChartPriceOnDropped](#), [ChartTimeOnDropped](#)

ChartSetSymbolPeriod

更改指定图表的交易品种和周期。函数不同步，例如，不等执行完成就发送命令。

```
bool ChartSetSymbolPeriod(  
    long          chart_id,      // 图表 ID  
    string        symbol,       // 交易品种名称  
    ENUM_TIMEFRAMES period      // 周期  
);
```

参量

chart_id

[in] 图表 ID. 0 意味着当前图表。

symbol

[in] 图表交易品种。 [NULL](#) 值意味着当前图表交易品种（附上EA交易）。

period

[in] 图表周期（时间表）。可以是[ENUM_TIMEFRAMES](#)值之一。0意味着当前图表周期。

返回值

如果成功返回true，否则false。

注释

交易品种/周期的更改导致图表中EA交易的重新初始化。

另见

[图表交易品种](#)， [图表周期](#)

ChartScreenShot

函数提供gif格式图表当前状态的屏幕截屏。

```
bool ChartScreenShot(
    long      chart_id,           // 图表 ID
    string     filename,          // 交易品种名称
    int        width,             // 宽度
    int        height,            // 高度
    ENUM_ALIGN_MODE align_mode=ALIGN_RIGHT // 对齐类型
);
```

参量

chart_id

[in] 图表 ID. 0 意味着当前图表。

filename

[in] 屏幕截图文件名。不能超过63个字符。屏幕截图文件位于directory \ Files。

width

[in] 屏幕截图宽度像素。

height

[in] 屏幕截图高度像素。

align_mode=ALIGN_RIGHT

[in] 屏幕截图的有限输出模式。 ALIGN_ RIGHT意味着对齐到右边距（从底部输出）。 ALIGN_ LEFT 意味着左对齐。

返回值

若成功返回true，否则false。

注释

如果要从图表中的某一点截图，首先用[ChartNavigate\(\)](#) 函数定位图表。如果截图水平大小小于图表窗口，根据align_ mode设置，会选择图表窗口的右边或者左边输出。

交易函数

该组函数用于管理交易活动。

交易函数在EA交易或者脚本中使用，交易函数只有在脚本或EA交易的"允许在线交易"多选框允许时调用。

函数	功能
OrderCalcMargin	用入金货币，计算指定订单类型所需的保证金
OrderCalcProfit	以入金货币计算基于传递参量的利润
OrderCheck	检测是否有足够资金执行所需 交易操作 。
OrderSend	发送 交易需求 到服务器
PositionsTotal	返回持仓数量
PositionGetSymbol	返回与持仓一致的交易品种
PositionSelect	选择一个持仓进行下一步工作
PositionGetDouble	返回持仓要求属性（双精度）
PositionGetInteger	返回持仓要求属性（日期时间或者整型）
PositionGetString	返回持仓要求属性（字符串）
OrdersTotal	返回订单数
OrderGetTicket	返回相应订单的订单号
OrderSelect	为下一步工作选择一订单
OrderGetDouble	返回订单要求属性（双精度型）
OrderGetInteger	返回订单要求属性（日期时间或者整型）
OrderGetString	返回订单要求属性（字符串）
HistorySelect	为服务器时间的指定周期检索交易和订单历史记录
HistorySelectByPosition	请求指定 仓位标识符 交易的历史记录
HistoryOrderSelect	为下一步工作选择历史订单
HistoryOrdersTotal	返回历史记录中订单数
HistoryOrderGetTicket	返回历史记录中相关订单的订单号
HistoryOrderGetDouble	返回（双精度）历史记录中订单需求属性
HistoryOrderGetInteger	返回（日期时间或者整型）历史记录中订单需求属性
HistoryOrderGetString	返回（字符串）历史记录中订单需求属性
HistoryDealSelect	历史记录中选择一个交易便于通过适当函数进一步调用它。
HistoryDealsTotal	返回历史记录交易数
HistoryDealGetTicket	返回历史相关交易订单号
HistoryDealGetDouble	返回历史记录中交易需求属性（双精度）

HistoryDealGetInteger	返回历史记录中交易需求属性（时间日期和整型）
HistoryDealGetString	返回历史记录中交易需求属性（字符串）

OrderCalcMargin

函数计算指定订单类型的需要保证金，在活期账户，当前的市场环境不能计算到当前的挂单和持仓。允许计算的保证金需要交易操作计划，值是返回的当前账户的值。

```
bool OrderCalcMargin(  
    ENUM_ORDER_TYPE    action,           // 订单类型  
    string              symbol,          // 交易品种名称  
    double              volume,          // 交易量  
    double              price,           // 开盘价  
    double&             margin           // 为获取保证金值的值  
);
```

参量

action

[in] 订单类型，可以是 [ENUM_ORDER_TYPE](#) 值中的一个。

symbol

[in] 交易品种名称。

volume

[in] 交易操作成交量。

price

[in] 开盘价。

margin

[out]要求保证金的值的变量成功执行到相应函数中去，计算会执行活期账户的开盘价而非交易订单，保证金值有很多元素，在不同的交易环境中相互区别。

返回值

如果成功，函数返回true，否则返回false，为了获得有关的 [错误](#) 信息，调用 [GetLastError\(\)](#) 函数。

另见

[OrderSend\(\)](#), [订单属性](#), [交易操作类型](#)

OrderCalcProfit

函数为当前账户计算利润，在当前的交易市场条件下，以参量为基础传递。函数使用交易操作的结构进行再评估，值是返回的当前账户的值。

```
bool OrderCalcProfit(
    ENUM_ORDER_TYPE    action,           // 订单类型 (ORDER_TYPE_BUY 或者 ORDER_TYP
    string              symbol,           // 交易品种名称
    double              volume,           // 交易量
    double              price_open,        // 开盘价
    double              price_close,       // 收盘价
    double&             profit            // 为获得利润值的变量
);
```

参量

action

[in] 订单类型，可以是 [ENUM_ORDER_TYPE](#) 项目里ORDER_TYPE_BUY 或 ORDER_TYPE_SELL值中的一个。

symbol

[in] 交易品种名称。

volume

[in] 交易操作成交量。

price_open

[in] 开盘价。

price_close

[in] 收盘价。

profit

[out] 计算利润的变量值在函数成功执行后编写进去，估计利润值依据许多因素，并在不同的市场环境中加以区别。

返回值

如果成功，函数返回true,否则返回false。如果指定无效订单类型，函数返回false，为了获得[错误](#)信息，可以调用 [GetLastError\(\)](#)。

另见

[OrderSend\(\)](#), [订单属性](#), [交易操作类型](#)

OrderCheck

OrderCheck() 函数检测是否有足够的钱执行需要的[交易操作](#)。检测结果在[MqlTradeCheckResult](#)结构域中。

```
bool OrderCheck(  
    MqlTradeRequest&    request,      // 请求结构  
    MqlTradeCheckResult& result       // 结果结构  
);
```

参量

request

[in] 执行 [MqlTradeRequest](#) 结构类型，该类型表述必要的操作活动。

result

[in,out] 指针指向 [MqlTradeCheckResult](#) 结构类型，显示检测结果。

返回值

如果操作时资金不足，或者错误填满了参量，函数返回false，如果成功检测了结构基础（检测指针），返回true-这并不表示要求的交易操作已经成功执行了。更多函数执行结果的细节描述，都分析在result结构域中。

为了获得有关[错误](#)的信息，调用 [GetLastError\(\)](#) 函数。

另见

[OrderSend\(\)](#), [交易操作类型](#), [交易请求结构](#), [检测结果请求结构](#), [交易请求结果结构](#)

OrderSend

通过向服务器发送请求，OrderSend() 函数用来执行交易操作。

```
bool OrderSend(
    MqlTradeRequest& request,      // query structure
    MqlTradeResult& result        // structure of the answer
);
```

参量

request

[in] 指针指向 [MqlTradeRequest](#) 结构类型操作，描述成客户的交易活动。

result

[in,out] 指针指向 [MqlTradeResult](#) 结构类型操作，描述成功完成后的交易操作结构（如果真值可以返回）。

返回值

成功检测基础结构（索引检测）返回真值-这并不是交易操作成功执行的标志。更多函数执行结果的细节描述，都分析在result结构域中。

注释

交易需要通过多个步骤检测交易服务器，第一，检测是否所有要求操作request参量都正确填满，如果没有错误，服务器接收下一步订单，如果命令通过服务器成功接收，OrderSend() 函数返回真值。

在向服务器发送请求之前推荐检验请求，为了检验请求，使用[OrderCheck\(\)](#) 函数。检验是否有足够的资金来执行交易操作，并在[交易请求检测结果](#)中返回许多有用的参量：

- [返回码](#) 在检测请求中包括错误信息；
- 在交易操作执行后显示账户余额；
- 在交易操作执行后显示产权值；
- 在交易操作执行后显示浮点类型值；
- 交易操作所需保证金；
- 在交易操作执行后可用产权数量；
- 在交易操作执行后建立的保证金水平；
- 注释回应代码，错误描述。

当放置市场订单时需要注释，使用OrderSend() 方式操作的成果示例并不总是表示成功操作订单。在返回的 [result structure](#) result中的retcode 值包括[trade server return code](#)，并且该值依据[type of operation](#) 的域 deal 或者 order。

每个接收到的订单存储在交易服务器中排队处理，直到可以执行的条件发生：

- 临界期，
- 反面要求的外观，
- 当执行价格出现时命令执行，
- 收到的取消订单的命令。

订单执行时，交易服务器向客户端发送关于[交易事件](#)程序发生的信息，可以通过[OnTrade\(\)](#) 函数执行。

示例：

```
// --- ORDER_MAGI C值
```



```

input long order_magic=55555;
//+-----+
//| 启动函数的脚本程序 |
//+-----+
void OnStart()
{
//--- 确保账户是样本
if(AccountInfoInteger(ACCOUNT_TRADE_MODE)==ACCOUNT_TRADE_MODE_REAL)
{
Alert("Script operation is not allowed on a live account!");
return;
}
//--- 下订单或者删除订单
if(GetOrdersTotalByMagic(order_magic)==0)
{
//--- 无当前订单 - 下订单
int res=SendRandomPendingOrder(order_magic);
Print("Return code of the trade server ",res);
}
else // 有订单 - 删除订单
{
DeleteAllOrdersByMagic(order_magic);
}
}
//+-----+
//| 接收当前带有指定的ORDER_MAGIC的订单号 |
//+-----+
int GetOrdersTotalByMagic(long const magic_number)
{
long order_ticket;
int total=0;
//--- 在全部历史记录中选择
HistorySelect(0,TimeCurrent());
//--- 检查通过所有挂单
for(int i=0;i<OrdersTotal();i++)
if(order_ticket=OrderGetTicket(i))
if(magic_number==OrderGetInteger(ORDER_MAGIC)) total++;
//---
return(total);
}
//+-----+
//| 删除所有带有指定ORDER_MAGIC的挂单 |
//+-----+
void DeleteAllOrdersByMagic(long const magic_number)
{
long order_ticket;
int total=0;
//--- 在全部历史记录中选择

```

```

HistorySelect(0,TimeCurrent());
//--- 检查所有挂单
for(int i=OrdersTotal()-1;i>=0;i--)
    if(order_ticket=OrderGetTicket(i))
        //--- 带有恰当ORDER_MAGIC的订单
        if(magic_number==OrderGetInteger(ORDER_MAGIC))
        {
            MqlTradeResult result;
            MqlTradeRequest request;
            request.order=order_ticket;
            request.action=TRADE_ACTION_REMOVE;
            OrderSend(request,result);
            //--- 编写服务器回复到日志
            Print(__FUNCTION__," ",result.comment," reply code ",result.retcode);
        }
//---
}
//+-----+
//| 随机设置挂单 |
//+-----+
int SendRandomPendingOrder(long const magic_number)
{
//--- 准备请求
MqlTradeRequest request;
request.action=TRADE_ACTION_PENDING; // 设置挂单
request.magic=magic_number; // ORDER_MAGIC
request.symbol=_Symbol; // 交易品种
request.volume=0.1; // 0.1为单位的交易量
request.sl=0; // 没有指定止损价位
request.tp=0; // 没有指定盈利价位
//--- 形成订单类型
request.type=GetRandomType(); // 订单类型
//--- 形成挂单价格
request.price=GetRandomPrice(request.type); // 开盘价
//--- 发送交易请求
MqlTradeResult result;
OrderSend(request,result);
//--- 编写服务器回复到日志
Print(__FUNCTION__," ",result.comment);
if(result.retcode==10016) Print(result.bid,result.ask,result.price);
//--- 返回交易服务器回复的代码
return result.retcode;
}
//+-----+
//| 返回随机挂单类型 |
//+-----+
ENUM_ORDER_TYPE GetRandomType()
{

```

```

    int t=MathRand()%4;
//--- 0<=t<4
    switch(t)
    {
        case(0):return(ORDER_TYPE_BUY_LIMIT);
        case(1):return(ORDER_TYPE_SELL_LIMIT);
        case(2):return(ORDER_TYPE_BUY_STOP);
        case(3):return(ORDER_TYPE_SELL_STOP);
    }
//--- 不正确值
    return(WRONG_VALUE);
//---
}
//+-----+
//|  返回随机价格  |
//+-----+
double GetRandomPrice(ENUM_ORDER_TYPE type)
{
    int t=(int)type;
//--- 交易品种止损水平
    int distance=(int)SymbolInfoInteger(_Symbol,SYMBOL_TRADE_STOPS_LEVEL);
//--- 接收上一个订单号数据
    MqlTick last_tick;
    SymbolInfoTick(_Symbol,last_tick);
//--- 按照类型计算价格
    double price;
    if(t==2 || t==5) // ORDER_TYPE_BUY_LIMIT or ORDER_TYPE_SELL_STOP
    {
        price=last_tick.bid; // 不同于卖价
        price=price-(distance+(MathRand()%10)*5)*_Point;
    }
    else // ORDER_TYPE_SELL_LIMIT or ORDER_TYPE_BUY_STOP
    {
        price=last_tick.ask; // 不同于买价
        price=price+(distance+(MathRand()%10)*5)*_Point;
    }
    return(price);
}

```

另见

[交易操作类型](#), [交易请求结构](#), [请求检测结果结构](#), [交易请求结果结构](#)

PositionsTotal

返回持仓的数量。

```
int PositionsTotal();
```

返回值

整型 值

Note

对于每一个[交易品种](#)而言，任意给出时间值都只有一个[仓位](#)可以开放，它能促成一笔或者更多[交易](#)，不要弄混[挂单](#)位置，也会在客户端“工具箱”的“交易”标签中显示。

[交易账户](#) 总仓位数不能超过全部[金融工具数](#)。

另见

[PositionGetSymbol\(\)](#)，[PositionSelect\(\)](#)，[仓位属性](#)

PositionGetSymbol

返回的交易品种与开仓位置的值保持一致，使用 [PositionGetDouble](#)、[PositionGetInteger](#)、[PositionGetString](#) 函数为下一项操作自动选择位置。

```
string PositionGetSymbol(  
    int index    // 仓位列表中的数量  
);
```

参量

index

[in] 开仓位置的列表位置数量。

返回值

[字符串](#) 型值，如果找不到位置，返回空字符串，为了获得 [错误代码](#)，调用 [GetLastError\(\)](#) 函数。

注释

对于每一个[交易品种](#)，任何给出的时间段，只有一个[仓位](#)可以开放，它能促成一笔或多笔[交易](#)的产生，不要弄混[挂单](#)位置，也会在客户端“工具箱”的“交易”中显示。

[交易账户](#) 总仓位数不能超过[金融工具数](#)的总量。

另见

[PositionsTotal\(\)](#)、[PositionSelect\(\)](#)、[仓位属性](#)

PositionSelect

为接下来的工作选择一个仓位，如果函数成功完成，返回true，如果失败返回false，调用 [GetLastError\(\)](#) 函数获取错误信息。

```
bool PositionSelect(  
    string symbol    // 交易品种名称  
);
```

参量

symbol
[in] 金融证券名称

返回值

布尔型值

注释

对于每个[交易品种](#)，任何给出时间，仅可以持有一个[仓位](#)，它是一个或者多个[交易](#)的结果，不要弄混[挂单](#)仓位，也会在客户端“工具箱”的“交易”中显示。

函数PositionSelect() 复制程序函数的位置数据，进一步调用[PositionGetDouble\(\)](#)，[PositionGetInteger\(\)](#) 和 [PositionGetString\(\)](#) 返回较早的复制数据。表明该位置本身已经不再存在（或者成交量，方向等的改变），但位置数据仍旧可以获得，为了确保关于位置的新数据的接受，在提及他们之前，推荐使用 PositionSelect() 函数。

另见

[PositionGetSymbol\(\)](#)，[PositionsTotal\(\)](#)，[仓位属性](#)

PositionGetDouble

函数返回持仓的要求性质，使用[PositionGetSymbol](#)或者 [PositionSelect](#)预选出来，仓位属性必须是双精度型，有两种变体函数。

1. 立即返回属性值。

```
double PositionGetDouble(
    ENUM_POSITION_PROPERTY_DOUBLE property_id // 属性标识符
);
```

2. 返回true或者false，取决于函数执行成功与否，如果成功，性质值通过引用安置在最后参量的接受变量里。

```
bool PositionGetDouble(
    ENUM_POSITION_PROPERTY_DOUBLE property_id, // 属性标识符
    double& double_var // 这里接受属性值
);
```

参量

property_id

[in] 位置属性标识符。值可能是[ENUM_POSITION_PROPERTY_DOUBLE](#) 计算中一个。

double_var

[out] 双精度类型值，如果函数调用失败，返回0。

返回值

[双精度](#) 类型值。如果函数失败，返回0。

注释

对于每个[交易品种](#)，任何给出时间，仅可以持有一个[仓位](#)，它是一个或者多个[交易](#)的结果，不要弄混[挂单](#)仓位，也会在客户端“工具箱”的“交易”中显示。

为确保关于仓位的新数据的接收，在提及之前调用 [PositionSelect\(\)](#) 函数。

另见

[PositionGetSymbol\(\)](#), [PositionSelect\(\)](#), [仓位属性](#)

PositionGetInteger

函数返回持仓需求属性，使用 [PositionGetSymbol](#) 或者 [PositionSelect](#). 预选出来，仓位属性的类型是时间日期整型，有两种变体函数。

1. 立即返回属性值

```
long PositionGetInteger (
    ENUM_POSITION_PROPERTY_INTEGER property_id    // 属性标识符
);
```

2. 返回true或者false，取决于函数执行成功与否，如果成功，性质值通过引用安置在最后参量的接受变量里。

```
bool PositionGetInteger (
    ENUM_POSITION_PROPERTY_INTEGER property_id,    // 属性标识符
    long& long_var                                // 这里接受属性值
);
```

参量

property_id

[in] 仓位属性标识符，值可能是 [ENUM_POSITION_PROPERTY_INTEGER](#) 中的一个。

long_var

[out] 接受要求性质的字符串变量值。

返回值

[长整型](#) 值。如果函数失败，返回0。

注释

对于每个[交易品种](#)，任何给出时间，仅可以持有一个[仓位](#)，它是一个或者多个[交易](#)的结果，不要弄混[挂单](#)仓位，也会在客户端“工具箱”的“交易”中显示。

为确保关于仓位的新数据的接收，在提及之前调用 [PositionSelect\(\)](#) 函数。

另见

[PositionGetSymbol\(\)](#), [PositionSelect\(\)](#), [仓位属性](#)

PositionGetString

函数返回持仓需求属性，使用[PositionGetSymbol](#)或者[PositionSelect](#)预选出来。仓位属性可以是字符串型。有2种变体函数。

1. 立即返回属性值

```
string PositionGetString(
    ENUM_POSITION_PROPERTY_STRING property_id    // 属性标识符
);
```

2. 返回true或者false，取决于函数执行成功与否，如果成功，性质值通过引用安置在最后参量的接受变量里。

```
bool PositionGetString(
    ENUM_POSITION_PROPERTY_STRING property_id,    // 属性标识符
    string& string_var                            // 这里接受属性值
);
```

参量

property_id

[in] 仓位属性标识符，值可能是 [ENUM_POSITION_PROPERTY_STRING](#) 中的一个。

string_var

[out] 接受要求性质的字符串变量值。

返回值

[字符串](#) 值。如果函数失败，返回空字符串。

注释

对于每个[交易品种](#)，任何给出时间，仅可以持有一个[仓位](#)，它是一个或者多个[交易](#)的结果，不要弄混[挂单](#)仓位，也会在客户端“工具箱”的“交易”中显示。

为确保关于仓位的新数据的接收，在提及之前调用 [PositionSelect\(\)](#) 函数。

另见

[PositionGetSymbol\(\)](#), [PositionSelect\(\)](#), [仓位属性](#)

OrdersTotal

返回当前订单数。

```
int OrdersTotal();
```

返回值

[整型](#) 值。

注释

不要使仓位[挂单](#)混乱，它显示在客户端“工具箱”的“交易”标签中。订单是[交易](#)要求，而仓位就是一个或者多个[交易](#)的结果。

对于每个[交易品种](#)而言，在任何给定的时间里，仅持有一个[仓位](#)，而同一个交易品种却可以有多个挂单。

另见

[OrderSelect\(\)](#)，[OrderGetTicket\(\)](#)，[订单属性](#)

OrderGetTicket

返回类似订单票据，使用函数自动选择订单工作。

```
ulong OrderGetTicket(
    int index    // 订单列表中的数量
);
```

参量

index

[in] 当前订单列表中的订单数量。

返回值

无符长整型 值。如果函数失败，返回0。

注释

不要使仓位挂单混乱，它显示在客户端“工具箱”的“交易”标签中。订单是[交易](#)要求，而仓位就是一个或者多个[交易](#)的结果。

对于每个[交易品种](#)，任何给出时间，只有一个[仓位](#)位置是可以持有的，而相同的交易品种可以有很多挂单。

OrderGetTicket() 函数复制有关订单的数据到程序环境中，远程调用[OrderGetDouble\(\)](#)、[OrderGetInteger\(\)](#)、[OrderGetString\(\)](#)，返回先前的复制数据。表示该命令本身不存在（或者开仓价格，止损数值/目标数值水平或者届期改变），但是命令数据仍旧包含。为了确保接收订单新命令，推荐调用 OrderGetTicket() 命令。

示例：

```
void OnStart()
{
//--- 订单属性返回值的变量
    ulong    ticket;
    double    open_price;
    double    initial_volume;
    datetime  time_setup;
    string     symbol;
    string     type;
    long      order_magic;
//--- 当前挂单量
    uint      total=OrdersTotal();
//--- 反复检查通过订单
    for(uint i=0;i<total;i++)
    {
        //--- 通过列表中的仓位返回订单报价
        if(ticket=OrderGetTicket(i))
        {
            //--- 返回订单属性
            open_price=    =OrderGetDouble(ORDER_PRICE_OPEN);
            time_setup    =OrderGetInteger(ORDER_TIME_SETUP);
```

```
symbol      =OrderGetString(ORDER_SYMBOL);
order_magic =OrderGetInteger(ORDER_MAGIC);
positionID  =OrderGetInteger(ORDER_POSITION_ID);
initial_volume=OrderGetDouble(ORDER_VOLUME_INITIAL);
type        =EnumToString(ORDER_TYPE);
//--- 准备和显示订单信息
printf("#ticket %d %s %G %s at %G was set up at %s",
        ticket,           // 订单报价
        type,             // 类型
        initial_volume,   // 已下交易量
        symbol,           // 交易品种
        open_price,       // 规定的开盘价
        TimeToString(time_setup)// 下订单时间
    );
    }
}
//---
}
```

另见

[OrdersTotal\(\)](#), [OrderSelect\(\)](#), [OrderGetInteger\(\)](#)

OrderSelect

选择工作订单。如果函数成功调用返回真值，如果函数没能完成，返回错误值，了解更多关于错误调用的信息，使用 [GetLastError\(\)](#)。

```
bool OrderSelect(  
    ulong ticket    // 订单号  
);
```

参量

ticket
[in] 订单号。

返回值

布尔型值。

注释

不要使仓位挂单混乱，它显示在客户端“工具箱”的“交易”标签中。对于每个[交易品种](#)，任何给出时间，只有一个[仓位](#)位置是可以持有的，而相同的交易品种可以有很多挂单。

OrderSelect() 函数复制有关订单的数据到程序环境中，远程调用[OrderGetDouble\(\)](#) -[OrderGetInteger\(\)](#) -[OrderGetString\(\)](#)，返回先前的复制数据。表示该命令本身不存在（或者开仓价格，止损数值/目标数值水平或者届期改变），但是命令数据仍旧包含。为了确保接收订单新命令，推荐调用OrderSelect() 命令。

另见

[OrderGetInteger\(\)](#)， [OrderGetDouble\(\)](#)， [OrderGetString\(\)](#)， [OrderCalcProfit\(\)](#) -
[OrderGetTicket\(\)](#)， [订单属性](#)

OrderGetDouble

返回订单的需求属性，使用 [OrderGetTicket](#) 或者 [OrderSelect](#) 进行预选择。订单属性必须是双精度类型。有两种变量函数可以使用。

1. 立即返回属性值。

```
double OrderGetDouble (
    ENUM_ORDER_PROPERTY_DOUBLE property_id // 属性标识符
);
```

2. 依据函数是否成功调用返回真值或者错误值。如果成功，目标变量的属性值通过引用传递到最后参量。

```
bool OrderGetDouble (
    ENUM_ORDER_PROPERTY_DOUBLE property_id, // 属性标识符
    double& double_var // 这里接受属性值
);
```

参量

property_id

[in] 订单性质标识符，值可以是 [ENUM_ORDER_PROPERTY_DOUBLE](#) 值中的一个。

double_var

[out] 双精度类型变量接收要求属性的值。

返回值

[双精度](#) 值。如果函数失败，返回0。

注释

不要使仓位挂单混乱，它显示在客户端“工具箱”的“交易”标签中。对于每个[交易品种](#)，任何给出时间，只有一个[仓位](#)位置是可以持有的，而相同的交易品种可以有很多挂单。

为确保关于新订单的新数据的接受，推荐调用 [OrderSelect\(\)](#) 函数。

另见

[OrdersTotal\(\)](#), [OrderGetTicket\(\)](#), [订单属性](#)

OrderGetInteger

返回订单性质的要求，使用[OrderGetTicket](#) 或者 [OrderSelect](#)重新选择，属性一定是日期时间，整型。有2种变量函数可以使用。

1. 立即返回属性值。

```
long OrderGetInteger(
    ENUM_ORDER_PROPERTY_INTEGER property_id // 属性标识符
);
```

2. 依据函数成功与否返回真值或者错误值。如果成功，目标变量的属性值通过引用传递到最后参量。

```
bool OrderGetInteger(
    ENUM_ORDER_PROPERTY_INTEGER property_id, // 属性标识符
    long& long_var // 这里接受属性值
);
```

参量

property_id

[in] 订单性质标识符。值可以是 [ENUM_ORDER_PROPERTY_INTEGER](#) 列举值中的一个。

long_var

[out] 接收要求属性值的长整型变量。

返回值

[长整型](#) 值。如果函数失败，返回0。

注释

不要使仓位[挂单](#)混乱，它显示在客户端“工具箱”的“交易”标签中。对于每个[交易品种](#)，任何给出时间，只有一个[仓位](#)位置是可以持有的，而相同的交易品种可以有很多挂单。

为确保关于新订单的新数据的接受，推荐调用 [OrderSelect\(\)](#) 函数。

另见

[OrdersTotal\(\)](#), [OrderGetTicket\(\)](#), [订单属性](#)

OrderGetString

返回要求的订单属性，使用 [OrderGetTicket](#) 或者 [OrderSelect](#) 预选择。订单属性一定是字符串型。有2种变量函数可以使用。

1. 立即返回属性值。

```
string OrderGetString(
    ENUM_ORDER_PROPERTY_STRING property_id // 属性标识符
);
```

2. 依据函数成功与否返回真值或者错误值。如果成功，目标变量的属性值通过引用传递到最后参量。

```
bool OrderGetString(
    ENUM_ORDER_PROPERTY_STRING property_id, // 属性标识符
    string& string_var // 这里接受属性值
);
```

参量

property_id

[in] 订单性质标识符，值可以是 [ENUM_ORDER_PROPERTY_STRING](#) 列举值中的一个。

string_var

[out] 接收要求属性值的字符串型变量。

返回值

[字符串](#) 类型。

注释

不要使仓位[挂单](#)混乱，它显示在客户端“工具箱”的“交易”标签中。对于每个[交易品种](#)，任何给出时间，只有一个[仓位](#)位置是可以持有的，而相同的交易品种可以有很多挂单。

为确保关于新订单的新数据的接受，推荐调用 [OrderSelect\(\)](#) 函数。

另见

[OrdersTotal\(\)](#), [OrderGetTicket\(\)](#), [订单属性](#)

HistorySelect

在服务器特定时间段，检索交易和订单的历史。

```
bool HistorySelect(
    datetime from_date,    // 开始日期
    datetime to_date       // 结束日期
);
```

参量

from_date

[in] 要求开始日期。

to_date

[in] 要求的结束日期。

返回值

如果成功返回true，否则返回false。

注释

HistorySelect() 建立一系列MQL5程序的订单和交易列表，使用类似函数来执行列表元素。订单列表大小可以使用 [HistoryDealsTotal\(\)](#) 函数返回。使用 [HistoryOrdersTotal\(\)](#) 可以显示历史订单列表的大小。在订单列表中选择应该使用 [HistoryOrderGetTicket\(\)](#)，订单列表 [HistoryDealGetTicket\(\)](#) 更适合。

在使用 [HistoryOrderSelect\(\)](#) 后，MQL5程序中的历史可用订单列表就会重设并被发现的订单填满。如果成功执行 [the search of an order by the ticket](#)，同样适用于MQL5程序可行性订单中-通过 [HistoryDealSelect\(\)](#) 重设并再次成功填满。

示例：

```
void OnStart()
{
    color BuyColor =clrBlue;
    color SellColor=clrRed;
    //--- 请求交易历史记录
    HistorySelect(0,TimeCurrent());
    //--- 创建物件
    string name;
    uint total=HistoryDealsTotal();
    ulong ticket=0;
    double price;
    double profit;
    datetime time;
    string symbol;
    long type;
    long entry;
    //--- 所有交易
    for(uint i=0;i<total;i++)
```

```

{
//--- 尽力获得交易报价
if(ticket=HistoryDealGetTicket(i))
{
//--- 获得交易属性
price =HistoryDealGetDouble(ticket,DEAL_PRICE);
time  =HistoryDealGetInteger(ticket,DEAL_TIME);
symbol=HistoryDealGetString(ticket,DEAL_SYMBOL);
type  =HistoryDealGetInteger(ticket,DEAL_TYPE);
entry =HistoryDealGetInteger(ticket,DEAL_ENTRY);
profit=HistoryDealGetDouble(ticket,DEAL_PROFIT);
//--- 只对当前交易品种
if(price && time && symbol==Symbol())
{
//--- 创建价格物件
name="TradeHistory_Deal_"+string(ticket);
if(entry) ObjectCreate(0,name,OBJ_ARROW_RIGHT_PRICE,0,time,price,0,0);
else      ObjectCreate(0,name,OBJ_ARROW_LEFT_PRICE,0,time,price,0,0);
//--- 设置物件属性
ObjectSetInteger(0,name,OBJPROP_SELECTABLE,0);
ObjectSetInteger(0,name,OBJPROP_BACK,0);
ObjectSetInteger(0,name,OBJPROP_COLOR,type?BuyColor:SellColor);
if(profit!=0) ObjectSetString(0,name,OBJPROP_TEXT,"Profit: "+string(profit));
}
}
}
//--- 应用于图表
ChartRedraw();
}

```

另见

[HistoryOrderSelect\(\)](#), [HistoryDealSelect\(\)](#)

HistorySelectByPosition

检索特殊位置标识符交易和订单历史记录。

```
bool HistorySelectByPosition(  
    long position_id // 仓位标识符 - POSITION\_IDENTIFIER  
);
```

参量

position_id

[in] 位置标识符设置每个已执行命令和每笔交易。

返回值

如果成功返回真值，否则返回错误值。

注释

不能把出现在客户端"工具箱"的"交易" 标签中的当前[挂单](#)和交易历史订单弄乱。取消的或者导致的交易的[订单](#)列表可以在"工具箱"客户端中的"历史记录"标签浏览。

HistorySelectByPosition() 建立一系列命令列表和MQL5程序的交易，使用类似 [仓位标识符](#) 函数来执行列表元素。订单列表大小可以使用 [HistoryDealsTotal\(\)](#) 函数返回。使用[HistoryOrdersTotal\(\)](#)。可以显示历史订单列表的大小。使用[HistoryOrderGetTicket\(\)](#)，运行订单列表元素，订单列表元素—[HistoryDealGetTicket\(\)](#)。

在使用 [HistoryOrderSelect\(\)](#)，后，MQL5程序中的历史可用订单列表就会重设并被发现的订单填满。如果成功执行 [通过报价搜索订单](#)同样适用于MQL5程序可行性订单中-通过 [HistoryDealSelect\(\)](#) 重设并再次成功填满。

另见

[HistorySelect\(\)](#)， [HistoryOrderGetTicket\(\)](#)， [订单属性](#)

HistoryOrderSelect

从历史中选择订单，彻底调用适当函数。如果函数成功完成，返回真值，如果函数返回失败显示错误值。更多错误订单调用 [GetLastError\(\)](#)。

```
bool HistoryOrderSelect(  
    ulong ticket    // 订单号  
);
```

参量

ticket
[in] 订单号。

返回值

如果成功，返回真值，否则返回错误值。

注释

不能把出现在客户端"工具箱"的"交易" 标签中的当前[挂单](#)和交易历史订单弄乱。取消的或者导致的交易的[订单](#)列表可以在"工具箱"客户端中的"历史记录"标签浏览。

HistoryOrderSelect() 在MQL5程序从历史记录中删除一系列订单，如果成功执行HistoryOrderSelect()，可用调用和复制的单个字节，如果需要通过 [HistorySelect\(\)](#) 运行所有订单，最好使用 [HistoryOrderGetTicket\(\)](#)。

另见

[HistorySelect\(\)](#)， [HistoryOrderGetTicket\(\)](#)， [订单属性](#)

HistoryOrdersTotal

在历史中返回订单数量。优先调用HistoryOrdersTotal()，首先需要接收使用[HistorySelect\(\)](#)或者[HistorySelectByPosition\(\)](#)函数的交易和订单历史记录。

```
int HistoryOrdersTotal();
```

返回值

[整型](#) 值。

注释

不能把出现在客户端"工具箱"的"交易" 标签中的当前[挂单](#)和交易历史订单弄乱。取消的或者导致的交易的[订单](#)列表可以在"工具箱"客户端中的"历史记录"标签浏览。

另见

[HistorySelect\(\)](#), [HistoryOrderSelect\(\)](#), [HistoryOrderGetTicket\(\)](#), [订单属性](#)

HistoryOrderGetTicket

在历史中返回相关订单报价。优先调用HistoryOrderGetTicket()，首先需要接收使用[HistorySelect\(\)](#)或者[HistorySelectByPosition\(\)](#)函数的交易和订单历史记录。

```
ulong HistoryOrderGetTicket(
    int index // 订单列表中的数量
);
```

参量

index

[in] 订单列表中的订单数。

返回值

[无符长整型](#) 值。

注释

不能把出现在客户端"工具箱"的"交易" 标签中的当前[挂单](#)和交易历史订单弄乱。取消的或者导致的交易的[订单列表](#)可以在"工具箱"客户端中的"历史记录"标签浏览。

示例：

```
void OnStart()
{
    datetime from=0;
    datetime to=TimeCurrent();
    //--- 要求全部历史记录
    HistorySelect(from,to);
    //--- 返回订单属性值变量
    ulong ticket;
    double open_price;
    double initial_volume;
    datetime time_setup;
    datetime time_done;
    string symbol;
    string type;
    long order_magic;
    long positionID;
    //--- 当前挂单数量
    uint total=HistoryOrdersTotal();
    //--- 循环检测通过订单
    for(uint i=0;i<total;i++)
    {
        //--- 通过其列表中的位置返回订单报价
        if(ticket=HistoryOrderGetTicket(i))
        {
            //--- 返回订单属性
            open_price= HistoryOrderGetDouble(ticket,ORDER_PRICE_OPEN);
            time_setup= HistoryOrderGetInteger(ticket,ORDER_TIME_SETUP);
```

```

time_done=      HistoryOrderGetInteger(ticket,ORDER_TIME_DONE);
symbol=         HistoryOrderGetString(ticket,ORDER_SYMBOL);
order_magic=    HistoryOrderGetInteger(ticket,ORDER_MAGIC);
positionID =    HistoryOrderGetInteger(ticket,ORDER_POSITION_ID);
initial_volume= HistoryOrderGetDouble(ticket,ORDER_VOLUME_INITIAL);
type=GetOrderType(HistoryOrderGetInteger(ticket,ORDER_TYPE));
//--- 准备并显示订单信息
printf("#ticket %d %s %G %s at %G was set up at %s => done at %s, pos ID=%d"
       ticket,          // 订单报价
       type,            // 类型
       initial_volume,  // 已下的交易量
       symbol,          // 交易品种
       open_price,      // 规定的开盘价
       TimeToString(time_setup), // 下订单时间
       TimeToString(time_done), // 订单执行或者删除时间
       positionID       // 一个包括订单交易的仓位ID
       );
    }
}
//---
}
//+-----+
//|  返回订单类型字符串名称          |
//+-----+
string GetOrderType(long type)
{
    string str_type="unknown operation";
    switch(type)
    {
        case (ORDER_TYPE_BUY):      return("buy");
        case (ORDER_TYPE_SELL):     return("sell");
        case (ORDER_TYPE_BUY_LIMIT): return("buy limit");
        case (ORDER_TYPE_SELL_LIMIT): return("sell limit");
        case (ORDER_TYPE_BUY_STOP):  return("buy stop");
        case (ORDER_TYPE_SELL_STOP): return("sell stop");
        case (ORDER_TYPE_BUY_STOP_LIMIT): return("buy stop limit");
        case (ORDER_TYPE_SELL_STOP_LIMIT): return("sell stop limit");
    }
    return(str_type);
}

```

另见

[HistorySelect\(\)](#), [HistoryOrdersTotal\(\)](#), [HistoryOrderSelect\(\)](#), [订单属性](#)

HistoryOrderGetDouble

返回要求的订单属性，订单属性一定是双精度型。有2个变量属性可供使用。

1. 立即返回属性值。

```
double HistoryOrderGetDouble(
    ulong          ticket_number,    // 订单号
    ENUM_ORDER_PROPERTY_DOUBLE property_id // 属性标识符
);
```

2. 依据函数成功与否返回真值或者错误值。如果成功，目标变量的属性值通过引用传递到最后参量。

```
bool HistoryOrderGetDouble(
    ulong          ticket_number,    // 订单号
    ENUM_ORDER_PROPERTY_DOUBLE property_id, // 属性标识符
    double&        double_var       // 这里接受属性值
);
```

参量

ticket_number

[in] 订单号。

property_id

[in] 订单属性标识符。值可以是 [ENUM_ORDER_PROPERTY_DOUBLE](#) 枚举值中的一个。

double_var

[out] 接收需要的属性值双精度类型变量。

返回值

[双精度](#) 类型值。

注释

不能把出现在客户端"工具箱"的"交易" 标签中的当前[挂单](#)和交易历史订单弄乱。取消的或者导致的交易的[订单](#)列表可以在"工具箱"客户端中的"历史记录" 标签浏览。

另见

[HistorySelect\(\)](#), [HistoryOrdersTotal\(\)](#), [HistoryOrderSelect\(\)](#), [订单属性](#)

HistoryOrderGetInteger

返回订单的要求属性，订单属性一定是日期时间，整型，有2种变量函数可供使用。

1. 立即返回属性值。

```
long HistoryOrderGetInteger(
    ulong          ticket_number,    // 订单号
    ENUM_ORDER_PROPERTY_INTEGER property_id // 属性标识符
);
```

2. 依据函数成功与否返回真值或者错误值。如果成功，目标变量的属性值通过引用传递到最后参量。

```
bool HistoryOrderGetInteger(
    ulong          ticket_number,    // 订单号
    ENUM_ORDER_PROPERTY_INTEGER property_id, // 属性标识符
    long&          long_var         // 这里接受属性值
);
```

参量

ticket_number

[in] 订单号。

property_id

[in] 订单属性标识符，值可以是 [ENUM_ORDER_PROPERTY_INTEGER](#) 列举值中的一个。

long_var

[out] 接收要求属性值的长型变量。

返回值

[长整型](#) 值。

注释

不能把出现在客户端"工具箱"的"交易" 标签中的当前[挂单](#)和交易历史订单弄乱。取消的或者导致的交易的[订单](#)列表可以在"工具箱"客户端中的"历史记录"标签浏览。

另见

[HistorySelect\(\)](#), [HistoryOrdersTotal\(\)](#), [HistoryOrderSelect\(\)](#) -[订单属性](#)

HistoryOrderGetString

返回订单要求属性，订单属性一定是字符串类型，有2个变量函数可供使用。

1. 立即返回属性值。

```
string HistoryOrderGetString(  
    ulong                ticket_number,    // 订单号  
    ENUM_ORDER_PROPERTY_STRING property_id    // 属性标识符  
);
```

2. 依据函数成功与否返回真值或者错误值。如果成功，目标变量的属性值通过引用传递到最后参量。

```
bool HistoryOrderGetString(  
    ulong                ticket_number,    // 订单号  
    ENUM_ORDER_PROPERTY_STRING property_id,    // 属性标识符  
    string&              string_var        // 这里接受属性值  
);
```

参量

ticket_number

[in] 订单号。

property_id

[in] 订单属性标识符。值可以是 [ENUM_ORDER_PROPERTY_STRING](#) 枚举值中的一个。

string_var

[out] 字符串型变量。

返回值

[字符串](#) 型值。

注释

不能把出现在客户端"工具箱"的"交易" 标签中的当前[挂单](#)和交易历史订单弄乱。取消的或者导致的交易的[订单](#)列表可以在"工具箱"客户端中的"历史记录"标签浏览。

另见

[HistorySelect\(\)](#), [HistoryOrdersTotal\(\)](#), [HistoryOrderSelect\(\)](#), [订单属性](#)

HistoryDealSelect

通过调用恰当函数选择历史交易。如果函数成功完成，返回真值，如果函数失败返回错误值。有关错误更多细节，调用 [GetLastError\(\)](#)。

```
bool HistoryDealSelect(  
    ulong ticket    // 交易订单号  
);
```

参量

ticket

[in] 交易订单号。

返回值

如果成功返回真值，否则返回错误值。

注释

不要弄混 [订单](#)，[交易](#) 和 [仓位](#)。每笔交易都按照订单执行，每个仓位都是一个或多个交易的值。

HistoryDealSelect() 在MQL5程序从历史记录中删除一系列订单，如果成功执行HistoryDealSelect()，可用调用和复制的单个交易，如果需要通过 [HistorySelect\(\)](#) 运行所有订单，最好使用 [HistoryDealGetTicket\(\)](#)。

另见

[HistorySelect\(\)](#)，[HistoryDealGetTicket\(\)](#)，[交易属性](#)

HistoryDealsTotal

返回历史交易数量，先前调用HistoryDealsTotal()，首先有必要接收交易历史和使用[HistorySelect\(\)](#)或者[HistorySelectByPosition\(\)](#)函数的订单。

```
int HistoryDealsTotal();
```

返回值

[整型](#) 值。

注释

不要弄混 [订单](#)，[交易](#) 和 [仓位](#)。每笔交易都按照订单执行，每个仓位都是一个或多个交易的值。

另见

[HistorySelect\(\)](#)，[HistoryDealGetTicket\(\)](#)，[交易属性](#)

HistoryDealGetTicket

函数选择进一步交易过程并返回历史交易票据。优先调用HistoryDealGetTicket()，首先有必要接收使用[HistorySelect\(\)](#) 或者 [HistorySelectByPosition\(\)](#) 函数的交易和订单历史记录。

```
ulong HistoryDealGetTicket (
    int index      // 订单号交易
);
```

参量

index

[in] 交易列表中的交易数量。

返回值

[无符长整型](#) 值。

注释

不要弄混 [订单](#)，[交易](#) 和 [仓位](#)。每笔交易都按照订单执行，每个仓位都是一个或多个交易的值。

示例：

```

void OnStart()
{
    ulong deal_ticket;           // 交易订单号
    ulong order_ticket;          // 执行交易的订单的订单号
    datetime transaction_time;    // 交易执行时间
    long deal_type;              // 交易操作类型
    long position_ID;            // 仓位ID
    string deal_description;      // 操作描述
    double volume;               // 操作交易量
    string symbol;               // 交易的交易品种
    //--- 设置交易历史记录请求的起止时间
    datetime from_date=0;        // 从最开始
    datetime to_date=TimeCurrent(); // 到当前时间
    //--- 特定周期中请求交易历史记录
    HistorySelect(from_date,to_date);
    //--- 交易列表中的全部数量total number in the list of deals
    int deals=HistoryDealsTotal();
    //--- 现在处理每一个交易
    for(int i=0;i<deals;i++)
    {
        deal_ticket=           HistoryDealGetTicket(i);
        volume=                HistoryDealGetDouble(deal_ticket,DEAL_VOLUME);
        transaction_time=(datetime)HistoryDealGetInteger(deal_ticket,DEAL_TIME);
        order_ticket=          HistoryDealGetInteger(deal_ticket,DEAL_ORDER);
        deal_type=              HistoryDealGetInteger(deal_ticket,DEAL_TYPE);
        symbol=                 HistoryDealGetString(deal_ticket,DEAL_SYMBOL);
        position_ID=            HistoryDealGetInteger(deal_ticket,DEAL_POSITION_ID);
        deal_description=        GetDealDescription(deal_type,volume,symbol,order_tic
        //--- 为交易号执行格式化
        string print_index=StringFormat("% 3d",i);
        //--- 显示交易信息
        Print(print_index+": deal #",deal_ticket," at ",transaction_time,deal_descripti
    }
}

//+-----+
//| 返回操作的字符串描述 |
//+-----+
string GetDealDescription(long deal_type,double volume,string symbol,long ticket,long
{
    string descr;
    //---
    switch(deal_type)
    {
        case DEAL_TYPE_BALANCE:    return ("balance");
        case DEAL_TYPE_CREDIT:     return ("credit");
        case DEAL_TYPE_CHARGE:     return ("charge");
        case DEAL_TYPE_CORRECTION: return ("correction");
        case DEAL_TYPE_BUY:        descr="buy"; break;
        case DEAL_TYPE_SELL:       descr="sell"; break;
    }
    descr=StringFormat("%s %G %s (order #%d, position ID %d)",
        descr, // 当前描述。
        volume, // 交易的交易量。
        symbol, // 交易的交易品种。
        ticket, // 引起交易的订单的订单号。
        pos_ID // 仓位交易执行的ID号。
    );
    return(descr);
}
//---
}

```

另见

[HistorySelect\(\)](#), [HistoryDealsTotal\(\)](#), [HistoryDealSelect\(\)](#), [交易属性](#)

HistoryDealGetDouble

返回要求的交易属性，交易属性一定是双精度型，有2中变量函数可供使用。

1. 立即返回属性值。

```
double HistoryDealGetDouble(  
    ulong          ticket_number,    // 订单号  
    ENUM_DEAL_PROPERTY_DOUBLE property_id // 属性标识符  
);
```

2. 依据函数成功与否返回真值或者错误值。如果成功，目标变量的属性值通过引用传递到最后参量。

```
bool HistoryDealGetDouble(  
    ulong          ticket_number,    // 订单号  
    ENUM_DEAL_PROPERTY_DOUBLE property_id, // 属性标识符  
    double&        double_var      // 这里接受属性值  
);
```

参量

ticket_number

[in] 交易订单号。

property_id

[in] 交易属性标识符。值可以是 [ENUM_DEAL_PROPERTY_DOUBLE](#) enumeration列举值中的一个。

double_var

[out] 接收要求属性值的双精度类型变量。

返回值

[双精度](#) 型值。

注释

不要弄混 [订单](#)， [交易](#) 和 [仓位](#)。 每笔交易都按照订单执行，每个仓位都是一个或多个交易的值。

另见

[HistorySelect\(\)](#)， [HistoryDealsTotal\(\)](#)， [HistoryDealSelect\(\)](#)， [交易属性](#)

HistoryDealGetInteger

返回要求交易的属性，交易属性一定是日期时间，整型。有2个变量函数可供使用。

1. 立即返回属性值。

```
long HistoryDealGetInteger(
    ulong          ticket_number,    // 订单号
    ENUM_DEAL_PROPERTY_INTEGER property_id // 属性标识符
);
```

2. 依据函数成功与否返回真值或者错误值。如果成功，目标变量的属性值通过引用传递到最后参量。

```
bool HistoryDealGetInteger(
    ulong          ticket_number,    // 订单号
    ENUM_DEAL_PROPERTY_INTEGER property_id, // 属性标识符
    long&          long_var         // 这里接受属性值
);
```

参量

ticket_number

[in] 交易订单号。

property_id

[in] 交易属性标识符。值可以是 [ENUM_DEAL_PROPERTY_INTEGER](#) 列举值中的一个。

long_var

[out] 接收要求属性值的长型变量。

返回值

[长整型](#) 值。

注释

不要弄混 [订单](#)，[交易](#) 和 [仓位](#)。每笔交易都按照订单执行，每个仓位都是一个或多个交易的值。

另见

[HistorySelect\(\)](#)，[HistoryDealsTotal\(\)](#)，[HistoryDealSelect\(\)](#)，[交易属性](#)

HistoryDealGetString

返回要求的交易属性，交易属性一定是字符串型，有2个变量函数可供使用。

1. 立即返回属性值。

```
string HistoryDealGetString(
    ulong          ticket_number,    // 订单号
    ENUM_DEAL_PROPERTY_STRING property_id // 属性标识符
);
```

2. 依据函数成功与否返回真值或者错误值。如果成功，目标变量的属性值通过引用传递到最后参量。

```
bool HistoryDealGetString(
    ulong          ticket_number,    // 订单号
    ENUM_DEAL_PROPERTY_STRING property_id, // 属性标识符
    string&        string_var       // 这里接受属性值
);
```

参量

ticket_number

[in] 交易订单号。

property_id

[in] 交易属性标识符。值可以是 [ENUM_DEAL_PROPERTY_STRING](#) 列举值中的一个。

string_var

[out] 接收要求属性值的字符串型变量。

返回值

[字符串型](#) 值。

注释

不要弄混 [订单](#)，[交易](#) 和 [仓位](#)。每笔交易都按照订单执行，每个仓位都是一个或多个交易的值。

另见

[HistorySelect\(\)](#)，[HistoryDealsTotal\(\)](#)，[HistoryDealSelect\(\)](#)，[交易属性](#)

程序端全局变量

有一组为全局变量工作的函数设置。

客户端的全局变量不能与MQL5程序中已声明的[全局范围](#)变量混淆

自从上一次访问全局变量保存在客户端四周，然后就自动删除。访问全局变量不仅要设置新值，还有访问全局变量值。

客户端的全局变量容易同时从MQL5程序加载到客户端。

函数	功能
GlobalVariableCheck	检测带有指定名的全局变量的存在性
GlobalVariableTime	返回上次访问全局变量的时间
GlobalVariableDel	删除全局变量
GlobalVariableGet	返回全局变量值
GlobalVariableName	全局变量列表中依据序列号返回全局变量名
GlobalVariableSet	为全局变量设新值
GlobalVariablesFlush	强制在磁盘 保存全局变量内容
GlobalVariableTemp	给全局变量设新值，只在程序端当前状态存在
GlobalVariableSetOnCondition	根据状态设置现存全局变量的新值
GlobalVariablesDeleteAll	删除指定前缀的全局变量
GlobalVariablesTotal	返回全局变量总数

GlobalVariableCheck

检测带有指定名称的全局变量的存在性。

```
bool GlobalVariableCheck(  
    string name    // 全局变量名称  
);
```

参量

name

[in] 全局变量名称

返回值

如果全局变量存在，返回真值，否则返回错误值。

全局变量在客户端保存自最后使用的四周，然后自动删除。

另见

[GlobalVariableTime\(\)](#)

GlobalVariableTime

当全局变量最后访问时返回时间。

```
datetime GlobalVariableTime(  
    string name    // 名称  
);
```

参量

name

[in] 全局变量名称。

返回值

函数返回上次访问的指定全局变量，调用变量获得值也需要考虑是否访问，为了获得[错误](#)细节，调用 [GetLastError\(\)](#) 函数。

注释

全局变量在客户端保存自最后使用的四周，然后自动删除。

另见

[GlobalVariableCheck\(\)](#)

GlobalVariableDel

从客户端中删除全局变量。

```
bool GlobalVariableDel(  
    string name    // 全局变量名称  
);
```

参量

name

[in] 全局变量名称。

返回值

如果成功，函数返回真值，否则返回错误值。为获得有关[错误](#)信息，有必要调用 [GetLastError\(\)](#) 函数。

注释

全局变量在客户端保存自最后使用的四周，然后自动删除。

GlobalVariableGet

返回客户端存在的全局变量存在值，有2个变量函数可以使用。

1. 立即返回全局变量值。

```
double GlobalVariableGet(  
    string name      // 全局变量名称  
);
```

2. 返回真值或者错误值取决于函数是否成功运行。如果成功，客户端全局变量通过引用将变量传递到第二参量中。

```
bool GlobalVariableGet(  
    string name,          // 全局变量名称  
    double& double_var    // 该变量包括全局变量值 variable  
);
```

参量

name

[in] 全局变量名称。

double_var

[out] 双精度函数的变量目标，接收存储在客户端的全局变量的值。

返回值

存在的全局变量值或者 [error](#) 是0，了解更多关于误差值，调用 [GetLastError\(\)](#)

注释

全局变量在客户端保存自最后使用的四周，然后自动删除。

GlobalVariableName

通过序列号返回全局变量名称。

```
string GlobalVariableName(  
    int index    // 全局变量列表中全局变量数  
);
```

参量

index

[in] 全局变量列表中的序号，应该大于或等于0，小于 [GlobalVariablesTotal\(\)](#) 函数。

返回值

全局变量的名称通过序列号在全局变量列表中体现出来，更多关于误差的细节，调用 [GetLastError\(\)](#)。

注释

全局变量在客户端保存自最后使用的四周，然后自动删除。

GlobalVariableSet

为全局变量设置新值，如果变量不存在，系统增添新的全局变量。

```
datetime GlobalVariableSet(  
    string  name,      // 全局变量名称  
    double  value      // 设置值  
);
```

参量

name

[in] 全局变量名称。

value

[in] 新数值。

返回值

如果成功，函数返回上一修正时间，否则返回0，更多关于 [错误](#) 的细节，调用 [GetLastError\(\)](#)。

注释

全局变量在客户端保存自最后使用的四周，然后自动删除。

GlobalVariablesFlush

强制存储所有全局变量信息到磁盘。

```
void GlobalVariablesFlush();
```

返回值

无返回值。

注释

当工作结束，终端编辑所有全局变量，但数据会在计算机操作失败时突然丢失。该函数允许独立控制存储全局变量过程，以防发生偶然事故。

GlobalVariableTemp

函数试图建立暂时的全局变量，如果变量不存在，系统增设一个新的暂时全局变量。

```
bool GlobalVariableTemp(  
    string name    // 全局变量名称  
);
```

参量

name

[in] 暂时全局变量名称。

返回值

如果成功，函数返回真值，否则是错误值，更多关于 [错误](#) 的细节，调用 [GetLastError\(\)](#)。

注释

暂时的全局变量只在客户端运行时存在，在客户端停止工作时自动删除。执行[GlobalVariablesFlush\(\)](#) 暂时全局变量时不编辑到磁盘里。

建立暂时全局变量以后，可以像 [客户端全局变量](#) 一样访问并修改。

GlobalVariableSetOnCondition

如果当前值等于第三参量check_value，设置现有全局变量的新值。如果没有全局变量，函数将生成错误ERR_GLOBALVARIABLE_NOT_FOUND (4501) 并且返回false。

```
bool GlobalVariableSetOnCondition(  
    string name,           // 全局变量名称  
    double value,          // 条件是true下的变量新值  
    double check_value     // 检测值的状态  
);
```

参量

name

[in] 全局变量名称。

value

[in] 新值。

check_value

[in] 检测当前全局变量值。

返回值

如果成功，函数返回真值，否则返回错误值。更多关于 [error](#) 的细节，调用 [GetLastError\(\)](#)。如果当前全局变量值不同于check_value，函数返回false。

注释

函数将为全局变量提供自动通道，所以它可以提供一个信号量在同一个客户终端内同时被几个EA调用。

GlobalVariablesDeleteAll

删除客户端的全局变量。

```
int GlobalVariablesDeleteAll(  
    string    prefix_name=NULL,    // 所有带有前缀的全局变量  
    datetime  limit_data=0        // 所有该日期前已更改的全局变量  
);
```

参量

prefix_name=NULL

[in] 将被删除的全局变量的命名前缀。如果指定前缀是NULL或者空字符串，所有满足数据标准的变量都会被删除。

limit_data=0

[in] 通过上次修正时间来选择全局变量，函数删除在此日期之前改变的全局变量，如果参量是0，所有满足第一标准（前缀）的变量都会被删除。

返回值

删除变量数量。

Note

如果每个选项等于0（*prefix_name = NULL and limit_data = 0*），函数删除终端所有的全局变量。如果所有的参量都是制定的，删除与该参量相关的全局变量。

全局变量在客户端保存自最后使用的四周，然后自动删除。

GlobalVariablesTotal

返回客户端所有全局变量的数量。

```
int GlobalVariablesTotal();
```

返回值

全局变量数量。

注释

全局变量在客户端保存自最后使用的四周，然后自动删除。调用全局变量不仅需要设立新值，也需要读取全局变量的值。

文件函数

这是一组文档工作函数。

出于安全考虑，工作文件必须严格由MQL5语言管理。使用MQL5实施文件操作的文件意味着，不能在文件沙箱外。

有两个目录（附带子目录）放置工作文档：

- terminal_ path\MQL5\FILES\（终端菜单中选择浏览"文件" - "打开数据目录"）；
- 安装在电脑中所有终端的普通文件夹-通常在C:\Documents and Settings\All Users\Application Data\MetaQuotes\Terminal\Common\目录中。

获得目录名称的程序方式，使用[TerminalInfoString\(\)](#) 函数，使用[ENUM_TERMINAL_INFO_STRING](#) 表达式：

```
//--- 存储程序端数据的文件夹
string terminal_data_path=TerminalInfoString(TERMINAL_DATA_PATH);
//--- 客户端常用文件夹
string common_data_path=TerminalInfoString(TERMINAL_COMMONDATA_PATH);
```

禁止从其他目录使用文件。

函数	功能
FileFindFirst	依照指定过滤器启动目录中的文件搜索
FileFindNext	继续通过FileFindFirst() 函数启动的搜索
FileFindClose	结束搜索程序
FileOpen	打开指定名字和标记的文件
FileDelete	删除指定文件
FileFlush	将输入/输出文件缓冲区的所有数据写入磁盘
FileIsEnding	读取过程中定义文件末端
FileIsLineEnding	读取过程中定义文本行末端
FileClose	关闭之前打开的文件
FileIsExist	检测文件存在性
FileCopy	从本地或者共享文件夹复制原文件到另一个文件
FileMove	移动或者重命名文件
FileReadArray	读取除了BIN类型文件字符串外的任何类型数组
FileReadBool	读取CSV类型文件当前位置字符串直到定界符（或者直到文本行末端）和转换读取的字符串到布尔型值
FileReadDatetime	读取CSV类型文件以"YYYY.MM.DD HH:MI:SS", "YYYY.MM.DD" or "HH:MI:SS"格式字符串并且转换它到日期时间值
FileReadDouble	从文件指针当前位置读取双精度值

FileReadFloat	从文件指针当前位置读取浮点值
FileReadInteger	从文件指针当前位置读取整型，短整型或者字符型值
FileReadLong	从文件指针当前位置读取一个长整型值
FileReadNumber	从CSV类型文件读取当前位置字符串直到定界符（或者直到文本行末端）并且转换读取字符串到双精度值
FileReadString	在文件中文件指针当前位置读取字符串
FileReadStruct	文件指针当前位置读取二进制文件内容到作为参量传递的结构
FileSeek	根据指定位置有关的指定二进制数移动文件指针位置
FileSize	返回相应打开文件大小
FileTell	返回相应打开文件的文件指针的当前位置
FileWrite	向CSV或者TXT类型文件写入数据
FileWriteArray	写入字符串以外任何类型数组到BIN类型文件
FileWriteDouble	从文件指针当前位置写入双精度型值到二进制文件
FileWriteFloat	从文件指针当前位置写入浮点型值到二进制文件
FileWriteInteger	从文件指针当前位置写入整型值到二进制文件
FileWriteLong	从文件指针当前位置写入长整型值到二进制文件
FileWriteString	从文件指针当前位置写入字符串参量值到BIN或者TXT文件
FileWriteStruct	从文件指针当前位置写入作为参量传递的结构到二进制文件
FolderCreate	
FolderDelete	清除选定目录。如果文件夹不是空的，那么无法清除。
FolderClean	删除指定文件夹中所有文件

如果文件打开使用 [FileOpen\(\)](#) 函数编写，那么路径中所有指定的子文件夹将会创建。

FileFindFirst

函数在目录中开始文件搜索与其类似的过滤器。

```
long FileFindFirst(
    string   file_filter,           // 字符串 - 搜寻过滤器
    string&   returned_filename,    // 找到的文件名
    int      common_flag           // 定义搜索
);
```

参量

file_filter

[in] 搜索过滤器。与Files 目录相关联的子目录（或者嵌套在子目录里的序列），需要寻找文件的地方，指定在过滤器中

returned_filename

[out] 返回常量，如果成功了，放置第一个发现的文件名称。

common_flag

[in] [标记](#) 决定文件位置，如果common_flag = FILE_COMMON,文件为所有客户端放置共享文件夹。否则，文件放置在本地文件夹中。

返回值

返回对象搜索处理，通过[FileFindNext\(\)](#) 函数可以某种程度使用文件，当没有与过滤器相一致的文件（在特殊情况下-当目录是空的时候），使用INVALID_HANDLE。在搜索处理器后必须使用 [FileFindClose\(\)](#) 函数关闭。

注释

出于安全考虑，工作文件必须严格由MQL5语言管理。使用MQL5实施文件操作的文件意味着，不能在文件沙箱外。

示例：

```
void OnStart()
{
    string filename;
    //---
    int found=FileFindFirst("*.*",filename);
    if(found!=INVALID_HANDLE)
        Print("FileFindFirst returned ",filename);
    else
    {
        Print("Files not found!!! Continue search in Common");
        found=FileFindFirst("*.*",filename,FILE_COMMON);
        if(found!=INVALID_HANDLE)
            Print("FileFindFirst in Common returned ",filename);
    }
}
```

另见

[FileFindNext](#), [FileFindClose](#)

FileFindNext

函数继续通过 [FileFindFirst\(\)](#) 搜索起始位置。

```
bool FileFindNext(  
    long      search_handle,      // 搜索处理程序  
    string&    returned_filename  // 找到的文件名  
);
```

参量

search_handle

[in] 搜索处理，通过 [FileFindFirst\(\)](#) 恢复。

returned_filename

[out] 下一文件名称或者发现目录。

返回值

如果成功，返回真值，否则返回错误值。

示例：

```
void OnStart()  
{  
    string filename;  
    int i=0;  
    //---  
    int search=FileFindFirst("*.\"",filename);  
    if(search!=INVALID_HANDLE)  
    {  
        Print("FileFindFirst returned ",filename);  
        while(FileFindNext(search,filename))  
        {  
            i++;  
            Print(i,":",filename);  
        }  
        FileFindClose(search);  
    }  
    else  
        Print("Files not found!!!");  
}
```

另见

[FileFindFirst](#), [FileFindClose](#)

FileFindClose

函数关闭搜索处理。

```
void FileFindClose(  
    long search_handle    // 搜索处理程序  
);
```

参量

search_handle

[in] 搜索处理，通过 [FileFindFirst\(\)](#) 恢复。

返回值

无值返回。

注释

函数必须调用到空出来的系统资源。

FileIsExist

检测文件是否存在。

```
bool FileIsExist(  
    string file_name,      // 文件名  
    int common_flag=0      // 搜索区  
);
```

参量

file_name

[in] 被检测的文件名

common_flag=0

[in] [标记](#) 决定文件位置，如果common_flag = FILE_ COMMON,文件为所有客户端放置共享文件夹。否则，文件放置在本地文件夹中。

返回值

如果指定文件存在返回真值。

注释

出于安全考虑，工作文件必须严格由MQL5语言管理。使用MQL5实施文件操作的文件意味着，不能在文件沙箱外。

如果common_flag = FILE_ COMMON,函数为所有客户端在共享文件中寻找文件，否则函数在本地文件夹中寻找文件 (MQL5 \ Files or MQL5 \ Tester \ Files 检测状态下) 。

FileOpen

函数打开文件的指定名称和标签。

```
int FileOpen(
    string file_name,           // 文件名
    int open_flags,            // 标记的组合
    short delimiter='\t'       // 定界符
    uint codepage=CP_ACP       // 代码页
);
```

参量

file_name

[in] 文件名称包括子文件夹，如果文件为编辑开放，就会建立子文件夹。

open_flags

[in] [标记的组合](#) 决定文件夹的操作方式，标签如下定义

FILE_READ为阅读展开

FILE_WRITE 为编辑展开

FILE_BIN 二次编辑阅读模式（不会从字符串到字符串转换）

FILE_CSV csv类型文件（所有记录项目转换成统一码字符串或者ansi类型，被定界符分开）

FILE_TXT 简单文本文件（与csv相同，但不考虑定界符）

FILE_ANSI ANSI类型线（单一字节交易品种）

FILE_UNICODE UNICODE类型线（双精度函数字符）

FILE_SHARE_READ 从各自程序中共享阅读

FILE_SHARE_WRITE 从各自程序中共享编辑

FILE_COMMON 所有客户端的共享文件夹中的文件位置

delimiter='\t'

[in] 在txt或者csv文件中使用分隔符的值，如果csv文件标识符不是指定的，默认到标签，如果txt文件标识符不是制定的，就不使用分隔符。如果分隔符清晰建立成0，不使用分隔符。

codepage=CP_ACP

[in] 代码页的值，最多使用 [代码页](#) 提供占用常量。

返回值

如果文件成功打开，函数返回文件处理，然后访问文件数据，失败后返回 [INVALID_HANDLE](#)。

注释

出于安全考虑，工作文件必须严格由MQL5语言管理。使用MQL5实施文件操作的文件意味着，不能在文件沙箱外。

文件在客户端子文件 MQL5\files（或者 [\files](#)）中打开，如果FILE_COMMON在制定标签中，文件在所有MT5客户端的共享文件中打开。

示例：

```
//+-----+
//| 启动函数的脚本程序 |
//+-----+
```

```

void OnStart()
{
//--- 错误的打开文件方法
string terminal_data_path=TerminalInfoString(TERMINAL_DATA_PATH);
string filename=terminal_data_path+"\\MQL5\\Files\\"+"fractals.csv";
int filehandle=FileOpen(filename,FILE_WRITE|FILE_CSV);
if(filehandle<0)
{
    Print("Failed to open the file by the absolute path ");
    Print("Error code ",GetLastError());
}

//--- “文件沙盒效应”中正确的工作方法
ResetLastError();
filehandle=FileOpen("fractals.csv",FILE_WRITE|FILE_CSV);
if(filehandle!=INVALID_HANDLE)
{
    FileWrite(filehandle,TimeCurrent(),Symbol(),PERIOD_CURRENT);
    FileClose(filehandle);
    Print("FileOpen OK");
}
else Print("Operation FileOpen failed, error ",GetLastError());
//--- 在MQL5\Files\中创建附着目录的另一个示例
string subfolder="Research";
filehandle=FileOpen(subfolder+"\\fractals.txt",FILE_WRITE|FILE_CSV);
if(filehandle!=INVALID_HANDLE)
{
    FileWrite(filehandle,TimeCurrent(),Symbol(),PERIOD_CURRENT);
    FileClose(filehandle);
    Print("The file must be created in the folder "+terminal_data_path+"\\ "+subfolder);
}
else Print("File open failed, error ",GetLastError());
}

```

另见

[FileFindFirst](#), [创建文件夹](#), [打开文件标记](#)

FileClose

通过 [FileOpen\(\)](#) 关闭先前打开的文件。

```
void FileClose(  
    int file_handle    // 文件句柄  
);
```

参量

file_handle

[in] 通过 FileOpen() 返回文件说明符。

返回值

无值返回。

FileCopy

函数从本地或者共享文件中复制原始文件到另一文件中。

```
bool FileCopy(  
    string src_filename,    // 源文件名称  
    int common_flag,       // 位置  
    string dst_filename,    // 目标文件名称  
    int mode_flags         // 访问模式  
);
```

参量

src_filename

[in] 复制的文件名称。

common_flag

[in] [标记](#) 决定文件位置，如果common_flag = FILE_ COMMON,文件为所有客户端放置共享文件夹。否则，文件放置在本地文件夹中。

dst_filename

[in] 结果文件名。

mode_flags

[in] [访问标记](#)。参量只包括2个标签：FILE_ REWRITE 和/或 FILE_ COMMON -忽略其他标签。如果文件已经存在，FILE_ REWRITE 标签未指定，文件就不能再编辑，函数返回错误值。

返回值

若失败函数返回false。

注释

出于安全考虑，工作文件必须严格由MQL5语言管理。使用MQL5实施文件操作的文件意味着，不能在文件沙箱外。

如果新文件已经存在，复制文件在mode_flags 参量中取决于FILE_ REWRITE 标签的可用性。

FileDelete

在客户端中删除本地文件夹的指定文件。

```
bool FileDelete(  
    string file_name      // 要删除的文件名  
    int common_flag=0     // 要删除的文件位置  
);
```

参量

file_name

[in] 文件名称。

common_flag=0

[in] [标记](#) 决定文件位置，如果common_flag = FILE_ COMMON,文件为所有客户端放置共享文件夹。否则，文件放置在本地文件夹中。

返回值

如果失败函数返回 false。

注释

出于安全考虑，工作文件必须严格由MQL5语言管理。使用MQL5实施文件操作的文件意味着，不能在文件沙箱外。

删除客户端中本地文件夹中的指定文件(MQL5\files or MQL5\tester\files in case of testing) . 如果common_flag = FILE_ COMMON，函数就会从所有客户端的共享文件中删除文件。

FileMove

从本地或共享文件夹中移动文件到另一文件夹。

```
bool FileMove(  
    string src_filename,    // 进行移动操作的文件名  
    int common_flag,       // 位置  
    string dst_filename,   // 目标文件名  
    int mode_flags         // 访问模式  
);
```

参量

src_filename

[in] 移动/重命名的文件名称。

common_flag

[in] [标记](#) 决定文件位置，如果common_flag = FILE_ COMMON,文件为所有客户端放置共享文件夹。否则，文件放置在本地文件夹中。

dst_filename

[in] 操作后的文件名称。

mode_flags

[in] [访问标记](#)。参量只包括2个标签：FILE_ REWRITE 和/或者 FILE_ COMMON -忽略另外标签。如果文件已经存在并且FILE_ REWRITE 标签未指定，文件不会再编辑，函数返回错误值。

返回值

如果失败，函数返回错误值。

注释

出于安全考虑，工作文件必须严格由MQL5语言管理。使用MQL5实施文件操作的文件意味着，不能在文件沙箱外。

如果新文件已经存在，复制文件在mode_flags 参量中取决于FILE_ REWRITE 标签的可用性。

FileFlush

在输入/输出文件缓冲区中编辑所有保留数据到磁盘。

```
void FileFlush(  
    int file_handle    // 文件句柄  
);
```

参量

file_handle

[in] 通过 [FileOpen\(\)](#) 返回文件说明符。

返回值

没有返回值。

注释

函数FileFlush () 必须在阅读和编辑文件操作之间调用。

FileIsEnding

在阅读过程中编辑文件末尾。

```
bool FileIsEnding(  
    int file_handle // 文件句柄  
);
```

参量

file_handle

[in] 通过 [FileOpen\(\)](#) 返回文件标识符。

返回值

如果编辑或移动文件指标过程中能够到达文件末尾，函数返回真值。

FileIsLineEnding

在阅读过程中定义文本文件的末尾路线。

```
bool FileIsLineEnding(  
    int file_handle // 文件句柄  
);
```

参量

file_handle

[in] 文件说明符通过 [FileOpen\(\)](#) 返回。

返回值

如果阅读txt文件或者csv文件过程中到达路线末尾，返回真值（字符CR-LF）。

FileReadArray

从BIN类型数组的除字符串类型外任意类型中阅读文件（或许是结构数组，不包括字符串，和动态数组）。

```
uint FileReadArray(  
    int file_handle           // 文件句柄  
    void array[],             // 要记录的数组  
    int start_item=0,         // 开始编写数组位置  
    int items_count=WHOLE_ARRAY // 读取计数  
);
```

参量

file_handle

[in] 通过 [FileOpen\(\)](#) 返回文件说明符。

array[]

[out] 数据可以加载到的数组。

start_item=0

[in] 书写数组的开始仓位。

items_count=WHOLE_ARRAY

[in] 阅读元素的数量。默认，阅读整个数组（cnt=[WHOLE_ARRAY](#)）。

返回值

阅读元素的数量。

注释

字符串数组智能从TXT类型文件中阅读，如果必要的话，函数试着增加数组大小。

另见

[变量](#)

FileReadBool

从当前位置CSV类型字符串中读取文件到定界符（或者到文本路线的末尾）。

```
bool FileReadBool(  
    int file_handle // 文件句柄  
);
```

参量

file_handle

[in] 通过 [FileOpen\(\)](#) 返回文件说明符。

返回值

线型阅读可以建立"true", "false"或者整数符号表示 "0" 或者 "1". 非零值转换成逻辑真值，函数返回修正值。

另见

[布尔类型](#)

FileReadDatetime

从CSV类型文件中读取字符串文件格式 "YYYY.MM.DD HH:MI:SS", "YYYY.MM.DD" 或者 "HH:MI:SS"-转换到日期时间值类型。

```
datetime FileReadDatetime(  
    int file_handle // 文件句柄  
);
```

参量

file_handle

[in] 通过 [FileOpen\(\)](#) 返回文件说明符。

返回值

日期时间类型值。

另见

[日期时间类型](#), [字符串到时间](#), [时间到字符串](#)

FileReadDouble

从当前二进制文件位置阅读双精度浮点编号（双精度）。

```
double FileReadDouble(  
    int file_handle // 文件句柄  
);
```

参量

file_handle

[in] 通过 [FileOpen\(\)](#) 返回文件说明符。

返回值

双精度类型值。

注释

关于误差的更多细节，调用 [GetLastError\(\)](#)。

另见

[真实型（双精度型，浮点型）](#), [字符串到双精度](#), [双精度到字符串](#)

FileReadFloat

从当前二进制文件位置阅读双单精度浮点编号（浮点）。

```
float FileReadFloat(  
    int file_handle // 文件句柄  
);
```

参量

file_handle

[in] 通过 [FileOpen\(\)](#) 返回文件说明符

返回值

浮点类型值。

注释

更多关于误差细节，调用 [GetLastError\(\)](#)。

另见

[真实型（双精度型，浮点型）](#), [FileReadDouble](#)

FileReadInteger

从当前文件指示位置，函数读取整型，短型或图表型值，取决于规定字节长度。

```
int FileReadInteger(  
    int file_handle           // 文件句柄  
    int size=INT_VALUE       // 字节中整数的大小  
);
```

参量

file_handle

[in] 通过 [FileOpen\(\)](#) 返回文件说明符。

size=INT_VALUE

[in] 字节数量（大于等于4），可以编辑。提供类似常量：CHAR_VALUE = 1, SHORT_VALUE = 2 和 INT_VALUE = 4, 因此函数可以读取全部图表，短型或整型值。

返回值

整型值，自从整型值返回，可以转换成整数，标记或者未标记值。文件指针通过许多字节类型阅读整数类型。

另见

[整数到字符串](#), [字符串到整数](#), [整数类型](#)

FileReadLong

从二进制文件当前位置函数读取长型整数（8字节）。

```
long FileReadLong(  
    int file_handle // 文件句柄  
);
```

参量

file_handle

[in] 通过[FileOpen\(\)](#) 返回文件说明符。

返回值

长整型值。

另见

[整数类型](#), [FileReadInteger](#)

FileReadNumber

函数从CSV文件字符串中读取当前位置分隔符（或者直到文本字符串末尾）并转变阅读字符串到双精度类型值。

```
double FileReadNumber(  
    int file_handle // 文件句柄  
);
```

参量

file_handle

[in] 通过[FileOpen\(\)](#) 返回文件描述符。

返回值

双精度型值。

FileReadString

函数从文件指标当前位置读取字符串。

```
string FileReadString(  
    int file_handle      // 文件句柄  
    int size=-1          // 字符串长度  
);
```

参量

file_handle

[in] 通过 [FileOpen\(\)](#) 返回的文件描述符。

size=-1

[in] 读取的字符数。

返回值

阅读行（字符串）。

注释

当从二倍文件阅读时，字符串的长度阅读需要指定。从txt文件中不要求阅读字符串长度，字符串从当前位置填满“\n”字节，当从csv文件阅读时，也不要求字符串长度，字符串从当前位置直到最近的定界符或者到文本字符串末尾字节开始阅读。

如果文件使用 FILE_ ANSI [标记](#)，打开，线路阅读转换到同一码。

另见

[字符串类型](#)， [数据转换](#)

FileSeek

通过指定仓位相关字节数，函数移动文件指标的仓位。

```
void FileSeek(  
    int          file_handle    // 文件句柄  
    long         offset,        // 字节  
    ENUM_FILE_POSITION origin    // 参考位置  
);
```

参量

file_handle

[in] 通过 [FileOpen\(\)](#) 返回文件描述符。

offset

[in] 字节转换（也许获取负值）。

origin

[in] 移位起始点。可以是 [ENUM_FILE_POSITION](#) 值中一个。

返回值

无值返回。

注释

如果执行FileSeek() 函数结果导致负值转换（超过文件的水平边界），文件指标在文件起始运行时建立。

如果仓位的建立超过文件的右边界（比文件大），接下来的录入文件不是从文件末尾开始，而是从仓位开始。不确定值为先前文件末尾和新建仓位编辑。

FileReadStruct

从二进制文件中函数读取以参量传递的结构，从文件指针的当前仓位开始。

```
uint FileReadStruct(  
    int file_handle // 文件句柄  
    any_simple_struct str_object, // 读取的目标结构  
    int size=-1 // 字节中结构大小  
);
```

参量

file_handle

[in] 打开二进制文件的文件说明符。

str_object

[out] 结构对象，结构不包括字符串，[动态数组](#) 或者 [虚拟函数](#)。

size=-1

[in] 可以读取字节数量，如果大小未标明或者指示值比结构大小要大，需要使用指定结构的精确大小。

返回值

如果成功，函数返回读取的字节数量，文件指标通过相同的字节数量移动。

另见

[结构和类](#)

文件大小

函数返回文件字节大小。

```
ulong FileSize(  
    int file_handle    // 文件句柄  
);
```

参量

file_handle

[in] 通过 [FileOpen\(\)](#) 返回文件描述符。

返回值

整型值。

注释

调用[GetLastError\(\)](#) 函数获得[错误](#)信息。

FileTell

文件返回打开文件指针的当前仓位。

```
ulong FileTell(  
    int file_handle    // 文件句柄  
);
```

参量

file_handle

[in] 通过 [FileOpen\(\)](#) 返回文件描述符。

返回值

从文件开始，文件说明符的当前仓位字节数。

注释

调用[GetLastError\(\)](#) 函数获得[错误](#)信息。

FileWrite

函数预定给CSV文件录入数据，除非等于0，定界符自动插入，在编辑文件中，添加最后字节线 "\n" 。

```
uint FileWrite(  
    int file_handle    // 文件句柄  
    ...                // 记录参量的列表  
);
```

参量

file_handle

[in] 通过 [FileOpen\(\)](#) 返回文件说明符。

...

[in] 参量列表由逗号分开，编辑入文件中，编辑参量数量最高为63。

返回值

编辑字节数。

注释

数量转换成文本输出（参看Print() 函数）。双精度数据以小数点后16位精确度输出，数据以传统或科学模式显示-依据该模式是最紧凑的。浮点型数据以小数点后5位显示，为了以不同精确度输出真实数字或以清晰的指定模式，使用 [DoubleToString\(\)](#)。

布尔类型数量以true" 或者 "false"字符串显示，日期时间类型数量以"YYYY.MM.DD HH:MI:SS"显示。

另见

[注释](#), [打印](#), [字符串格式化](#)

FileWriteArray

函数可以编辑除字符串类型外任何类型的数组到BIN文件中（可以是不包含字符串或动态数组的结构数组）。

```
uint FileWriteArray(  
    int    file_handle           // 文件句柄  
    void    array[],             // 数组  
    int     start_item=0,        // 数字中启动索引  
    int     items_count=WHOLE_ARRAY // 单元号  
);
```

参量

file_handle

[in] 通过 [FileOpen\(\)](#) 返回文件说明符。

array[]

[out] 记录数组。

start_item=0

[in] 数组的原始索引（第一个记录元素的号码）。

items_count=WHOLE_ARRAY

[in] 记录项目数量（[WHOLE_ARRAY](#) 表示记录所有以 *start_item* 开头的项目指导数组末尾）。

返回值

记录项目数量。

注释

字符串数组可以记录在TXT文件中，字符串可以自动以 "\r\n" 字节结尾，依据文件类型ANSI 或者 UNICODE，字符串可以转换成ANSI编码。

另见

[变量](#)

FileWriteDouble

函数编辑双精度参量到文件中，从文件指针的当前仓位开始。

```
uint FileWriteDouble(  
    int      file_handle    // 文件句柄  
    double   dvalue         // 书写值  
);
```

参量

file_handle

[in] 通过 [FileOpen\(\)](#) 返回文件说明符。

dvalue

[in] 双精度类型值。

返回值

如果成功函数返回编辑的字节数量（在此情况下 [sizeof](#)(双精度) = 8）。文件指标以相同数量字节转换。

另见

[真实型（双精度型，浮点型）](#)

FileWriteFloat

函数编辑浮点参数的值到二次文件中，从文件指标当前仓位开始。

```
uint FileWriteFloat(  
    int    file_handle    // 文件句柄  
    float  fvalue         // 被写入的值  
);
```

参量

file_handle

[in] 通过 [FileOpen\(\)](#) 返回文件说明符

fvalue

[in] 浮点类型值。

返回值

如果成功函数返回编辑的字节数量(在此情况下 [sizeof](#)(浮点类型) =4) 。文件指标以相同数量字节转换。

另见

[真实型 \(双精度型, 浮点型\)](#), [FileWriteDouble](#)

FileWriteInteger

函数编辑整型参量值到二次文件中，从文件指标当前仓位开始。

```
uint FileWriteInteger(  
    int file_handle      // 文件句柄  
    int ivalue,          // 被编写值  
    int size=INT_VALUE   // 字节大小  
);
```

参量

file_handle

[in] 通过 [FileOpen\(\)](#) 返回文件说明符

ivalue

[in] 整数值。

size=INT_VALUE

[in] 字节数量（最多包括4字节）可以编辑。提供类似常量：CHAR_VALUE=1, SHORT_VALUE=2 和 INT_VALUE=4,因此函数可以编辑图表型，双字符型，短型，长短型，整型或者长整型的整数值。

返回值

如果成功函数返回字节编辑数量，文件指标以相同数量字节转换。

另见

[整数到字符串](#), [字符串到整数](#), [整数类型](#)

FileWriteLong

函数编辑长型参量值到二进制文件中，从文件指标当前仓位开始。

```
uint FileWriteLong(  
    int   file_handle    // 文件句柄  
    long  lvalue         // 被写入的值  
);
```

参量

file_handle

[in] 通过 [FileOpen\(\)](#) 返回文件说明符。

lvalue

[in] 长整型值。

返回值

如果成功函数返回编辑字节数量(在此情况下 [sizeof](#)(长型) =8) .文件指标以相同数量字节转变。

另见

[整数类型](#), [FileWriteInteger](#)

FileWriteString

从文件指标当前仓位开始，函数编辑字符串类型参量值到BIN或者TXT文件中。编辑TXT文件：如果有交易品种以'\n' (LF) 字符串开头，并没有先前的'\r' (CR) 字节，然后在 '\n'之前添加丢失的'\r'。

```
uint FileWriteString(  
    int      file_handle      // 文件句柄  
    string   svalue,          // 要编写的字符串  
    int      size=-1          // 交易品种数量  
);
```

参量

file_handle

[in] 通过 [FileOpen\(\)](#) 返回文件说明符。

svalue

[in] 字符串。

size=-1

[in] 想要编辑的字节数量，选项需要从字符串编辑到BIN文件中，如果大小未指定，全部末尾没有0的字符串都能编辑，如果指定大小小于字符串长度，只能编辑一部分的非零字符串。如果指定大小大于字符串的长度，字符串会占用0字符。

TXT类型文件，忽略参量，字符串完全编辑。

返回值

如果成功，函数返回字节编辑数量，文件指标以相同字节数量转换。

注释

当通过FILE_ UNICODE [标记](#) (或者没有 FILE_ ANSI标签) 打开编辑文件，编辑的字节数量是字符串字节编辑的两倍大。当记录文件以FILE_ ANSI标签打开，字节编辑数量会与字符串字节编辑数量相一致。

另见

[字符串类型](#), [字符串格式化](#)

FileWriteStruct

通过传递参量函数编辑二次结构文件目录，从文件指标的当前仓位开始。

```
uint FileWriteStruct(  
    int file_handle // 文件句柄  
    any_simple_struct str_object&, // 连接一个物件  
    int size=-1 // 写入字节大小  
);
```

参量

file_handle

[in] 通过[FileOpen\(\)](#) 返回文件说明符

str_object&

[in] 参考结构对象，结构不应该包含字符串，[动态数组](#) 或者 [虚拟函数](#)。

size=-1

[in] 想要记录的自己数量，如果大小不够指定或者指定字节数量大于结构大小，编辑全部结构。

返回值

如果成功，函数返回编辑的自己数量，文件指标以相同数量字节转换。

另见

[结构和类](#)

FolderCreate

函数在文件目录下新建目录（依据 common_flag 的值）。

```
bool FolderCreate(  
    string  folder_name,      // 带有新文件夹名的字符串  
    int     common_flag=0    // 范围  
);
```

参量

folder_name

[in] 想要新建的目录名称，包括文件夹的全部路径。

common_flag=0

[in] [标记](#) 决定目录位置，如果common_flag=FILE_COMMON,那么目录在所有客户端的共享目录中，否则，目录会在本地文件夹中(MQL5\files 或者 MQL5\tester\files)。

返回值

如果成功返回true，否则-false。

注释

出于安全考虑，工作文件必须严格由MQL5语言管理。使用MQL5实施文件操作的文件意味着，不能在文件沙箱外。

FolderDelete

函数删除指定目录，如果文件夹不是空的，就不能被删除。

```
bool FolderDelete(  
    string folder_name,      // 带有要删除的文件夹名的字符串  
    int common_flag=0       // 范围  
);
```

参量

folder_name

[in] 想要删除的目录名称，包括文件夹的全部路径。

common_flag=0

[in] [Flag](#) 决定目录位置，如果common_flag=FILE_COMMON,那么目录在所有客户端的共享目录中，否则，目录会在本地文件夹中(MQL5\files 或者 MQL5\tester\files)。

返回值

如果成功返回true，否则false。

注释

出于安全考虑，工作文件必须严格由MQL5语言管理。使用MQL5实施文件操作的文件意味着，不能在文件沙箱外。

如果目录至少包括一个文件和/或子目录，目录就不能删除，必须先清理。使用 [FolderClean\(\)](#) 清理文件夹中所有文件或者子目录。

FolderClean

函数删除指定文件夹中所有文件。

```
bool FolderClean(  
    string folder_name,      // 带有已删除的文件夹名的字符串  
    int common_flag=0       // 范围  
);
```

参量

folder_name

[in] 想要删除的所有文件的目录名称，包括文件夹的全部路径。

common_flag=0

[in] [标记](#) 决定目录位置，如果common_flag=FILE_COMMON,那么目录在所有客户端的共享目录中，否则，目录会在本地文件夹中(MQL5\files 或者 MQL5\tester\files)。

返回值

如果成功返回true，否则false。

注释

出于安全考虑，工作文件必须严格由MQL5语言管理。使用MQL5实施文件操作的文件意味着，不能在文件沙箱外。

该函数慎重使用，所有文件和子目录会被彻底删除。

自定义指标

这是一组用来创建自定义指标的函数，这些函数不能用于编辑EA交易和脚本。

函数	功能
SetIndexBuffer	将指定指标缓冲区和一维 双精度 动态 数组 绑定一起
IndicatorSetDouble	设置 双精度 型指标属性值
IndicatorSetInteger	设置 整型 型指标属性值
IndicatorSetString	设置 字符串 指标属性值
PlotIndexSetDouble	设置 双精度 型指标行属性值
PlotIndexSetInteger	设置 整型 指标行属性值
PlotIndexSetString	设置 字符串 指标行属性值
PlotIndexGetInteger	返回 整数 型指标行属性值

自定义指标中所有必要的运算都在预定义的[OnCalculate\(\)](#) 函数中，如果短时间使用OnCalculate() 函数调用，如

```
int OnCalculate (const int rates_total, const int prev_calculated, const int begin, c
```

而后rates_total参量包括price[]数组中的全部元素数量值，以输入参量传递到计算指标值中。

参量 prev_calculated 是先前调用的OnCalculate() 计算结果，允许为计算指标值组织一种节约算法，例如，如果当前值是rates_total = 1000, prev_calculated = 999, 足够为每个指标缓冲区计算值。

如果是关于输入数组价格的信息是不能利用的，它会为每个指标缓冲区计算1000个必要值。第一次调用OnCalculate() value prev_calculated = 0，如果price[]数组在某种程度上改变，case prev_calculated 会等于0。

begin参量表示价格数组原始值的数量，不包括计算数据。例如，如果Accelerator Oscillator 值（开头37个值没有计算）以输入参量使用，那么begin = 37。例如，列举一个简单指标：

```
#property indicator_chart_window
#property indicator_buffers 1
#property indicator_plots 1
//---- 绘图标签1
#property indicator_label1 "Label1"
#property indicator_type1 DRAW_LINE
#property indicator_color1 clrRed
#property indicator_style1 STYLE_SOLID
#property indicator_width1 1
//--- 指标缓冲区
```

```

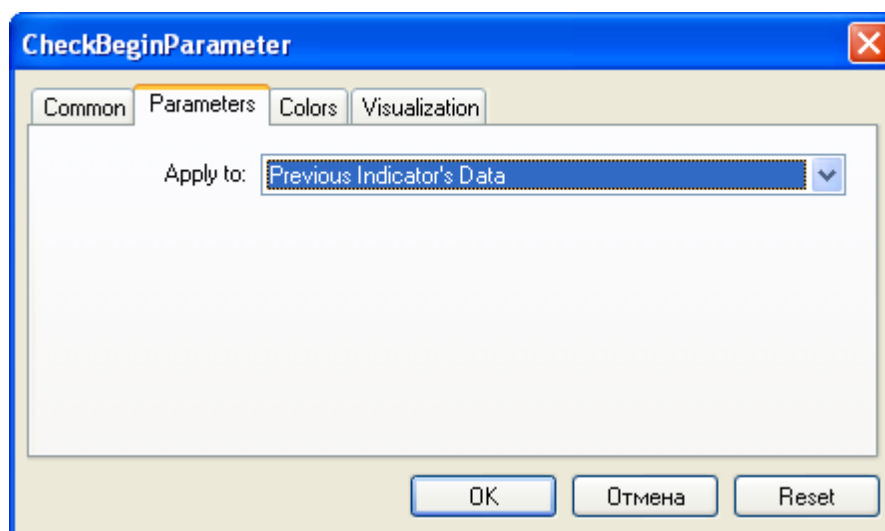
double          Label1Buffer[];

//+-----+
//| 自定义指标初始化函数          |
//+-----+
void OnInit()
{
//--- 指标缓冲区绘图
    SetIndexBuffer(0,Label1Buffer,INDICATOR_DATA);
//---
}
//+-----+
//| 自定义指标重复函数          |
//+-----+
int OnCalculate(const int rates_total,
                const int prev_calculated,
                const int begin,
                const double &price[])

{
//---
    Print("begin = ",begin," prev_calculated = ",prev_calculated," rates_total = ",rates_total);
//--- 为下一次调用返回prev_calculated值
    return(rates_total);
}

```

从"导航器" 界面离开到加速震荡指标窗口，我们认为该计算一定以先前指标的值为基础。



结果是，第一次调用prev_calculated的OnCalculate() 值等于0，之后调用等于rates_total 值（直到增加价格图表的柱的数量）。



开始参量的值直接等于原始字节数量，加速指标的值不能根据指标逻辑值来计算。如果浏览自定义指标加速器mq5的源代码，可以看到 `OnInit()` 函数线：

```
// --- 从被画的指数中设置第一柱
PlotIndexSetInteger(0, PLOT_DRAW_BEGIN, 37);
```

使用函数 `PlotIndexSetInteger(0, PLOT_DRAW_BEGIN, empty_first_values)`，可以建立自定义指标的指标数组中非存在第一值的编号，这样，我们不需要接收计算 `(empty_first_values)`。因此，可以机械使用：

1. 建立指标的原始值数字，不能用了计算另外自定义指标；
2. 当调用另一自定义指标时，获得关于第一忽略值的数量信息，不需要进入计算逻辑值。

MetaTrader 538

MQL5

MQL5 -

1.
2. () ,
3.
4. () ,
5. () ,
6.
7.

double,

#property indicator_buffers 3 // количество буферов
#property indicator_plots 2 // количество графических построений

MQL5

18:

<u>DRAW_NONE</u>		1	-

	,		
	"		
	"		
<u>DRAW_LINE</u>			
	(1	-
)		
<u>DRAW_SECTION</u>		1	-

	()		
<u>DRAW_HISTOGRAM</u>	()	1	-
<u>DRAW_HISTOGRAM2</u>		2	-

	()		
<u>DRAW_ARROW</u>	()	1	-
<u>DRAW_ZIGZAG</u>	<u>DRAW</u> <u>SECTIO</u> <u>N,</u>	2	-

<u>DRAW_COLOR_HISTOGRAM2</u>	<u>DRAW_HISTOGRAM2,</u>	2	1
<u>DRAW_COLOR_ARROW</u>	<u>DRAW_ARROW,</u>	1	1

DRAW_COLOR_ZIGZAG	DRAW_ZIGZAG	2	1
DRAW_COLOR_BARS	DRAW_BARS	4	1

<u>DRAW_COLOR_CANDLES</u>	<u>DRAW_CANDLE</u> <u>S</u>	4	1

double,

SetIndexBuffer().

'

-

'

.

"

"

.

'

```
ArraySetAsSeries( )
```

```
( INDICATOR_DATA ) .
```

```
( INDICATOR_CALCULATIONS ) .
```

```
( INDICATOR_COLOR_INDEX ) .
    color,
    double.
( INDICATOR_CALCULATIONS )
    CopyBuffer( ) .
```

```
#property indicator_buffers
```

```
#property indicator_buffers 3 // индикатор имеет 3 буфера
```

```
- 512.
```

```
Open, High, Low Close.
```

```
4
```

```
double
```

```
4
```

```
:
```

```
//--- в индикаторе четыре индикаторных буфера
#property indicator_buffers 4
//--- в индикаторе одно графическое построение
#property indicator_plots 1
//--- графическое построение под номером 1 будет отображаться японскими свечами
#property indicator_type1 DRAW_CANDLES
//--- японские свечи будут рисоваться цветом clrDodgerBlue
#property indicator_color1 clrDodgerBlue
//--- 4 массива под индикаторные буферы
double OBuffer[];
double HBuffer[];
double LBuffer[];
double CBuffer[];
```

4

4

SetIndexBuffer()

```
//--- связывание массивов с индикаторными буферами
SetIndexBuffer(0,OBuffer,INDICATOR_DATA); // первый буфер соответствует нулевому
SetIndexBuffer(1,HBuffer,INDICATOR_DATA); // второй буфер соответствует индексу 1
SetIndexBuffer(2,LBuffer,INDICATOR_DATA); // третий буфер соответствует индексу
SetIndexBuffer(3,CBuffer,INDICATOR_DATA); // четвертый буфер соответствует индексу
```

"

"

- DRAW_LINE.

1,

2.

```
//--- в индикаторе четыре индикаторных буфера
#property indicator_buffers 4
//--- в индикаторе одно графическое построение
#property indicator_plots 1
//--- графическое построение под номером 1 будет отображаться японскими свечами
#property indicator_type1 DRAW_CANDLES
//--- японские свечи будут рисоваться цветом clrDodgerBlue
#property indicator_color1 clrDodgerBlue
//--- 4 массива под индикаторные буферы
```

```
double OBuffer[];
double HBuffer[];
double LBuffer[];
double CBuffer[];
```

```
SetIndexBuffer(0,LineBuffer,INDICATOR_DATA); // второй буфер соответствует индекс
//--- связывание массивов с индикаторными буферами под японские свечи
SetIndexBuffer(1,OBuffer,INDICATOR_DATA); // второй буфер соответствует индекс
SetIndexBuffer(2,HBuffer,INDICATOR_DATA); // третий буфер соответствует индекс
SetIndexBuffer(3,LBuffer,INDICATOR_DATA); // четвертый буфер соответствует индекс
SetIndexBuffer(4,CBuffer,INDICATOR_DATA); // пятый буфер соответствует индекс
```

COLOR,

- DRAW__LINE
- DRAW__SECTION
- DRAW__HISTOGRAM
- DRAW__HISTOGRAM2
- DRAW__ARROW
- DRAW__ZIGZAG
- DRAW__FILLING
- DRAW__BARS
- DRAW__CANDLES

COLOR,

- DRAW__COLOR__LINE
- DRAW__COLOR__SECTION
- DRAW__COLOR__HISTOGRAM
- DRAW__COLOR__HISTOGRAM2

- DRAW_COLOR_ARROW
- DRAW_COLOR_ZIGZAG
- DRAW_COLOR_BARS
- DRAW_COLOR_CANDLES

- 64.

64-

#property indicator_color,

```
//--- зададим 8 цветов для раскраски свечей (они хранятся в специальном массиве)
#property indicator_color1 clrRed,clrBlue,clrGreen,clrYellow,clrMagenta,clrCyan,clrL
```

8

1

0.

```
//--- зададим цвет свечи clrRed
col_buffer[buffer_index]=0;
```

PlotIndexSetInteger() . :

```
//--- установим цвет для каждого индекса как свойство PLOT_LINE_COLOR
PlotIndexSetInteger(0, // номер графического стиля
                    PLOT_LINE_COLOR, // идентификатор свойства
                    plot_color_ind, // индекс цвета, куда запишем цвет
                    color_array[i]); // новый цвет
```

N

PLOT DRAW BEGIN

N

```
//--- связывание массивов с индикаторными буферами под японские свечи
PlotIndexSetInteger(номер_графического_построения,PLOT_DRAW_BEGIN,N);
```

•

•

indicator_plots-1 (

) .

- N -

DRAW_NONE

DRAW_NONE

"

IndicatorSetInteger
(INDICATOR_DIGITS, OnInit()) :

```
int OnInit()
{
//--- indicator buffers mapping
    SetIndexBuffer(0, InvisibleBuffer, INDICATOR_DATA);
//--- установим точность, с которой значение будет показываться в Окне данных
    IndicatorSetInteger(INDICATOR_DIGITS, 0);
//---
    return(0);
}
```

DRAW_NONE -

1.



```
//+-----+
//|                                     DRAW_NONE.mq5 |
//|                                     Copyright 2011, MetaQuotes Software Corp. |
//|                                     http://www.mql5.com |
//+-----+

#property copyright "Copyright 2011, MetaQuotes Software Corp."
#property link      "http://www.mql5.com"
#property version   "1.00"
#property indicator_chart_window
#property indicator_buffers 1
#property indicator_plots  1
//--- plot Invisible
#property indicator_label1  "Bar Index"
#property indicator_type1   DRAW_NONE
#property indicator_style1  STYLE_SOLID
#property indicator_color1  clrRed
#property indicator_width1  1
//--- indicator buffers
double      InvisibleBuffer[];
//+-----+
//| Custom indicator initialization function |
//+-----+

int OnInit()
{
//--- связывание массива и индикаторного буфера
    SetIndexBuffer(0, InvisibleBuffer, INDICATOR_DATA);
//--- установим точность, с которой значение будет показываться в "Окне данных"
    IndicatorSetInteger(INDICATOR_DIGITS, 0);
//---
    return(0);
}

//+-----+
//| Custom indicator iteration function |
//+-----+

int OnCalculate(const int rates_total,
               const int prev_calculated,
               const datetime &time[],
               const double &open[],
               const double &high[],
               const double &low[],
               const double &close[],
               const long &tick_volume[],
               const long &volume[],
               const int &spread[])
{
    static datetime lastbar=0;
//--- если это первый расчет индикатора
```

```

if (prev_calculated==0)
{
    //--- перенумеруем бары в первый раз
    CalcValues(rates_total,close);
    //--- запомним время открытия текущего бара в lastbar
    lastbar=(datetime)SeriesInfoInteger(_Symbol,_Period,SERIES_LASTBAR_DATE);
}
else
{
    //--- если появился новый бар, время его открытия не совпадает с lastbar
    if(lastbar!=SeriesInfoInteger(_Symbol,_Period,SERIES_LASTBAR_DATE))
    {
        //--- перенумеруем бары заново
        CalcValues(rates_total,close);
        //--- обнулим время открытия текущего бара в lastbar
        lastbar=(datetime)SeriesInfoInteger(_Symbol,_Period,SERIES_LASTBAR_DATE);
    }
}
//--- return value of prev_calculated for next call
return(rates_total);
}
//+-----+
//| нумерует бары как в таймсерии |
//+-----+
void CalcValues(int total,double const &array[])
{
    //--- зададим индикаторному буферу индексацию как в таймсерии
    ArraySetAsSeries(InvisibleBuffer,true);
    //--- заполним каждому бару его номер
    for(int i=0;i<total;i++) InvisibleBuffer[i]=i;
}

```

DRAW__LINE

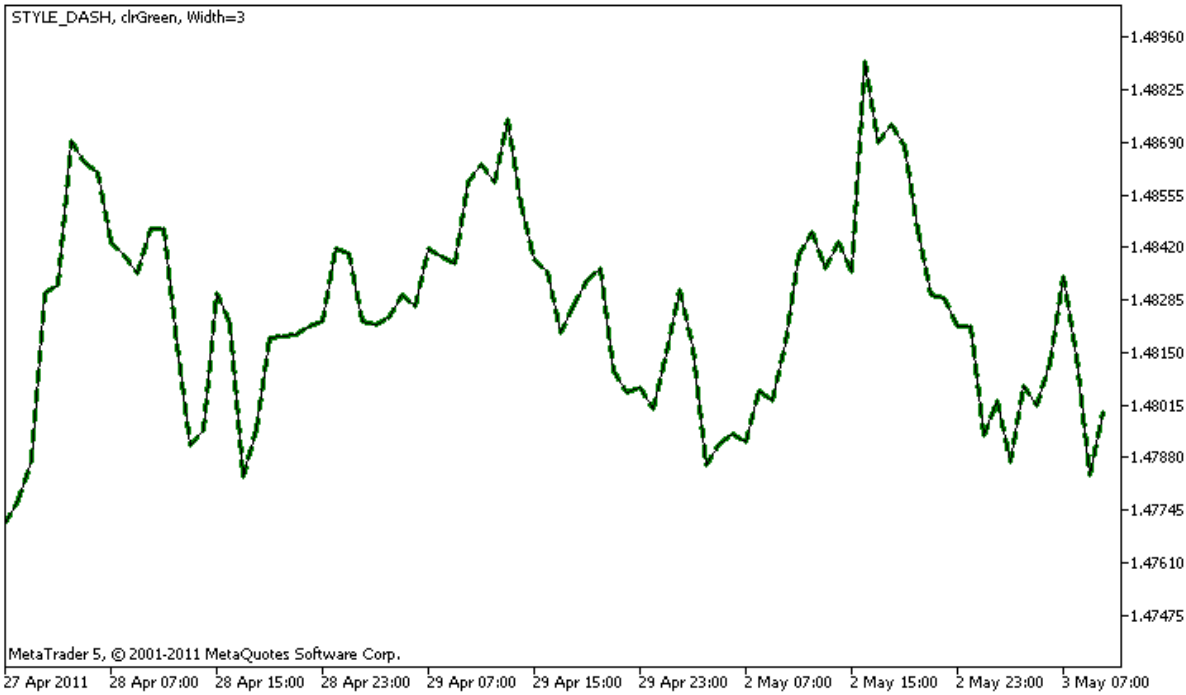
DRAW__LINE

[PlotIndexSetInteger\(\)](#).

DRAW__LINE - 1.

Close.

N=5



plot1

DRAW__LINE

[#property](#),

[OnCalculate\(\)](#)

N

)

```
//+-----+
//|                                     DRAW__LINE.mq5 |
//|                                     Copyright 2011, MetaQuotes Software Corp. |
//|                                     http://www.mql5.com |
//+-----+
#property copyright "Copyright 2011, MetaQuotes Software Corp."
```

```

#property link      "http://www.mql5.com"
#property version   "1.00"

#property description "Индикатор для демонстрации DRAW_LINE"
#property description "Рисует линию заданным цветом по ценам Close"
#property description "Цвет, толщина и стиль линии меняется случайным образом"
#property description "через каждые N тиков"

#property indicator_chart_window
#property indicator_buffers 1
#property indicator_plots 1
//--- свойства линии заданы с помощью директив компилятора
#property indicator_label1 "Line" // название построения для "Окна данных"
#property indicator_type1 DRAW_LINE // тип графического построения - линия
#property indicator_color1 clrRed // цвет линии
#property indicator_style1 STYLE_SOLID // стиль линии
#property indicator_width1 1 // толщина линии
//--- input параметр
input int N=5; // кол-во тиков для изменения
//--- индикаторный буфер для построения
double LineBuffer[];
//--- массив для хранения цветов
color colors[]={clrRed,clrBlue,clrGreen};
//--- массив для хранения стилей отрисовки линии
ENUM_LINE_STYLE styles[]={STYLE_SOLID,STYLE_DASH,STYLE_DOT,STYLE_DASHDOT,STYLE_DASHDO
//+-----+
//| Custom indicator initialization function |
//+-----+
int OnInit()
{
//--- связывание массива и индикаторного буфера
SetIndexBuffer(0,LineBuffer,INDICATOR_DATA);
//--- инициализация генератора псевдослучайных чисел
MathSrand(GetTickCount());
//---
return(0);
}
//+-----+
//| Custom indicator iteration function |
//+-----+
int OnCalculate(const int rates_total,
               const int prev_calculated,
               const datetime &time[],
               const double &open[],
               const double &high[],
               const double &low[],
               const double &close[],
               const long &tick_volume[],

```

```

        const long &volume[],
        const int &spread[])
    {
        static int ticks=0;
//--- считаем тики для изменения стиля, цвета и толщины линии
        ticks++;
//--- если накопилось критическое число тиков
        if(ticks>=N)
        {
            //--- меняем свойства линии
            ChangeLineAppearance();
            //--- сбрасываем счетчик тиков в ноль
            ticks=0;
        }

//--- блок расчета значений индикатора
        for(int i=0;i<rates_total;i++)
        {
            LineBuffer[i]=close[i];
        }

//--- вернем значение prev_calculated для следующего вызова функции
        return(rates_total);
    }
//+-----+
//| изменяет внешний вид отображаемой линии в индикаторе |
//+-----+
void ChangeLineAppearance()
{
    //--- строка для формирования информации о свойствах линии
    string comm="";
    //--- блок изменения цвета линии
    //--- получим случайное число
    int number=MathRand();
    //--- делитель числа равен размеру массива colors[]
    int size=ArraySize(colors);
    //--- получим индекс для выбора нового цвета как остаток от целочисленного деления
    int color_index=number%size;
    //--- установим цвет как свойство PLOT_LINE_COLOR
    PlotIndexSetInteger(0,PLOT_LINE_COLOR,colors[color_index]);
    //--- запишем цвет линии
    comm=comm+(string)colors[color_index];

    //--- блок изменения толщины линии
    number=MathRand();
    //--- получим толщину как остаток от целочисленного деления
    int width=number%5; // толщина задается от 0 до 4
    //--- установим цвет как свойство PLOT_LINE_WIDTH

```

```
PlotIndexSetInteger(0,PLOT_LINE_WIDTH,width);
//--- запишем толщину линии
comm=comm+", Width="+IntegerToString(width);

//--- блок изменения стиля линии
number=MathRand();
//--- делитель числа равен размеру массива styles
size=ArraySize(styles);
//--- получим индекс для выбора нового стиля как остаток от целочисленного деления
int style_index=number%size;
//--- установим цвет как свойство PLOT_LINE_COLOR
PlotIndexSetInteger(0,PLOT_LINE_STYLE,styles[style_index]);
//--- запишем стиль линии
comm=EnumToString(styles[style_index])+", "+comm;
//--- выведем информацию на график через комментарий
Comment(comm);
}
```

DRAW_SECTION

DRAW_SECTION

DRAW_LINE - PlotIndexSetInteger().

PLOT_EMPTY_VALUE.

```
//--- значение 0 (пустое значение) не будет участвовать в отрисовке
PlotIndexSetDouble(индекс_построения_DRAW_SECTION, PLOT_EMPTY_VALUE, 0);
```

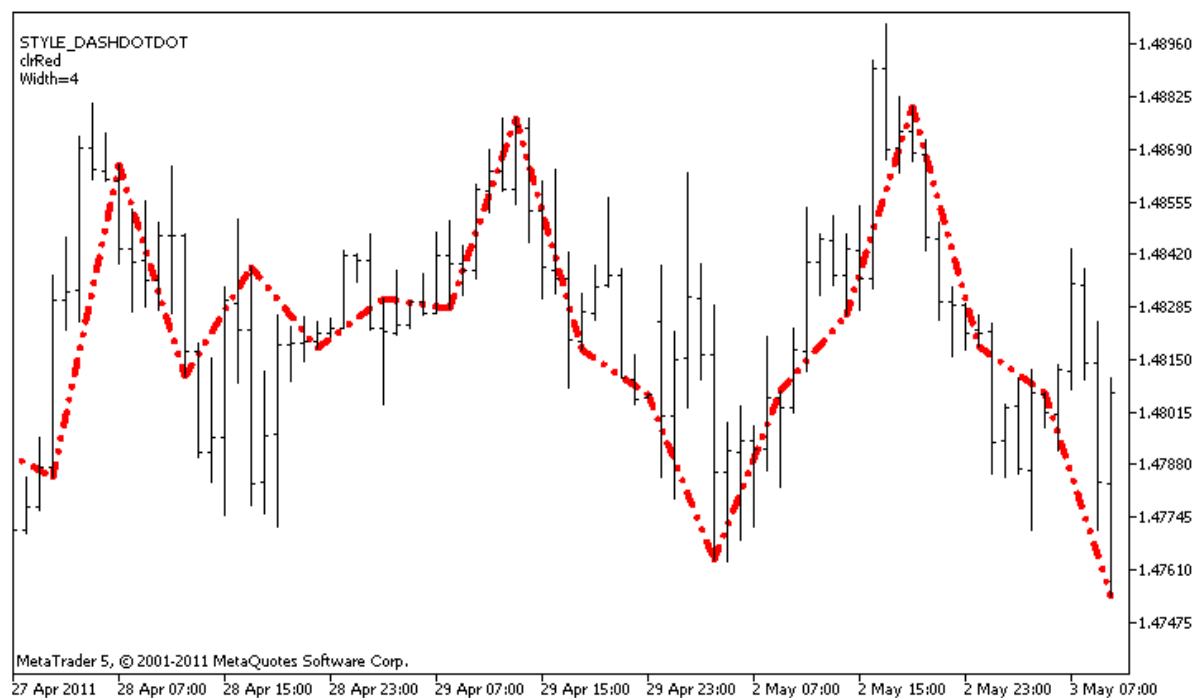
DRAW_SECTION

- 1.

Low.

High

N



plot1

DRAW_SECTION

#property,

OnCalculate()

N

).

```
//+-----+
//|                                     DRAW_SECTION.mq5 |
//|                                     Copyright 2011, MetaQuotes Software Corp. |
//|                                     http://www.mql5.com |
//+-----+

#property copyright "Copyright 2011, MetaQuotes Software Corp."
#property link      "http://www.mql5.com"
#property version   "1.00"

#property description "Индикатор для демонстрации DRAW_SECTION"
#property description "Рисует прямыми отрезками через каждые bars баров"
#property description "Цвет, толщина и стиль секций меняется случайным образом"
#property description "через каждые N тиков"

#property indicator_chart_window
#property indicator_buffers 1
#property indicator_plots 1
//--- plot Section
#property indicator_label1 "Section"
#property indicator_type1 DRAW_SECTION
#property indicator_color1 clrRed
#property indicator_style1 STYLE_SOLID
```

```

#property indicator_width1 1
//--- input параметр
input int      bars=5;           // длина секций в барах
input int      N=5;              // кол-во тиков для изменения стиля секций
//--- индикаторный буфер для построения
double         SectionBuffer[];
//--- вспомогательная переменная для вычисления концов секций
int            divider;
//--- массив для хранения цветов
color colors[]={clrRed,clrBlue,clrGreen};
//--- массив для хранения стилей отрисовки линии
ENUM_LINE_STYLE styles[]={STYLE_SOLID,STYLE_DASH,STYLE_DOT,STYLE_DASHDOT,STYLE_DASHDO
//+-----+
//| Custom indicator initialization function |
//+-----+
int OnInit()
{
//--- связывание массива и индикаторного буфера
    SetIndexBuffer(0,SectionBuffer,INDICATOR_DATA);
//--- значение 0 (пустое значение) не будет участвовать в отрисовке
    PlotIndexSetDouble(0,PLOT_EMPTY_VALUE,0);
//--- проверим параметр индикатора
    if(bars<=0)
    {
        PrintFormat("Недопустимое значение параметра bar=%d",bars);
        return(-1);
    }
    else divider=2*bars;
//---+
    return(0);
}
//+-----+
//| Custom indicator iteration function |
//+-----+
int OnCalculate(const int rates_total,
                const int prev_calculated,
                const datetime &time[],
                const double &open[],
                const double &high[],
                const double &low[],
                const double &close[],
                const long &tick_volume[],
                const long &volume[],
                const int &spread[])
{
    static int ticks=0;
//--- считаем тики для изменения стиля, цвета и толщины линии
    ticks++;

```

```

//--- если накопилось критическое число тиков
if(ticks>=N)
{
    //--- меняем свойства линии
    ChangeLineAppearance();
    //--- сбрасываем счетчик тиков в ноль
    ticks=0;
}

//--- номер бара, с которого начнем расчет значений индикатора
int start=0;
//--- если индикатор уже рассчитывали раньше, то установим start на предыдущий бар
if(prev_calculated>0) start=prev_calculated-1;
//--- здесь все расчеты значений индикатора
for(int i=start;i<rates_total;i++)
{
    //--- получим остаток от деления номера бара на 2*bars
    int rest=i%divider;
    //--- если номер бара делится без остатка на 2*bars
    if(rest==0)
    {
        //--- конец отрезка установим на цену High этого бара
        SectionBuffer[i]=high[i];
    }
    //--- если остаток от деления равен bars,
    else
    {
        //--- конец отрезка установим на цену High этого бара
        if(rest==bars) SectionBuffer[i]=low[i];
        //--- если ничего не подошло, то этот бар пропускаем - ставим значение 0
        else SectionBuffer[i]=0;
    }
}

//--- вернем значение prev_calculated для следующего вызова функции
return(rates_total);
}

//+-----+
//| изменяет внешний вид секций в индикаторе |
//+-----+
void ChangeLineAppearance()
{
    //--- строка для формирования информации о свойствах линии
    string comm="";
    //--- блок изменения цвета линии
    int number=MathRand(); // получим случайное число
    //--- делитель числа равен размеру массива colors[]
    int size=ArraySize(colors);
    //--- получим индекс для выбора нового цвета как остаток от целочисленного деления

```

```
int color_index=number%size;
//--- установим цвет как свойство PLOT_LINE_COLOR
PlotIndexSetInteger(0,PLOT_LINE_COLOR,colors[color_index]);
//--- запишем цвет линии
comm=comm+"\r\n"+(string)colors[color_index];

//--- блок изменения толщины линии
number=MathRand();
//--- получим толщину как остаток от целочисленного деления
int width=number%5; // толщина задается от 0 до 4
//--- установим толщину
PlotIndexSetInteger(0,PLOT_LINE_WIDTH,width);
//--- запишем толщину линии
comm=comm+"\r\nWidth="+IntegerToString(width);

//--- блок изменения стиля линии
number=MathRand();
//--- делитель числа равен размеру массива styles
size=ArraySize(styles);
//--- получим индекс для выбора нового стиля как остаток от целочисленного деления
int style_index=number%size;
//--- установим стиль линий
PlotIndexSetInteger(0,PLOT_LINE_STYLE,styles[style_index]);
//--- запишем стиль линии
comm="\r\n"+EnumToString(styles[style_index])+" "+comm;
//--- выведем информацию на график через комментарий
Comment(comm);
}
```

DRAW_HISTOGRAM

DRAW_HISTOGRAM

DRAW_LINE - _____
PlotIndexSetInteger().

DRAW_HISTOGRAM

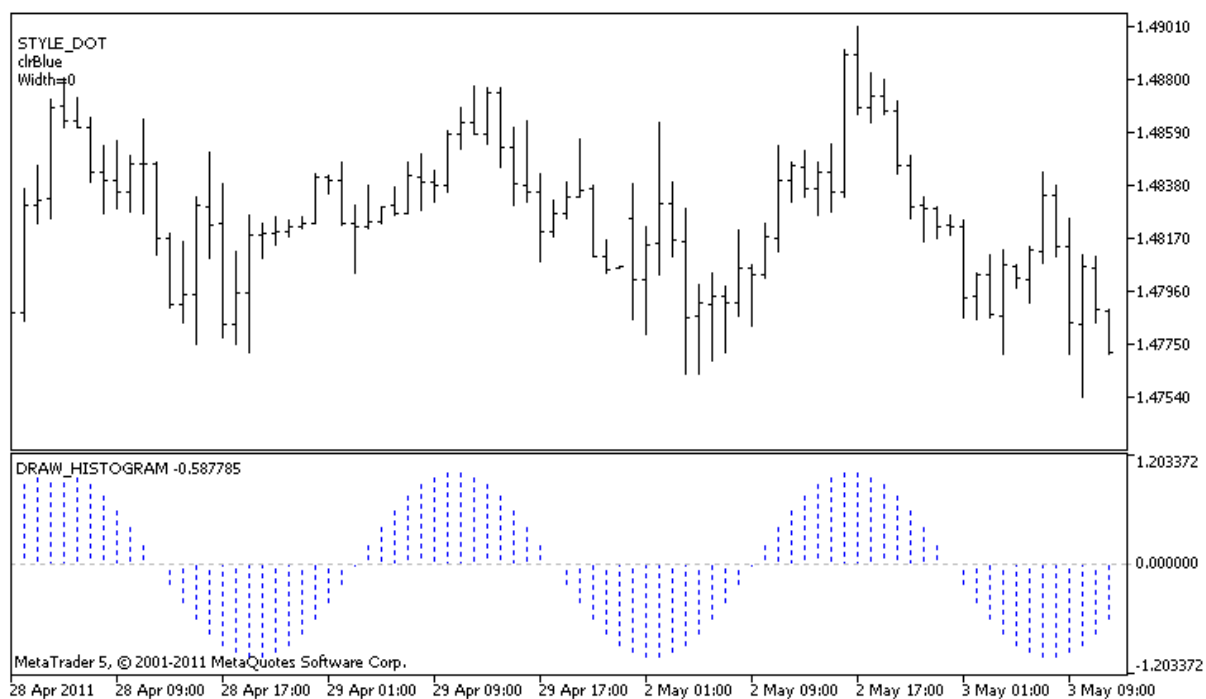
PowerOsMA.Bears

DRAW_HISTOGRAM - 1.

MathSin().

N

bars



```

plot1                                DRAW_HISTOGRAM
                                     #property,
OnCalculate( )
                                     N
                                     (
                                     "
                                     "
                                     ) .

```

```

//+-----+
//|                                     DRAW_HISTOGRAM.mq5 |
//|                                     Copyright 2011, MetaQuotes Software Corp. |
//|                                     http://www.mql5.com |
//+-----+
#property copyright "Copyright 2011, MetaQuotes Software Corp."
#property link      "http://www.mql5.com"
#property version   "1.00"

#property description "Индикатор для демонстрации DRAW_HISTOGRAM"
#property description "Рисует синусоиду гистограммой в отдельном окне"
#property description "Цвет и толщина столбиков меняется случайным образом"
#property description "через каждые N тиков"
#property description "Параметр bars задает количество баров в цикле синусоиды"

#property indicator_separate_window
#property indicator_buffers 1
#property indicator_plots   1
//--- plot Histogram
#property indicator_label1  "Histogram"
#property indicator_type1   DRAW_HISTOGRAM
#property indicator_color1  clrBlue
#property indicator_style1  STYLE_SOLID
#property indicator_width1  1
//--- input параметры
input int      bars=30;          // период синусоиды в барах
input int      N=5;              // кол-во тиков для изменения гистограммы
//--- indicator buffers
double         HistogramBuffer[];
//--- множитель для получения угла 2Pi в радианах при умножении на параметр bars
double         multiplier;
//--- массив для хранения цветов
color colors[]={clrRed,clrBlue,clrGreen};
//--- массив для хранения стилей отрисовки линии
ENUM_LINE_STYLE styles[]={STYLE_SOLID,STYLE_DASH,STYLE_DOT,STYLE_DASHDOT,STYLE_DASHDO
//+-----+
//| Custom indicator initialization function |
//+-----+
int OnInit()
{
//--- indicator buffers mapping
SetIndexBuffer(0,HistogramBuffer,INDICATOR_DATA);

```

```

//--- ВЫЧИСЛИМ МНОЖИТЕЛЬ
if(bars>1)multiplier=2.*M_PI/bars;
else
{
    PrintFormat("Задайте значение bars=%d больше 1",bars);
    //--- досрочное прекращение работы индикатора
    return(-1);
}
//---
return(0);
}
//+-----+
//| Custom indicator iteration function |
//+-----+
int OnCalculate(const int rates_total,
               const int prev_calculated,
               const datetime &time[],
               const double &open[],
               const double &high[],
               const double &low[],
               const double &close[],
               const long &tick_volume[],
               const long &volume[],
               const int &spread[])
{
    static int ticks=0;
    //--- считаем тики для изменения стиля, цвета и толщины линии
    ticks++;
    //--- если накопилось критическое число тиков
    if(ticks>=N)
    {
        //--- меняем свойства линии
        ChangeLineAppearance();
        //--- сбрасываем счетчик тиков в ноль
        ticks=0;
    }

    //--- вычисления значений индикатора
    int start=0;
    //--- если расчет уже производился на предыдущем запуске OnCalculate
    if(prev_calculated>0) start=prev_calculated-1; // установим начало расчетов с пред
    //--- заполняем индикаторный буфер значениями
    for(int i=start;i<rates_total;i++)
    {
        HistogramBuffer[i]=sin(i*multiplier);
    }
    //--- вернем значение prev_calculated для следующего вызова функции
    return(rates_total);
}

```

```

    }
//+-----+
//|  изменяет внешний вид линий в индикаторе |
//+-----+
void ChangeLineAppearance()
{
//--- строка для формирования информации о свойствах линии
    string comm="";
//--- блок изменения цвета линии
    int number=MathRand(); // получим случайное число
//--- делитель числа равен размеру массива colors[]
    int size=ArraySize(colors);
//--- получим индекс для выбора нового цвета как остаток от целочисленного деления
    int color_index=number%size;
//--- установим цвет как свойство PLOT_LINE_COLOR
    PlotIndexSetInteger(0,PLOT_LINE_COLOR,colors[color_index]);
//--- запишем цвет линии
    comm=comm+"\r\n"+(string)colors[color_index];

//--- блок изменения толщины линии
    number=MathRand();
//--- получим толщину как остаток от целочисленного деления
    int width=number%5; // толщина задается от 0 до 4
//--- установим толщину
    PlotIndexSetInteger(0,PLOT_LINE_WIDTH,width);
//--- запишем толщину линии
    comm=comm+"\r\nWidth="+IntegerToString(width);

//--- блок изменения стиля линии
    number=MathRand();
//--- делитель числа равен размеру массива styles
    size=ArraySize(styles);
//--- получим индекс для выбора нового стиля как остаток от целочисленного деления
    int style_index=number%size;
//--- установим стиль линий
    PlotIndexSetInteger(0,PLOT_LINE_STYLE,styles[style_index]);
//--- запишем стиль линии
    comm="\r\n"+EnumToString(styles[style_index])+""+comm;
//--- выведем информацию на график через комментарий
    Comment(comm);
}

```


DRAW__HISTOGRAM2

DRAW__HISTOGRAM2

DRAW__LINE -PlotIndexSetInteger().

DRAW__HISTOGRAM

DRAW__HISTOGRAM2 - 2.

Open Close.

N

OnInit()

- invisible_day.

PLOT_EMPTY_VALUE=0:

//--- установим пустое значение

PlotIndexSetDouble (индекс_построения_DRAW_SECTION, PLOT_EMPTY_VALUE, 0);



```

plot1
DRAW_HISTOGRAM2
#property,
OnCalculate( )
N
(
)

```

```
//+-----+
//|                                     DRAW_HISTOGRAM2.mq5 |
//|                                     Copyright 2011, MetaQuotes Software Corp. |
//|                                     http://www.mql5.com |
//+-----+
#property copyright "Copyright 2011, MetaQuotes Software Corp."
#property link      "http://www.mql5.com"
#property version   "1.00"

#property description "Индикатор для демонстрации DRAW_HISTOGRAM2"
#property description "Рисует на каждом баре отрезок между Open и Close"
#property description "Цвет, толщина и стиль меняется случайным образом"
#property description "через каждые N тиков"

#property indicator_chart_window
#property indicator_buffers 2
#property indicator_plots 1
//--- plot Histogram_2
#property indicator_label1  "Histogram_2"
#property indicator_type1   DRAW_HISTOGRAM2
#property indicator_color1  clrRed
#property indicator_style1  STYLE SOLID
```

```

#property indicator_width1 1
//--- input parameters
input int      N=5;           // кол-во тиков для изменения гистограммы
//--- indicator buffers
double         Histogram_2Buffer1[];
double         Histogram_2Buffer2[];
//--- день недели, для которого индикатор не рисуется
int invisible_day;
//--- массив для хранения цветов
color colors[]={clrRed,clrBlue,clrGreen};
//--- массив для хранения стилей отрисовки линии
ENUM_LINE_STYLE styles[]={STYLE_SOLID,STYLE_DASH,STYLE_DOT,STYLE_DASHDOT,STYLE_DASHDO
//+-----+
//| Custom indicator initialization function |
//+-----+
int OnInit()
{
//--- indicator buffers mapping
    SetIndexBuffer(0,Histogram_2Buffer1,INDICATOR_DATA);
    SetIndexBuffer(1,Histogram_2Buffer2,INDICATOR_DATA);
//--- установим пустое значение
    PlotIndexSetDouble(0,PLOT_EMPTY_VALUE,0);
//--- получим случайное число от 0 до 5
    invisible_day=MathRand()%6;
//---
    return(0);
}
//+-----+
//| Custom indicator iteration function |
//+-----+
int OnCalculate(const int rates_total,
               const int prev_calculated,
               const datetime &time[],
               const double &open[],
               const double &high[],
               const double &low[],
               const double &close[],
               const long &tick_volume[],
               const long &volume[],
               const int &spread[])
{
    static int ticks=0;
//--- считаем тики для изменения стиля, цвета и толщины линии
    ticks++;
//--- если накопилось критическое число тиков
    if(ticks>=N)
    {
        //--- меняем свойства линии

```

```

        ChangeLineAppearance();
        //--- сбрасываем счетчик тиков в ноль
        ticks=0;
    }

//--- вычисления значений индикатора
    int start=0;
//--- для получения дня недели по времени открытия каждого бара
    MqlDateTime dt;
//--- если расчет уже производился на предыдущем запуске OnCalculate
    if(prev_calculated>0) start=prev_calculated-1; // установим начало расчетов с пред
//--- заполняем индикаторный буфер значениями
    for(int i=start;i<rates_total;i++)
    {
        TimeToStruct(time[i],dt);
        if(dt.day_of_week==invisible_day)
        {
            Histogram_2Buffer1[i]=0;
            Histogram_2Buffer2[i]=0;
        }
        else
        {
            Histogram_2Buffer1[i]=open[i];
            Histogram_2Buffer2[i]=close[i];
        }
    }

//--- вернем значение prev_calculated для следующего вызова функции
    return(rates_total);
}

//+-----+
//| изменяет внешний вид линий в индикаторе |
//+-----+
void ChangeLineAppearance()
{
    //--- строка для формирования информации о свойствах линии
    string comm="";
    //--- блок изменения цвета линии
    int number=MathRand(); // получим случайное число
    //--- делитель числа равен размеру массива colors[]
    int size=ArraySize(colors);
    //--- получим индекс для выбора нового цвета как остаток от целочисленного деления
    int color_index=number%size;
    //--- установим цвет как свойство PLOT_LINE_COLOR
    PlotIndexSetInteger(0,PLOT_LINE_COLOR,colors[color_index]);
    //--- запишем цвет линии
    comm=comm+"\r\n"+(string)colors[color_index];

    //--- блок изменения толщины линии

```

```

    number=MathRand();
//--- получим толщину как остаток от целочисленного деления
    int width=number%5;    // толщина задается от 0 до 4
//--- установим толщину линий
    PlotIndexSetInteger(0,PLOT_LINE_WIDTH,width);
//--- запишем толщину линии
    comm=comm+"\r\nWidth="+IntegerToString(width);

//--- блок изменения стиля линии
    number=MathRand();
//--- делитель числа равен размеру массива styles
    size=ArraySize(styles);
//--- получим индекс для выбора нового стиля как остаток от целочисленного деления
    int style_index=number%size;
//--- установим стиль линий
    PlotIndexSetInteger(0,PLOT_LINE_STYLE,styles[style_index]);
//--- запишем стиль линии
    comm="\r\n"+EnumToString(styles[style_index])+" "+comm;
//--- добавим информацию о дне, который пропускается в расчетах
    comm="\r\nНеотрисовываемый день - "+EnumToString((ENUM_DAY_OF_WEEK)invisible_day)+
//--- выведем информацию на график через комментарий
    Comment(comm);
}

```

DRAW_ARROW

DRAW_ARROW

(

[Wingdings](#))[DRAW_LINE](#) -[PlotIndexSetInteger\(\)](#).[PLOT_ARROW](#).

```
//--- зададим код символа из шрифта Wingdings для отрисовки в PLOT_ARROW
PlotIndexSetInteger(0,PLOT_ARROW,code);
```

PLOT_ARROW=159 () .

) .

```
//--- зададим смещение стрелок по вертикали в пикселях
PlotIndexSetInteger(0,PLOT_ARROW_SHIFT,shift);
```

PLOT_ARROW_SHIFT

DRAW_ARROW

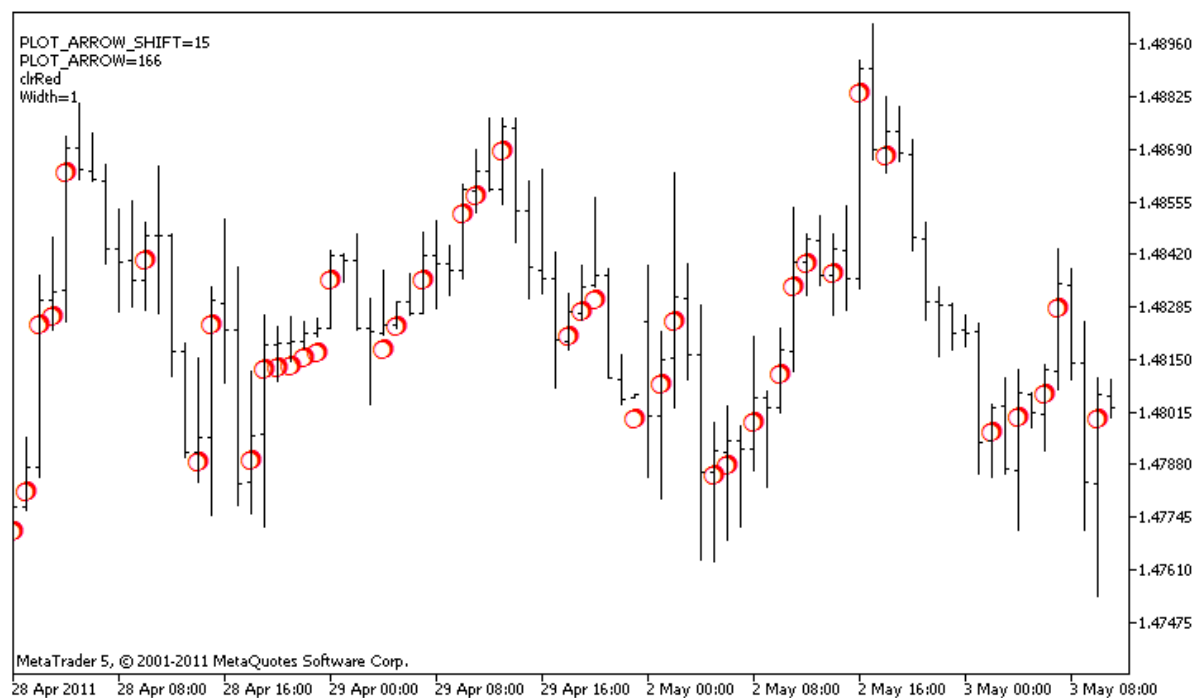
```
//--- установим пустое значение
PlotIndexSetDouble(индекс_построения_DRAW_ARROW,PLOT_EMPTY_VALUE,0);
```

DRAW_ARROW -

1.

Close

N



plot1

DRAW_ARROW

#property,

OnCalculate()

N

).

```
//+-----+
//|                                     DRAW_ARROW.mq5 |
//|                                     Copyright 2011, MetaQuotes Software Corp. |
//|                                     http://www.mql5.com |
//+-----+

#property copyright "Copyright 2011, MetaQuotes Software Corp."
#property link      "http://www.mql5.com"
#property version   "1.00"

#property description "Индикатор для демонстрации DRAW_ARROW"
#property description "Рисует на графике стрелки, задаваемые символами Unicode"
#property description "Цвет, размер, смещение и код символа стрелки меняется случайно"
#property description "через каждые N тиков"
#property description "Параметр code задает базовое значение: код=159 (кружок)"

#property indicator_chart_window
#property indicator_buffers 1
#property indicator_plots 1
//--- plot Arrows
#property indicator_label1 "Arrows"
#property indicator_type1  DRAW_ARROW
#property indicator_color1  clrGreen
```

```

#property indicator_width1 1
//--- input параметры
input int      N=5;           // кол-во тиков для изменения
input ushort   code=159;      // код символа для отрисовки в DRAW_ARROW
//--- индикаторный буфер для построения
double         ArrowsBuffer[];
//--- массив для хранения цветов
color colors[]={clrRed,clrBlue,clrGreen};
//+-----+
//| Custom indicator initialization function |
//+-----+
int OnInit()
{
//--- indicator buffers mapping
    SetIndexBuffer(0,ArrowsBuffer,INDICATOR_DATA);
//--- зададим код символа для отрисовки в PLOT_ARROW
    PlotIndexSetInteger(0,PLOT_ARROW,code);
//--- зададим смещение стрелок по вертикали в пикселях
    PlotIndexSetInteger(0,PLOT_ARROW_SHIFT,5);
//--- установим в качестве пустого значения 0
    PlotIndexSetDouble(0,PLOT_EMPTY_VALUE,0);
//---
    return(0);
}
//+-----+
//| Custom indicator iteration function |
//+-----+
int OnCalculate(const int rates_total,
                const int prev_calculated,
                const datetime &time[],
                const double &open[],
                const double &high[],
                const double &low[],
                const double &close[],
                const long &tick_volume[],
                const long &volume[],
                const int &spread[])
{
    static int ticks=0;
//--- считаем тики для изменения цвета, размера, смещения и кода стрелки
    ticks++;
//--- если накопилось критическое число тиков
    if(ticks>=N)
    {
        //--- меняем свойства линии
        ChangeLineAppearance();
        //--- сбрасываем счетчик тиков в ноль
        ticks=0;
    }
}

```



```

    }

//--- блок расчета значений индикатора
    int start=1;
    if(prev_calculated>0) start=prev_calculated-1;
//--- цикл расчета
    for(int i=1;i<rates_total;i++)
    {
        //--- если текущая цена Close больше предыдущей, ставим стрелку
        if(close[i]>close[i-1])
            ArrowsBuffer[i]=close[i];
        //--- в противном случае указываем нулевое значение
        else
            ArrowsBuffer[i]=0;
    }

//--- return value of prev_calculated for next call
    return(rates_total);
}

//+-----+
//| изменяет внешний вид символов в индикаторе |
//+-----+
void ChangeLineAppearance()
{
    //--- строка для формирования информации о свойствах индикатора
    string comm="";
    //--- блок изменения цвета стрелки
    int number=MathRand(); // получим случайное число
    //--- делитель числа равен размеру массива colors[]
    int size=ArraySize(colors);
    //--- получим индекс для выбора нового цвета как остаток от целочисленного деления
    int color_index=number%size;
    //--- установим цвет как свойство PLOT_LINE_COLOR
    PlotIndexSetInteger(0,PLOT_LINE_COLOR,colors[color_index]);
    //--- запишем цвет линии
    comm=comm+"\r\n"+(string)colors[color_index];

    //--- блок изменения размера стрелок
    number=MathRand();
    //--- получим толщину как остаток от целочисленного деления
    int width=number%5; // размер задается от 0 до 4
    //--- установим цвет как свойство PLOT_LINE_WIDTH
    PlotIndexSetInteger(0,PLOT_LINE_WIDTH,width);
    //--- запишем размер стрелок
    comm=comm+"\r\nWidth="+IntegerToString(width);

    //--- блок изменения кода стрелки (PLOT_ARROW)
    number=MathRand();
    //--- получим остаток от целочисленного деления для вычисления нового кода стрелки (о

```

```
int code_add=number%20;
//--- установим новый код символа как сумму code+code_add
PlotIndexSetInteger(0,PLOT_ARROW,code+code_add);
//--- запишем код символа PLOT_ARROW
comm="\r\n"+"PLOT_ARROW="+IntegerToString(code+code_add)+comm;

//--- блок изменения смещения стрелок по вертикали в пикселях
number=MathRand();
//--- получим смещение как остаток от целочисленного деления
int shift=20-number%41;
//--- установим новое смещение от -20 до 20
PlotIndexSetInteger(0,PLOT_ARROW_SHIFT,shift);
//--- запишем смещение PLOT_ARROW_SHIFT
comm="\r\n"+"PLOT_ARROW_SHIFT="+IntegerToString(shift)+comm;

//--- выведем информацию на график через комментарий
Comment(comm);
}
```

DRAW_ZIGZAG

DRAW_ZIGZAG

DRAW_SECTION,DRAW_SECTION - _____PlotIndexSetInteger().

" "

PLOT_EMPTY_VALUE:

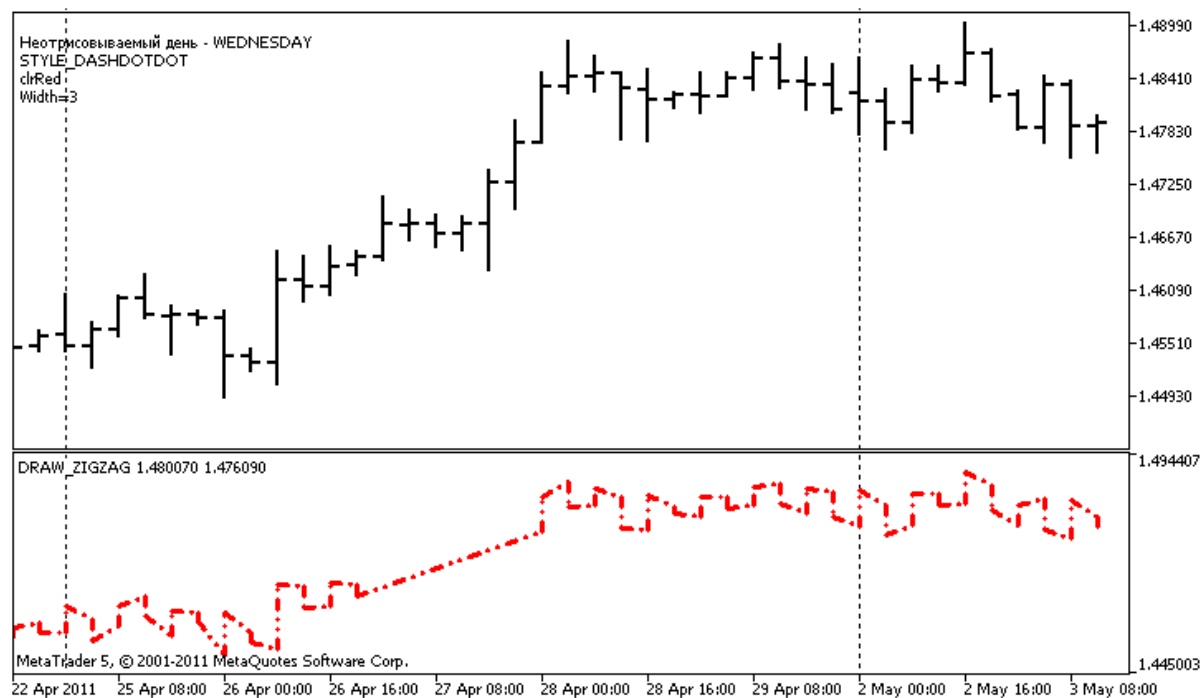
```
//--- значение 0 (пустое значение) не будет участвовать в отрисовке
PlotIndexSetDouble(индекс_построения_DRAW_ZIGZAG,PLOT_EMPTY_VALUE,0);
```

DRAW_ZIGZAG -

2.

High Low.

N



plot 1

DRAW_ZIGZAG

#property,

OnCalculate()

N

) .

```

-----+
//|
//|                                     DRAW_ZIGZAG.mq5 |
//|                                     Copyright 2011, MetaQuotes Software Corp. |
//|                                     http://www.mql5.com |
//+-----+

#property copyright "Copyright 2011, MetaQuotes Software Corp."
#property link      "http://www.mql5.com"
#property version   "1.00"

#property description "Индикатор для демонстрации DRAW_ZIGZAG"
#property description "Рисует прямыми отрезками \"пилу\", пропуская бары одного дня"
#property description "День пропусков выбирается случайным образом при запуске индикатора"
#property description "Цвет, толщина и стиль отрезков меняется случайным"
#property description "образом через каждые N тиков"

#property indicator_separate_window
#property indicator_buffers 2
#property indicator_plots 1
//--- plot ZigZag
#property indicator_label1 "ZigZag"
#property indicator_type1 DRAW_ZIGZAG
#property indicator_color1 clrBlue

```

```

#property indicator_style1 STYLE_SOLID
#property indicator_width1 1
//--- input параметры
input int      N=5;                // кол-во тиков для изменения
//--- indicator buffers
double         ZigZagBuffer1[];
double         ZigZagBuffer2[];
//--- день недели, для которого индикатор не рисуется
int invisible_day;
//--- массив для хранения цветов
color colors[]={clrRed,clrBlue,clrGreen};
//--- массив для хранения стилей отрисовки линии
ENUM_LINE_STYLE styles[]={STYLE_SOLID,STYLE_DASH,STYLE_DOT,STYLE_DASHDOT,STYLE_DASHDO
//+-----+
//| Custom indicator initialization function |
//+-----+
int OnInit()
{
//--- связывание массивов и индикаторных буферов
    SetIndexBuffer(0,ZigZagBuffer1,INDICATOR_DATA);
    SetIndexBuffer(1,ZigZagBuffer2,INDICATOR_DATA);
//--- получим случайное число от 0 до 6, для этого дня индикатор не рисуется
    invisible_day=MathRand()%6;
//--- значение 0 (пустое значение) не будет участвовать в отрисовке
    PlotIndexSetDouble(0,PLOT_EMPTY_VALUE,0);
//--- значение 0 (пустое значение) не будет участвовать в отрисовке
    PlotIndexSetString(0,PLOT_LABEL,"ZigZag1;ZigZag2");
//---
    return(0);
}
//+-----+
//| Custom indicator iteration function |
//+-----+
int OnCalculate(const int rates_total,
               const int prev_calculated,
               const datetime &time[],
               const double &open[],
               const double &high[],
               const double &low[],
               const double &close[],
               const long &tick_volume[],
               const long &volume[],
               const int &spread[])
{
    static int ticks=0;
//--- считаем тики для изменения стиля, цвета и толщины линии
    ticks++;
//--- если накопилось достаточное число тиков

```

```

    if(ticks>=N)
    {
        //--- меняем свойства линии
        ChangeLineAppearance();
        //--- сбрасываем счетчик тиков в ноль
        ticks=0;
    }

    //--- структура времени понадобится для получения дня недели каждого бара
    MqlDateTime dt;

    //--- позиция начала расчетов
    int start=0;
    //--- если индикатор рассчитывался на предыдущем тике, то начинаю расчет с предпослед
    if(prev_calculated!=0) start=prev_calculated-1;
    //--- цикл расчетов
    for(int i=start;i<rates_total;i++)
    {
        //--- запишем время открытия бара в структуру
        TimeToStruct(time[i],dt);
        //--- если день недели этого бара равен invisible_day
        if(dt.day_of_week==invisible_day)
        {
            //--- запишем пустые значения в буфера для этого бара
            ZigZagBuffer1[i]=0;
            ZigZagBuffer2[i]=0;
        }
        //--- если день недели подходящий, то заполняем буфера
        else
        {
            //--- если номер бара четный
            if(i%2==0)
            {
                //--- пишем в 1-ый буфер High, во 2-ой Low
                ZigZagBuffer1[i]=high[i];
                ZigZagBuffer2[i]=low[i];
            }
            //--- номер бара нечетный
            else
            {
                //--- заполняем бар в обратном порядке
                ZigZagBuffer1[i]=low[i];
                ZigZagBuffer2[i]=high[i];
            }
        }
    }

    //--- return value of prev_calculated for next call
    return(rates_total);

```

```

    }
//+-----+
//| изменяет внешний вид отрезков в зигзаге |
//+-----+
void ChangeLineAppearance()
{
//--- строка для формирования информации о свойствах ZigZag
    string comm="";
//--- блок изменения цвета зигзага
    int number=MathRand(); // получим случайное число
//--- делитель числа равен размеру массива colors[]
    int size=ArraySize(colors);
//--- получим индекс для выбора нового цвета как остаток от целочисленного деления
    int color_index=number%size;
//--- установим цвет как свойство PLOT_LINE_COLOR
    PlotIndexSetInteger(0,PLOT_LINE_COLOR,colors[color_index]);
//--- запишем цвет линии
    comm=comm+"\r\n"+(string)colors[color_index];

//--- блок изменения толщины линии
    number=MathRand();
//--- получим толщину как остаток от целочисленного деления
    int width=number%5; // толщина задается от 0 до 4
//--- установим цвет как свойство PLOT_LINE_WIDTH
    PlotIndexSetInteger(0,PLOT_LINE_WIDTH,width);
//--- запишем толщину линии
    comm=comm+"\r\nWidth="+IntegerToString(width);

//--- блок изменения стиля линии
    number=MathRand();
//--- делитель числа равен размеру массива styles
    size=ArraySize(styles);
//--- получим индекс для выбора нового стиля как остаток от целочисленного деления
    int style_index=number%size;
//--- установим цвет как свойство PLOT_LINE_COLOR
    PlotIndexSetInteger(0,PLOT_LINE_STYLE,styles[style_index]);
//--- запишем стиль линии
    comm="\r\n"+EnumToString(styles[style_index])+""+comm;
//--- добавим информацию о дне, который пропускается в расчетах
    comm="\r\nНеотрисовываемый день - "+EnumToString((ENUM_DAY_OF_WEEK)invisible_day)+
//--- выведем информацию на график через комментарий
    Comment(comm);
}

```

DRAW_FILLING

DRAW_FILLING

•

•

PlotIndexSetInteger().PLOT_EMPTY_VALUE:

```
//--- значение 0 (пустое значение) не будет участвовать в отрисовке
PlotIndexSetDouble(индекс_построения_DRAW_FILLING,PLOT_EMPTY_VALUE,0);
```

DRAW_FILLING -

2.

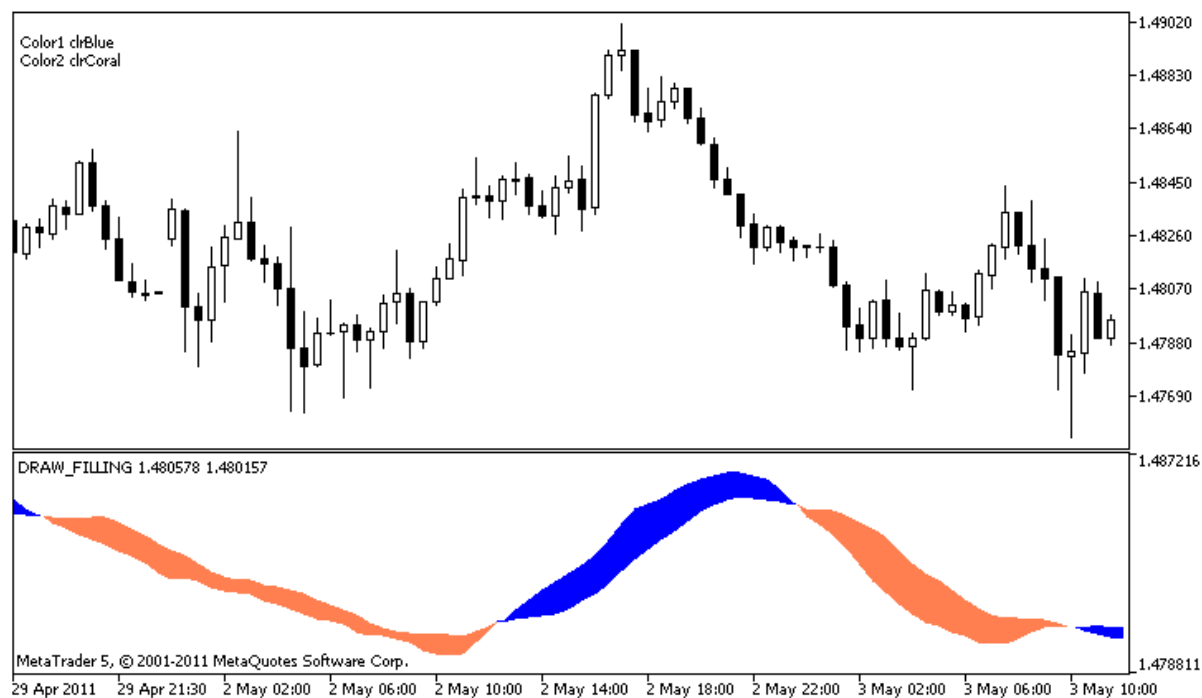
•

•

N

N

) .



plot1

DRAW_FILLING

#property,

OnCalculate()

```
//+-----+
//|                                     DRAW_FILLING.mq5 |
//|                                     Copyright 2011, MetaQuotes Software Corp. |
//|                                     http://www.mql5.com |
//+-----+

#property copyright "Copyright 2011, MetaQuotes Software Corp."
#property link      "http://www.mql5.com"
#property version   "1.00"

#property description "Индикатор для демонстрации DRAW_FILLING"
#property description "Рисует в отдельном окне канал между двумя средними"
#property description "Цвет заливки канала меняется случайным образом"
#property description "через каждые N тиков"

#property indicator_separate_window
#property indicator_buffers 2
#property indicator_plots 1
//--- plot Intersection
#property indicator_label1 "Intersection"
#property indicator_type1  DRAW_FILLING
#property indicator_color1  clrRed,clrBlue
#property indicator_width1  1
//--- input параметры
input int      Fast=13;          // период быстрой скользящей средней
```

```

input int      Slow=21;           // период медленной скользящей средней
input int      shift=1;           // сдвиг средних в будущее (положительный)
input int      N=5;               // кол-во тиков для изменения
//--- индикаторные буфера
double         IntersectionBuffer1[];
double         IntersectionBuffer2[];
int fast_handle;
int slow_handle;
//--- массив для хранения цветов
color colors[]={clrRed,clrBlue,clrGreen,clrAquamarine,clrBlanchedAlmond,clrBrown,clrC
//+-----+
//| Custom indicator initialization function |
//+-----+
int OnInit()
{
//--- indicator buffers mapping
    SetIndexBuffer(0,IntersectionBuffer1,INDICATOR_DATA);
    SetIndexBuffer(1,IntersectionBuffer2,INDICATOR_DATA);
//---
    PlotIndexSetInteger(0,PLOT_SHIFT,shift);
//---
    fast_handle=iMA(_Symbol,_Period,Fast,0,MODE_SMA,PRICE_CLOSE);
    slow_handle=iMA(_Symbol,_Period,Slow,0,MODE_SMA,PRICE_CLOSE);
//---
    return(0);
}
//+-----+
//| Custom indicator iteration function |
//+-----+
int OnCalculate(const int rates_total,
                const int prev_calculated,
                const datetime &time[],
                const double &open[],
                const double &high[],
                const double &low[],
                const double &close[],
                const long &tick_volume[],
                const long &volume[],
                const int &spread[])
{
    static int ticks=0;
//--- считаем тики для изменения стиля, цвета и толщины линии
    ticks++;
//--- если накопилось достаточное число тиков
    if(ticks>=N)
    {
        //--- меняем свойства линии
        ChangeLineAppearance();
    }
}

```

```

    //--- сбрасываем счетчик тиков в ноль
    ticks=0;
}

//--- делаем первый расчет индикатора или данные изменились и требуется полный перер
if(prev_calculated==0)
{
    //--- копируем все значения индикаторов в соответствующие буферы
    int copied1=CopyBuffer(fast_handle,0,0,rates_total,IntersectionBuffer1);
    int copied2=CopyBuffer(slow_handle,0,0,rates_total,IntersectionBuffer2);
}
else // экономно заполняем только те данные, которые обновились
{
    //--- получим разницу в барах между текущим и предыдущим запуском OnCalculate()
    int to_copy=rates_total-prev_calculated;
    //--- если разницы нет, то все равно будем копировать одно значение - на нулевое
    if(to_copy==0) to_copy=1;
    //--- копируем to_copy значений в самый конец индикаторных буферов
    int copied1=CopyBuffer(fast_handle,0,0,to_copy,IntersectionBuffer1);
    int copied2=CopyBuffer(slow_handle,0,0,to_copy,IntersectionBuffer2);
}

//--- return value of prev_calculated for next call
return(rates_total);
}

//+-----+
//| изменяет цвета заливки канала |
//+-----+
void ChangeLineAppearance()
{
    //--- строка для формирования информации о свойствах линии
    string comm="";
    //--- блок изменения цвета линии
    int number=MathRand(); // получим случайное число
    //--- делитель числа равен размеру массива colors[]
    int size=ArraySize(colors);

    //--- получим индекс для выбора нового цвета как остаток от целочисленного деления
    int color_index1=number%size;
    //--- установим первый цвет как свойство PLOT_LINE_COLOR
    PlotIndexSetInteger(0,PLOT_LINE_COLOR,0,colors[color_index1]);
    //--- запишем первый цвет
    comm=comm+"\r\nColor1 "+(string)colors[color_index1];

    //--- получим индекс для выбора нового цвета как остаток от целочисленного деления
    number=MathRand(); // получим случайное число
    int color_index2=number%size;
    //--- установим второй цвет как свойство PLOT_LINE_COLOR
    PlotIndexSetInteger(0,PLOT_LINE_COLOR,1,colors[color_index2]);
}

```

```
//--- запишем второй цвет  
    comm=comm+"\r\nColor2 "+(string)colors[color_index2];  
//--- выведем информацию на график через комментарий  
    Comment(comm);  
}
```

DRAW_BARS

DRAW_BARS

Low Close.

Open, High,

PlotIndexSetInteger().

PLOT EMPTY VALUE:

```
//--- значение 0 (пустое значение) не будет участвовать в отрисовке
PlotIndexSetDouble(индекс построения DRAW BARS, PLOT_EMPTY_VALUE, 0);
```

DRAW_BARS -

4.

: Open, High, Low Close.

N . N

$$) \cdot \left(\frac{1}{\sqrt{\pi}} e^{-x^2} \right)$$



plot1

DRAW_BARS

[#property,](#)[OnCalculate\(\)](#)

```
//+-----+
//|                                     DRAW_BARS.mq5 |
//|                                     Copyright 2011, MetaQuotes Software Corp. |
//|                                     http://www.mql5.com |
//+-----+

#property copyright "Copyright 2011, MetaQuotes Software Corp."
#property link      "http://www.mql5.com"
#property version   "1.00"

#property description "Индикатор для демонстрации DRAW_BARS"
#property description "Рисует в отдельном окне бары по выбранному символу"
#property description "Цвет и толщина баров, а также символ, меняются случайным "
#property description "образом через каждые N тиков"

#property indicator_separate_window
#property indicator_buffers 4
#property indicator_plots 1
//--- plot Bars
#property indicator_label1 "Bars"
#property indicator_type1  DRAW_BARS
#property indicator_color1  clrGreen
#property indicator_style1  STYLE_SOLID
#property indicator_width1  1
//--- input параметры
```

```

input int      N=5;           // количество тиков для смены вида
input int      bars=500;      // сколько баров показывать
input bool     messages=false; // вывод сообщений в лог "Эксперты"
//--- индикаторные буфера
double        BarsBuffer1[];
double        BarsBuffer2[];
double        BarsBuffer3[];
double        BarsBuffer4[];
//--- имя символа
string symbol;
//--- массив для хранения цветов
color colors[]={clrRed,clrBlue,clrGreen,clrPurple,clrBrown,clrIndianRed};
//+-----+
//| Custom indicator initialization function |
//+-----+
int OnInit()
{
//--- если bars слишком мало - досрочно завершаем работу
    if(bars<50)
    {
        Comment("Укажите большее количество баров! Работа индикатора прекращена");
        return(-1);
    }
//--- indicator buffers mapping
    SetIndexBuffer(0,BarsBuffer1,INDICATOR_DATA);
    SetIndexBuffer(1,BarsBuffer2,INDICATOR_DATA);
    SetIndexBuffer(2,BarsBuffer3,INDICATOR_DATA);
    SetIndexBuffer(3,BarsBuffer4,INDICATOR_DATA);
//--- имя символа, по которому рисуются бары
    symbol=_Symbol;
//--- установим отображение символа
    PlotIndexSetString(0,PLOT_LABEL,symbol+" Open;"+symbol+" High;"+symbol+" Low;"+symbol+" Close");
    IndicatorSetString(INDICATOR_SHORTNAME,"DRAW_BARS("+symbol+")");
//--- пустое значение
    PlotIndexSetDouble(0,PLOT_EMPTY_VALUE,0.0);
//---
    return(0);
}
//+-----+
//| Custom indicator iteration function |
//+-----+
int OnCalculate(const int rates_total,
               const int prev_calculated,
               const datetime &time[],
               const double &open[],
               const double &high[],
               const double &low[],
               const double &close[],

```

```

        const long &tick_volume[],
        const long &volume[],
        const int &spread[])
{
    static int ticks=0;
    //--- считаем тики для изменения стиля, цвета и толщины линии
    ticks++;
    //--- если накопилось достаточное число тиков
    if(ticks>=N)
    {
        //--- выберем новый символ из окна "Обзор рынка"
        symbol=GetRandomSymbolName();
        //--- меняем свойства линии
        ChangeLineAppearance();

        int tries=0;
        //--- сделаем 5 попыток заполнить буфера ценами из symbol
        while(!CopyFromSymbolToBuffers(symbol,rates_total) && tries<5)
        {
            //--- счетчик вызовов функции CopyFromSymbolToBuffers()
            tries++;
        }
        //--- сбрасываем счетчик тиков в ноль
        ticks=0;
    }
    //--- return value of prev_calculated for next call
    return(rates_total);
}

//+-----+
//|   заполняем индикаторные буфера ценами   |
//+-----+
bool CopyFromSymbolToBuffers(string name,int total)
{
    //--- в массив rates[] будем копировать цены Open, High, Low и Close
    MqlRates rates[];
    //--- счетчик попыток
    int attempts=0;
    //--- сколько скопировано
    int copied=0;
    //--- делаем 25 попыток получить таймсерия по нужному символу
    while(attempts<25 && (copied=CopyRates(name,_Period,0,bars,rates))<0)
    {
        Sleep(100);
        attempts++;
        if(messages) PrintFormat("%s CopyRates(%s) attempts=%d",__FUNCTION__,name,attempts);
    }
    //--- если не удалось скопировать достаточное количество баров
    if(copied!=bars)

```



```

{
    //--- сформируем строку сообщения
    string comm=StringFormat("Для символа %s удалось получить только %d баров из %d",
                             name,
                             copied,
                             bars
                            );

    //--- выведем сообщение в комментарий на главное окно графика
    Comment(comm);
    //--- выводим сообщения
    if(messages) Print(comm);
    return(false);
}
else
{
    //--- установим отображение символа
    PlotIndexSetString(0,PLOT_LABEL,name+" Open;" +name+" High;" +name+" Low;" +name+"
    IndicatorSetString(INDICATOR_SHORTNAME,"DRAW_BARS (" +name+" )");
}
//--- инициализируем буфера пустыми значениями
ArrayInitialize(BarsBuffer1,0.0);
ArrayInitialize(BarsBuffer2,0.0);
ArrayInitialize(BarsBuffer3,0.0);
ArrayInitialize(BarsBuffer4,0.0);
//--- копируем цены в буферы
for(int i=0;i<copied;i++)
{
    //--- вычислим соответствующий индекс для буферов
    int buffer_index=total-copied+i;
    //--- записываем цены в буфера
    BarsBuffer1[buffer_index]=rates[i].open;
    BarsBuffer2[buffer_index]=rates[i].high;
    BarsBuffer3[buffer_index]=rates[i].low;
    BarsBuffer4[buffer_index]=rates[i].close;
}
return(true);
}
//+-----+
//| возвращает случайным образом символ из Market Watch |
//+-----+
string GetRandomSymbolName()
{
    //--- количество символов, показываемых в окне "Обзор рынка"
    int symbols=SymbolsTotal(true);
    //--- позиция символа в списке - случайное число от 0 до symbols
    int number=MathRand()%symbols;
    //--- вернем имя символа по указанной позиции
    return SymbolName(number,true);
}

```

```

    }
//+-----+
//| изменяет внешний вид баров |
//+-----+
void ChangeLineAppearance()
{
//--- строка для формирования информации о свойствах баров
    string comm="";
//--- блок изменения цвета баров
    int number=MathRand(); // получим случайное число
//--- делитель числа равен размеру массива colors[]
    int size=ArraySize(colors);
//--- получим индекс для выбора нового цвета как остаток от целочисленного деления
    int color_index=number%size;
//--- установим цвет как свойство PLOT_LINE_COLOR
    PlotIndexSetInteger(0,PLOT_LINE_COLOR,colors[color_index]);
//--- запишем цвет линии
    comm=comm+"\r\n"+(string)colors[color_index];

//--- блок изменения толщины баров
    number=MathRand();
//--- получим толщину как остаток от целочисленного деления
    int width=number%5; // толщина задается от 0 до 4
//--- установим цвет как свойство PLOT_LINE_WIDTH
    PlotIndexSetInteger(0,PLOT_LINE_WIDTH,width);
//--- запишем толщину линии
    comm=comm+"\r\nWidth="+IntegerToString(width);

//--- запишем имя символа
    comm="\r\n"+symbol+comm;

//--- выведем информацию на график через комментарий
    Comment(comm);
}

```

DRAW_CANDLES

DRAW_CANDLES

Open, High, Low Close.

PlotIndexSetInteger().

PLOT_EMPTY_VALUE:

```
//--- значение 0 (пустое значение) не будет участвовать в отрисовке
PlotIndexSetDouble(индекс_построения_DRAW_CANDLES,PLOT_EMPTY_VALUE,0);
```

DRAW_CANDLES

- 4.

: Open,

High, Low Close.

N

N

(" "



plot1

#property,

OnCalculate(_)

```
//+-----+
//|                                     DRAW_CANDLES.mq5 |
//|                                     Copyright 2011, MetaQuotes Software Corp. |
//|                                     http://www.mql5.com |
//+-----+

#property copyright "Copyright 2011, MetaQuotes Software Corp."
#property link      "http://www.mql5.com"
#property version   "1.00"

#property description "Индикатор для демонстрации DRAW_CANDLES."
#property description "Рисует в отдельном окне разным цветом свечи по случайно выбран"
#property description " "
#property description "Цвет и толщина свечей, а также символ, меняются"
#property description "случайным образом через каждые N тиков."

#property indicator_separate_window
#property indicator_buffers 4
#property indicator_plots 1
//--- plot Bars
#property indicator_label1 "DRAW_CANDLES1"
#property indicator_type1  DRAW_CANDLES
#property indicator_color1  clrGreen
#property indicator_style1  STYLE_SOLID
#property indicator_width1  1
```

```

//--- input параметры
input int      N=5;                // количество тиков для смены вида
input int      bars=500;           // сколько баров показывать
input bool     messages=false;     // вывод сообщений в лог "Эксперты"
//--- индикаторные буфера
double         Candle1Buffer1[];
double         Candle1Buffer2[];
double         Candle1Buffer3[];
double         Candle1Buffer4[];
//--- имя символа
string symbol;
//--- массив для хранения цветов
color colors[]={clrRed,clrBlue,clrGreen,clrPurple,clrBrown,clrIndianRed};
//+-----+
//| Custom indicator initialization function |
//+-----+
int OnInit()
{
//--- если bars слишком мало - досрочно завершаем работу
    if(bars<50)
    {
        Comment("Укажите большее количество баров! Работа индикатора прекращена");
        return(-1);
    }
//--- indicator buffers mapping
    SetIndexBuffer(0,Candle1Buffer1,INDICATOR_DATA);
    SetIndexBuffer(1,Candle1Buffer2,INDICATOR_DATA);
    SetIndexBuffer(2,Candle1Buffer3,INDICATOR_DATA);
    SetIndexBuffer(3,Candle1Buffer4,INDICATOR_DATA);
//--- пустое значение
    PlotIndexSetDouble(0,PLOT_EMPTY_VALUE,0);
//--- имя символа, по которому рисуются бары
    symbol=_Symbol;
//--- установим отображение символа
    PlotIndexSetString(0,PLOT_LABEL,symbol+" Open;"+symbol+" High;"+symbol+" Low;"+symbol+" Close;");
    IndicatorSetString(INDICATOR_SHORTNAME,"DRAW_CANDLES("+symbol+")");
//---
    return(0);
}
//+-----+
//| Custom indicator iteration function |
//+-----+
int OnCalculate(const int rates_total,
               const int prev_calculated,
               const datetime &time[],
               const double &open[],
               const double &high[],

```

```

        const double &low[],
        const double &close[],
        const long &tick_volume[],
        const long &volume[],
        const int &spread[])

{
    static int ticks=INT_MAX-100;
//--- считаем тики для изменения стиля, цвета и толщины линии
    ticks++;
//--- если накопилось достаточное число тиков
    if(ticks>=N)
    {
        //--- выберем новый символ из окна "Обзор рынка"
        symbol=GetRandomSymbolName();
        //--- сменим вид
        ChangeLineAppearance();
        //--- выберем новый символ из окна "Обзор рынка"
        int tries=0;
        //--- сделаем 5 попыток заполнить буфера plot1 ценами из symbol
        while(!CopyFromSymbolToBuffers(symbol,rates_total,0,
            Candle1Buffer1,Candle1Buffer2,Candle1Buffer3,Candle1Buffer4)
            && tries<5)
        {
            //--- счетчик вызовов функции CopyFromSymbolToBuffers()
            tries++;
        }
        //--- сбрасываем счетчик тиков в ноль
        ticks=0;
    }
//--- return value of prev_calculated for next call
    return(rates_total);
}

//+-----+
//|   заполняет указанную свечу   |
//+-----+
bool CopyFromSymbolToBuffers(string name,
                             int total,
                             int plot_index,
                             double &buff1[],
                             double &buff2[],
                             double &buff3[],
                             double &buff4[])
{
//--- в массив rates[] будем копировать цены Open, High, Low и Close
    MqlRates rates[];
//--- счетчик попыток
    int attempts=0;

```

```

//--- сколько скопировано
int copied=0;
//--- делаем 25 попыток получить таймсерию по нужному символу
while(attempts<25 && (copied=CopyRates(name,_Period,0,bars,rates)<0)
{
    Sleep(100);
    attempts++;
    if(messages) PrintFormat("%s CopyRates(%s) attempts=%d",__FUNCTION__,name,attempts);
}
//--- если не удалось скопировать достаточное количество баров
if(copied!=bars)
{
    //--- сформируем строку сообщения
    string comm=StringFormat("Для символа %s удалось получить только %d баров из %d",
        name,
        copied,
        bars
    );

    //--- выведем сообщение в комментарий на главное окно графика
    Comment(comm);
    //--- выводим сообщения
    if(messages) Print(comm);
    return(false);
}
else
{
    //--- установим отображение символа
    PlotIndexSetString(plot_index,PLOT_LABEL,name+" Open;" +name+" High;" +name+" Low");
}
//--- инициализируем буфера пустыми значениями
ArrayInitialize(buff1,0.0);
ArrayInitialize(buff2,0.0);
ArrayInitialize(buff3,0.0);
ArrayInitialize(buff4,0.0);
//--- на каждом тике копируем цены в буферы
for(int i=0;i<copied;i++)
{
    //--- вычислим соответствующий индекс для буферов
    int buffer_index=total-copied+i;
    //--- записываем цены в буфера
    buff1[buffer_index]=rates[i].open;
    buff2[buffer_index]=rates[i].high;
    buff3[buffer_index]=rates[i].low;
    buff4[buffer_index]=rates[i].close;
}
return(true);
}
//+-----+

```

```

//| возвращает случайным образом символ из Market Watch |
//+-----+
string GetRandomSymbolName()
{
//--- количество символов, показываемых в окне "Обзор рынка"
    int symbols=SymbolsTotal(true);
//--- позиция символа в списке - случайное число от 0 до symbols
    int number=MathRand()%symbols;
//--- вернем имя символа по указанной позиции
    return SymbolName(number,true);
}
//+-----+
//| изменяет внешний вид баров |
//+-----+
void ChangeLineAppearance()
{
//--- строка для формирования информации о свойствах баров
    string comm="";
//--- блок изменения цвета баров
    int number=MathRand(); // получим случайное число
//--- делитель числа равен размеру массива colors[]
    int size=ArraySize(colors);
//--- получим индекс для выбора нового цвета как остаток от целочисленного деления
    int color_index=number%size;
//--- установим цвет как свойство PLOT_LINE_COLOR
    PlotIndexSetInteger(0,PLOT_LINE_COLOR,colors[color_index]);
//--- запишем цвет
    comm=comm+"\r\n"+(string)colors[color_index];
//--- запишем имя символа
    comm="\r\n"+symbol+comm;
//--- выведем информацию на график через комментарий
    Comment(comm);
}

```


DRAW_COLOR_LINE

DRAW_COLOR_LINE
DRAW_LINE,

COLOR,

()

PlotIndexSetInteger().

DRAW_COLOR_LINE - 2:

•

;

•

#property

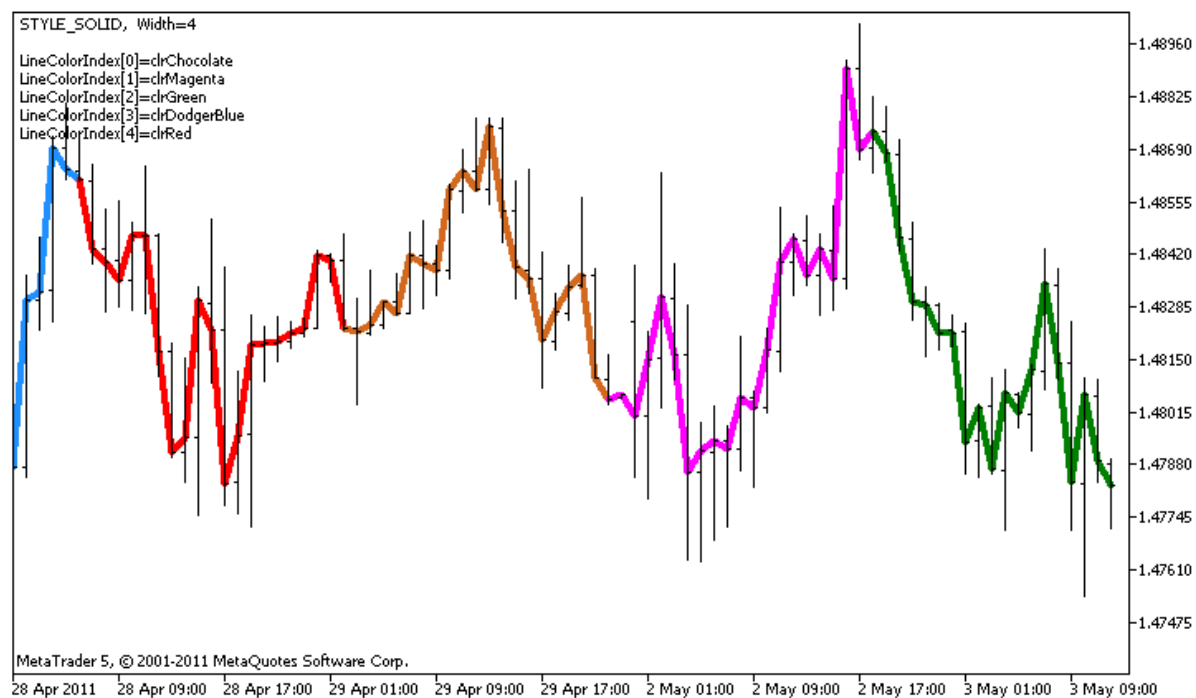
indicator_color1

64.

```
//--- зададим 5 цветов для раскраски каждого бара (они хранятся в специальном массиве
#property indicator_color1 clrRed,clrBlue,clrGreen,clrOrange,clrDeepPink // (можно у
```

Close.

N=5



ChangeColors

() .

```
//+-----+
//|  изменяет цвет участков линии |
//+-----+
void ChangeColors(color &cols[],int plot_colors)
{
//--- количество цветов
    int size=ArraySize(cols);
//---
    string comm=ChartGetString(0,CHART_COMMENT)+"\r\n\r\n";

//--- для каждого цветового индекса зададим новый цвет случайным образом
    for(int plot_color_ind=0;plot_color_ind<plot_colors;plot_color_ind++)
    {
        //--- получим случайное число
        int number=MathRand();
        //--- получим индекс в массиве col[] как остаток от целочисленного деления
        int i=number%size;
        //--- установим цвет для каждого индекса как свойство PLOT_LINE_COLOR
        PlotIndexSetInteger(0,                                // номер графического стиля
                           PLOT_LINE_COLOR,                 // идентификатор свойства
                           plot_color_ind,                   // индекс цвета, куда запишем цвет
                           cols[i]);                          // новый цвет

        //--- запишем цвета
        comm=comm+StringFormat("LineColorIndex[%d]=%s \r\n",plot_color_ind,ColorToString(cols[i]));
        ChartSetString(0,CHART_COMMENT,comm);
    }
}
```

```

    }
//---
}

```

```

"
"

```

```

ColorLineColors[] (

```

```

) .

```

[PlotIndexSetInteger\(\)](#).

```

plot1

```

```

DRAW_COLOR_LINE

```

```

#property,

```

[OnCalculate\(\)](#)

```

N Length (

```

```

)

```

```

) .

```

```

//+-----+
//|                                     DRAW_COLOR_LINE.mq5 |
//|                                     Copyright 2011, MetaQuotes Software Corp. |
//|                                     http://www.mql5.com |
//+-----+
#property copyright "Copyright 2011, MetaQuotes Software Corp."
#property link      "http://www.mql5.com"
#property version   "1.00"

#property description "Индикатор для демонстрации DRAW_COLOR_LINE"
#property description "Рисует цветными кусками по 20 баров линию по ценам Close"
#property description "Толщина, стиль и цвет участков линии меняется случайным"
#property description "образом через каждые N тиков"

#property indicator_chart_window
#property indicator_buffers 2
#property indicator_plots 1
//--- plot ColorLine
#property indicator_label1 "ColorLine"
#property indicator_type1 DRAW_COLOR_LINE
//--- зададим 5 цветов для раскраски каждого бара (они хранятся в специальном массиве
#property indicator_color1 clrRed,clrBlue,clrGreen,clrOrange,clrDeepPink // (можно у
#property indicator_style1 STYLE_SOLID
#property indicator_width1 1
//--- input параметры
input int      N=5;           // кол-во тиков для изменения
input int      Length=20;     // длина каждого участка цвета в барах

```

```

int          line_colors=5; // количество заданных цветов равно 5 - смотри выше #p
//--- буфер для отрисовки
double       ColorLineBuffer[];
//--- буфер для хранения цвета отрисовки линии на каждом баре
double       ColorLineColors[];

//--- массив для хранения цветов содержит 7 элементов
color colors[]={clrRed,clrBlue,clrGreen,clrChocolate,clrMagenta,clrDodgerBlue,clrGold}
//--- массив для хранения стилей отрисовки линии
ENUM_LINE_STYLE styles[]={STYLE_SOLID,STYLE_DASH,STYLE_DOT,STYLE_DASHDOT,STYLE_DASHDO

//+-----+
//| Custom indicator initialization function |
//+-----+
int OnInit()
{
//--- связывание массива и индикаторного буфера
    SetIndexBuffer(0,ColorLineBuffer,INDICATOR_DATA);
    SetIndexBuffer(1,ColorLineColors,INDICATOR_COLOR_INDEX);
//--- инициализация генератора псевдослучайных чисел
    MathSrand(GetTickCount());
//---
    return(0);
}
//+-----+
//| Custom indicator iteration function |
//+-----+
int OnCalculate(const int rates_total,
               const int prev_calculated,
               const datetime &time[],
               const double &open[],
               const double &high[],
               const double &low[],
               const double &close[],
               const long &tick_volume[],
               const long &volume[],
               const int &spread[])
{
    static int ticks=0;
//--- считаем тики для изменения стиля, цвета и толщины линии
    ticks++;
//--- если накопилось критическое число тиков
    if(ticks>=N)
    {
        //--- меняем свойства линии
        ChangeLineAppearance();
        //--- меняем цвета, которыми рисуются цветные участки линии
        ChangeColors(colors,5);
        //--- сбрасываем счетчик тиков в ноль
    }
}

```

```

        ticks=0;
    }

//--- блок расчета значений индикатора
    for(int i=0;i<rates_total;i++)
    {
        //--- запишем значение индикатора в буфер
        ColorLineBuffer[i]=close[i];
        //--- теперь случайным образом зададим для этого бара индекс цвета
        int color_index=i%(5*Length);
        color_index=color_index/Length;
        //--- для этого бара линия будет рисоваться цветом, который хранится под номером
        ColorLineColors[i]=color_index;
    }

//--- вернем значение prev_calculated для следующего вызова функции
    return(rates_total);
}

//+-----+
//|  изменяет цвет участков линии  |
//+-----+
void ChangeColors(color &cols[],int plot_colors)
{
    //--- количество цветов
    int size=ArraySize(cols);
    //---
    string comm=ChartGetString(0,CHART_COMMENT)+"\r\n\r\n";

    //--- для каждого цветового индекса зададим новый цвет случайным образом
    for(int plot_color_ind=0;plot_color_ind<plot_colors;plot_color_ind++)
    {
        //--- получим случайное число
        int number=MathRand();
        //--- получим индекс в массиве col[] как остаток от целочисленного деления
        int i=number%size;
        //--- установим цвет для каждого индекса как свойство PLOT_LINE_COLOR
        PlotIndexSetInteger(0, // номер графического стиля
                           PLOT_LINE_COLOR, // идентификатор свойства
                           plot_color_ind, // индекс цвета, куда запишем цвет
                           cols[i]); // новый цвет

        //--- запишем цвета
        comm=comm+StringFormat("LineColorIndex[%d]=%s \r\n",plot_color_ind,ColorToString(cols[i]));
        ChartSetString(0,CHART_COMMENT,comm);
    }
    //---
}

//+-----+
//|  изменяет внешний вид отображаемой линии в индикаторе  |

```

```
//+-----+
void ChangeLineAppearance()
{
//--- строка для формирования информации о свойствах линии
    string comm="";
//--- блок изменения толщины линии
    int number=MathRand();
//--- получим толщину как остаток от целочисленного деления
    int width=number%5; // толщина задается от 0 до 4
//--- установим цвет как свойство PLOT_LINE_WIDTH
    PlotIndexSetInteger(0,PLOT_LINE_WIDTH,width);
//--- запишем толщину линии
    comm=comm+" Width="+IntegerToString(width);

//--- блок изменения стиля линии
    number=MathRand();
//--- делитель числа равен размеру массива styles
    int size=ArraySize(styles);
//--- получим индекс для выбора нового стиля как остаток от целочисленного деления
    int style_index=number%size;
//--- установим цвет как свойство PLOT_LINE_COLOR
    PlotIndexSetInteger(0,PLOT_LINE_STYLE,styles[style_index]);
//--- запишем стиль линии
    comm=EnumToString(styles[style_index])+", "+comm;
//--- выведем информацию на график через комментарий
    Comment(comm);
}
```

DRAW__COLOR__SECTION

DRAW__COLOR__SECTION
DRAW_SECTION,

DRAW__COLOR__SECTION,
COLOR,

()

DRAW_SECTION - PlotIndexSetInteger() .

PLOT_EMPTY_VALUE.

```
//--- значение 0 (пустое значение) не будет участвовать в отрисовке
PlotIndexSetDouble(индекс_построения_DRAW_COLOR_SECTION,PLOT_EMPTY_VALUE,0);
```

DRAW__COLOR__SECTION - 2:

•

;

•

() .

#property

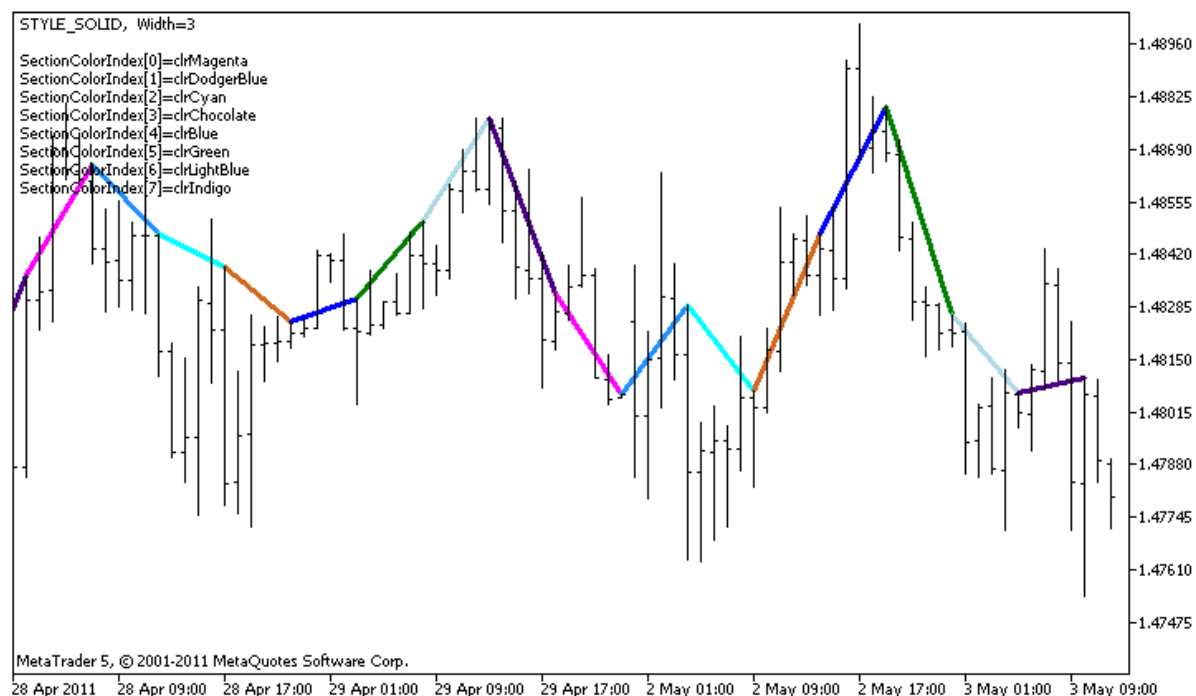
indicator__color1

64.

5

High.

N



```

plot1      DRAW_COLOR_SECTION
8
OnCalculate( )
colors[]
N
(
)

```

```

//+-----+
//|                                     DRAW_COLOR_SECTION.mq5 |
//|                                     Copyright 2011, MetaQuotes Software Corp. |
//|                                     http://www.mql5.com |
//+-----+

#property copyright "Copyright 2011, MetaQuotes Software Corp."
#property link      "http://www.mql5.com"
#property version   "1.00"

#property description "Индикатор для демонстрации DRAW_COLOR_SECTION"
#property description "Рисует цветными отрезками длиной в заданное число баров"
#property description "Цвет, толщина и стиль секций меняется случайным образом"
#property description "через каждые N тиков"

#property indicator_chart_window
#property indicator_buffers 2
#property indicator_plots 1
//--- plot ColorSection
#property indicator_label1 "ColorSection"
#property indicator_type1  DRAW_COLOR_SECTION

```



```
//--- зададим 8 цветов для раскраски секций (они хранятся в специальном массиве)
#property indicator_color1 clrRed,clrGold,clrMediumBlue,clrLime,clrMagenta,clrBrown,
#property indicator_style1 STYLE_SOLID
#property indicator_width1 1
//--- input параметры
input int N=5; // кол-во тиков для изменения
input int bars_in_section=5; // длина секций в барах
//--- вспомогательная переменная для вычисления концов секций
int divider;
int color_sections;
//--- буфер для отрисовки
double ColorSectionBuffer[];
//--- буфер для хранения цвета отрисовки линии на каждом баре
double ColorSectionColors[];
//--- массив для хранения цветов содержит 14 элементов
color colors[]=
{
    clrRed,clrBlue,clrGreen,clrChocolate,clrMagenta,clrDodgerBlue,clrGoldenrod,
    clrIndigo,clrLightBlue,clrAliceBlue,clrMoccasin,clrWhiteSmoke,clrCyan,clrMediumPurple;
};
//--- массив для хранения стилей отрисовки линии
ENUM_LINE_STYLE styles[]={STYLE_SOLID,STYLE_DASH,STYLE_DOT,STYLE_DASHDOT,STYLE_DASHDO
//+-----+
//| Custom indicator initialization function |
//+-----+
int OnInit()
{
    //--- indicator buffers mapping
    SetIndexBuffer(0,ColorSectionBuffer,INDICATOR_DATA);
    SetIndexBuffer(1,ColorSectionColors,INDICATOR_COLOR_INDEX);
    //--- значение 0 (пустое значение) не будет участвовать в отрисовке
    PlotIndexSetDouble(0,PLOT_EMPTY_VALUE,0);
    //---- количество цветов для раскраски секций
    color_sections=8; // см. комментарий к свойству #property indicator_color1
    //--- проверим параметр индикатора
    if(bars_in_section<=0)
    {
        PrintFormat("Недопустимое значение длины секции=%d",bars_in_section);
        return(-1);
    }
    else divider=color_sections*bars_in_section;
    //---
    return(0);
}
//+-----+
//| Custom indicator iteration function |
//+-----+
int OnCalculate(const int rates total,
```

```

        const int prev_calculated,
        const datetime &time[],
        const double &open[],
        const double &high[],
        const double &low[],
        const double &close[],
        const long &tick_volume[],
        const long &volume[],
        const int &spread[])

{
    static int ticks=0;
//--- считаем тики для изменения стиля, цвета и толщины линии
    ticks++;
//--- если накопилось критическое число тиков
    if(ticks>=N)
    {
        //--- меняем свойства линии
        ChangeLineAppearance();
        //--- меняем цвета, которыми рисуются секции
        ChangeColors(colors,color_sections);
        //--- сбрасываем счетчик тиков в ноль
        ticks=0;
    }

//--- номер бара, с которого начнем расчет значений индикатора
    int start=0;
//--- если индикатор уже рассчитывали раньше, то установим start на предыдущий бар
    if(prev_calculated>0) start=prev_calculated-1;
//--- здесь все расчеты значений индикатора
    for(int i=start;i<rates_total;i++)
    {
        //--- если номер бара делится без остатка на длину_секции, значит это конец сек
        if(i%bars_in_section==0)
        {
            //--- конец отрезка установим на цену High этого бара
            ColorSectionBuffer[i]=high[i];
            //--- остаток от деления номера бара на длина_секции*количество_цветов
            int rest=i%divider;
            //получим номер цвета = от 0 до количество_цветов-1
            int color_indext=rest/bars_in_section;
            ColorSectionColors[i]=color_indext;
        }
        //--- если остаток от деления равен bars,
    else
    {
        //--- если ничего не подошло, то этот бар пропускаем - ставим значение 0
        ColorSectionBuffer[i]=0;
    }
}

```

```

    }
    //--- вернем значение prev_calculated для следующего вызова функции
    return(rates_total);
}
//+-----+
//|  изменяет цвет участков линии |
//+-----+
void ChangeColors(color &cols[],int plot_colors)
{
    //--- количество цветов
    int size=ArraySize(cols);
    //---
    string comm=ChartGetString(0,CHART_COMMENT)+"\r\n\r\n";

    //--- для каждого цветового индекса зададим новый цвет случайным образом
    for(int plot_color_ind=0;plot_color_ind<plot_colors;plot_color_ind++)
    {
        //--- получим случайное число
        int number=MathRand();
        //--- получим индекс в массиве col[] как остаток от целочисленного деления
        int i=number%size;
        //--- установим цвет для каждого индекса как свойство PLOT_LINE_COLOR
        PlotIndexSetInteger(0, // номер графического стиля
                           PLOT_LINE_COLOR, // идентификатор свойства
                           plot_color_ind, // индекс цвета, куда запишем цвет
                           cols[i]); // новый цвет

        //--- запишем цвета
        comm=comm+StringFormat("SectionColorIndex[%d]=%s \r\n",plot_color_ind,ColorToString(cols[i]));
        ChartSetString(0,CHART_COMMENT,comm);
    }
    //---
}
//+-----+
//|  изменяет внешний вид отображаемой линии в индикаторе |
//+-----+
void ChangeLineAppearance()
{
    //--- строка для формирования информации о свойствах линии
    string comm="";
    //--- блок изменения толщины линии
    int number=MathRand();
    //--- получим толщину как остаток от целочисленного деления
    int width=number%5; // толщина задается от 0 до 4
    //--- установим цвет как свойство PLOT_LINE_WIDTH
    PlotIndexSetInteger(0,PLOT_LINE_WIDTH,width);
    //--- запишем толщину линии
    comm=comm+" Width="+IntegerToString(width);
}

```

```
//--- блок изменения стиля линии
    number=MathRand();
//--- делитель числа равен размеру массива styles
    int size=ArraySize(styles);
//--- получим индекс для выбора нового стиля как остаток от целочисленного деления
    int style_index=number%size;
//--- установим цвет как свойство PLOT_LINE_COLOR
    PlotIndexSetInteger(0,PLOT_LINE_STYLE,styles[style_index]);
//--- запишем стиль линии
    comm=EnumToString(styles[style_index])+", "+comm;
//--- выведем информацию на график через комментарий
    Comment(comm);
}
```

DRAW__COLOR__HISTOGRAM

DRAW__COLOR__HISTOGRAM

[DRAW__HISTOGRAM](#) - [PlotIndexSetInteger\(\)](#).

DRAW__COLOR__HISTOGRAM

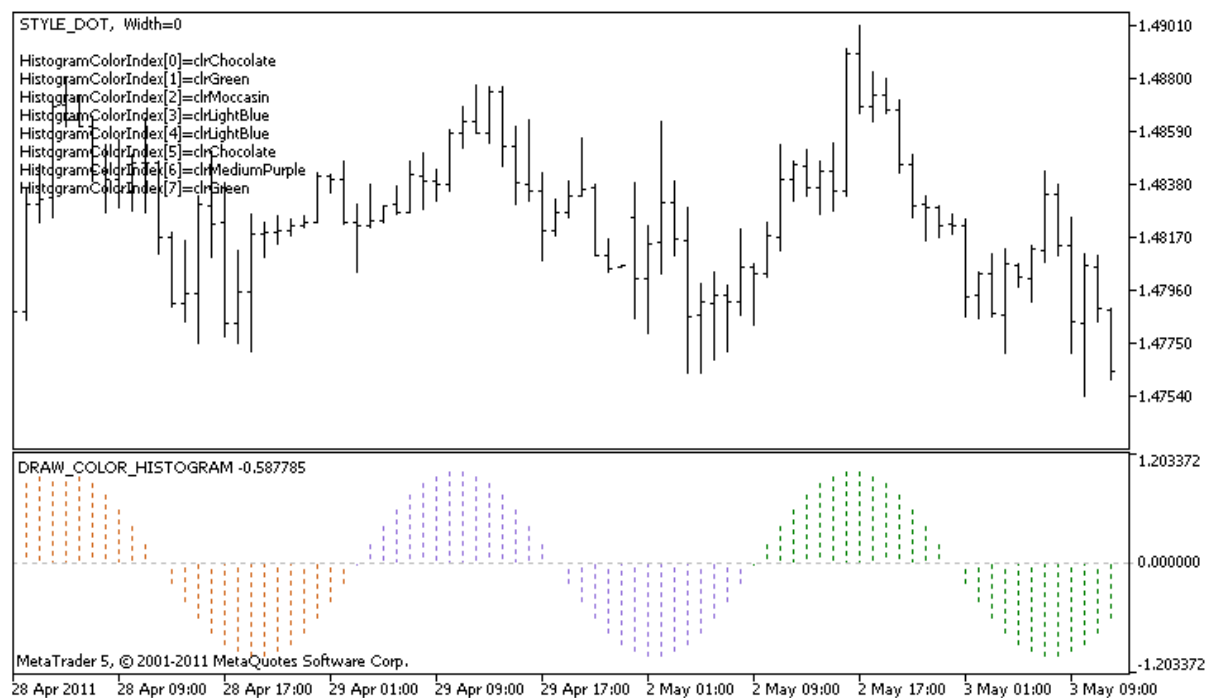
[Awesome Oscillator](#) [Market Facilitation Index](#).

DRAW__COLOR__HISTOGRAM - 2:

- [PlotIndexSetInteger\(\)](#).
- [MathSin\(\)](#).

indicator__color1 #property
64.

[MathSin\(\)](#). N
bars



```

plot1          DRAW_COLOR_HISTOGRAM          5
#property indicator_color1,
OnCalculate( )
14          ,          colors[]          N
(
)

```

```

//+-----+
//|          DRAW_COLOR_HISTOGRAM.mq5 |
//|          Copyright 2011, MetaQuotes Software Corp. |
//|          http://www.mql5.com |
//+-----+

#property copyright "Copyright 2011, MetaQuotes Software Corp."
#property link      "http://www.mql5.com"
#property version   "1.00"

#property description "Индикатор для демонстрации DRAW_COLOR_HISTOGRAM"
#property description "Рисует синусоиду гистограммой в отдельном окне"
#property description "Цвет и толщина столбиков меняется случайным образом"
#property description "через каждые N тиков"
#property description "Параметр bars задает количество баров для повторяемости синусоиды"

#property indicator_separate_window
#property indicator_buffers 2
#property indicator_plots 1

//--- input параметры
input int      bars=30;          // период синусоиды в барах
input int      N=5;              // кол-во тиков для изменения гистограммы

```

```

//--- plot Color_Histogram
#property indicator_label1  "Color_Histogram"
#property indicator_type1   DRAW_COLOR_HISTOGRAM
//--- зададим 8 цветов для раскраски секций (они хранятся в специальном массиве)
#property indicator_color1  clrRed,clrGreen,clrBlue,clrYellow,clrMagenta,clrCyan,clrM
#property indicator_style1  STYLE_SOLID
#property indicator_width1  1
//--- буфер значений
double      Color_HistogramBuffer[];
//--- буфер индексов цветов
double      Color_HistogramColors[];
//--- множитель для получения угла 2Pi в радианах при умножении на параметр bars
double      multiplier;
int         color_sections;
//--- массив для хранения цветов содержит 14 элементов
color colors[]=
{
    clrRed,clrBlue,clrGreen,clrChocolate,clrMagenta,clrDodgerBlue,clrGoldenrod,
    clrIndigo,clrLightBlue,clrAliceBlue,clrMoccasin,clrWhiteSmoke,clrCyan,clrMediumPur
};
//--- массив для хранения стилей отрисовки линии
ENUM_LINE_STYLE styles[]={STYLE_SOLID,STYLE_DASH,STYLE_DOT,STYLE_DASHDOT,STYLE_DASHDO
//+-----+
//| Custom indicator initialization function                                     |
//+-----+
int OnInit()
{
    //--- indicator buffers mapping
    SetIndexBuffer(0,Color_HistogramBuffer,INDICATOR_DATA);
    SetIndexBuffer(1,Color_HistogramColors,INDICATOR_COLOR_INDEX);
    //---- количество цветов для раскраски синусоиды
    color_sections=8;    // см. комментарий к свойству #property indicator_color1
    //--- вычислим множитель
    if(bars>1)multiplier=2.*M_PI/bars;
    else
    {
        PrintFormat("Задайте значение bars=%d больше 1",bars);
        //--- досрочное прекращение работы индикатора
        return(-1);
    }
    //---
    return(0);
}
//+-----+
//| Custom indicator iteration function                                     |
//+-----+
int OnCalculate(const int rates_total,
               const int prev_calculated,

```

```

        const datetime &time[],
        const double &open[],
        const double &high[],
        const double &low[],
        const double &close[],
        const long &tick_volume[],
        const long &volume[],
        const int &spread[])

{
    static int ticks=0;
//--- считаем тики для изменения стиля, цвета и толщины линии
    ticks++;
//--- если накопилось критическое число тиков
    if(ticks>=N)
    {
        //--- меняем свойства линии
        ChangeLineAppearance();
        //--- меняем цвета, которыми рисуется гистограмма
        ChangeColors(colors,color_sections);
        //--- сбрасываем счетчик тиков в ноль
        ticks=0;
    }

//--- вычисления значений индикатора
    int start=0;
//--- если расчет уже производился на предыдущем запуске OnCalculate
    if(prev_calculated>0) start=prev_calculated-1; // установим начало расчетов с пред
//--- заполняем индикаторный буфер значениями
    for(int i=start;i<rates_total;i++)
    {
        //--- значение
        Color_HistogramBuffer[i]=sin(i*multiplier);
        //--- цвет
        int color_index=i%(bars*color_sections);
        color_index/=bars;
        Color_HistogramColors[i]=color_index;
    }
//--- вернем значение prev_calculated для следующего вызова функции
    return(rates_total);
}

//+-----+
//|  изменяет цвет участков линии  |
//+-----+
void ChangeColors(color &cols[],int plot_colors)
{
//--- количество цветов
    int size=ArraySize(cols);
//---

```



```

string comm=ChartGetString(0,CHART_COMMENT)+"\r\n\r\n";

//--- для каждого цветового индекса зададим новый цвет случайным образом
for(int plot_color_ind=0;plot_color_ind<plot_colors;plot_color_ind++)
{
    //--- получим случайное число
    int number=MathRand();
    //--- получим индекс в массиве col[] как остаток от целочисленного деления
    int i=number%size;
    //--- установим цвет для каждого индекса как свойство PLOT_LINE_COLOR
    PlotIndexSetInteger(0, // номер графического стиля
                       PLOT_LINE_COLOR, // идентификатор свойства
                       plot_color_ind, // индекс цвета, куда запишем цвет
                       cols[i]); // новый цвет

    //--- запишем цвета
    comm=comm+StringFormat("HistogramColorIndex[%d]=%s \r\n",plot_color_ind,ColorTo
    ChartSetString(0,CHART_COMMENT,comm);
}

//---
}

//+-----+
//| изменяет внешний вид отображаемой линии в индикаторе |
//+-----+
void ChangeLineAppearance()
{
    //--- строка для формирования информации о свойствах линии
    string comm="";
    //--- блок изменения толщины линии
    int number=MathRand();
    //--- получим толщину как остаток от целочисленного деления
    int width=number%5; // толщина задается от 0 до 4
    //--- установим цвет как свойство PLOT_LINE_WIDTH
    PlotIndexSetInteger(0,PLOT_LINE_WIDTH,width);
    //--- запишем толщину линии
    comm=comm+" Width="+IntegerToString(width);

    //--- блок изменения стиля линии
    number=MathRand();
    //--- делитель числа равен размеру массива styles
    int size=ArraySize(styles);
    //--- получим индекс для выбора нового стиля как остаток от целочисленного деления
    int style_index=number%size;
    //--- установим цвет как свойство PLOT_LINE_COLOR
    PlotIndexSetInteger(0,PLOT_LINE_STYLE,styles[style_index]);
    //--- запишем стиль линии
    comm=EnumToString(styles[style_index])+", "+comm;
    //--- выведем информацию на график через комментарий
    Comment(comm);
}

```

```
}
```

DRAW__COLOR__HISTOGRAM2

DRAW__COLOR__HISTOGRAM2

DRAW__HISTOGRAM2

DRAW__HISTOGRAM2 -PlotIndexSetInteger().

DRAW__COLOR__HISTOGRAM2

DRAW__COLOR__HISTOGRAM2 - 3:

•

;

•

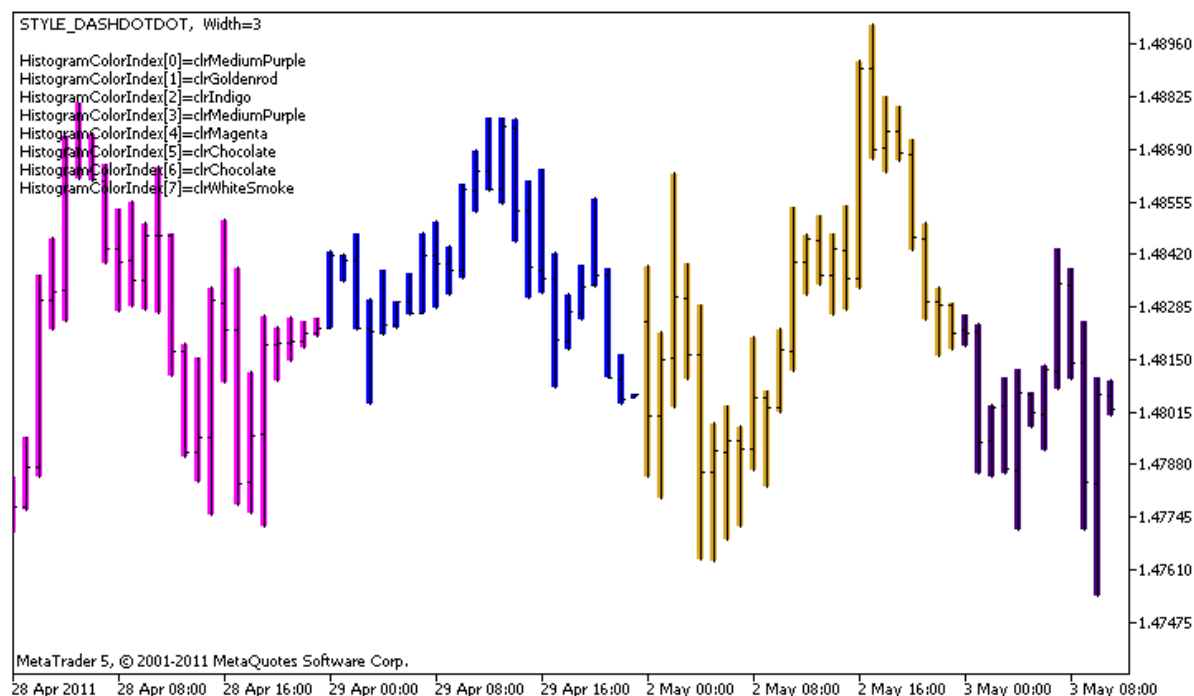
(

)

).

High Low.

N



```

plot1          DRAW_COLOR_HISTOGRAM2          5
                                     #property indicator_color1,
                                     OnCalculate( )
14                                     colors[]
N                                     (
                                     "
                                     )

```

```

//+-----+
//|                                     DRAW_COLOR_HISTOGRAM2.mq5 |
//|                                     Copyright 2011, MetaQuotes Software Corp. |
//|                                     http://www.mql5.com |
//+-----+

#property copyright "Copyright 2011, MetaQuotes Software Corp."
#property link      "http://www.mql5.com"
#property version   "1.00"

#property description "Индикатор для демонстрации DRAW_COLOR_HISTOGRAM2"
#property description "Рисует на каждом баре отрезок между Open и Close"
#property description "Цвет, толщина и стиль меняется случайным образом"
#property description "через каждые N тиков"

#property indicator_chart_window
#property indicator_buffers 3
#property indicator_plots 1
//--- plot ColorHistogram_2
#property indicator_label1 "ColorHistogram_2"
#property indicator_type1 DRAW_COLOR_HISTOGRAM2

```

```

//--- зададим 5 цветов для раскраски гистограммы по дням недели (они хранятся в специ
#property indicator_color1 clrRed,clrBlue,clrGreen,clrYellow,clrMagenta
#property indicator_style1 STYLE_SOLID
#property indicator_width1 1

//--- input параметр
input int      N=5;           // кол-во тиков для изменения гистограммы
int          color_sections;
//--- буферы значений
double       ColorHistogram_2Buffer1[];
double       ColorHistogram_2Buffer2[];
//--- буфер индексов цвета
double       ColorHistogram_2Colors[];
//--- массив для хранения цветов содержит 14 элементов
color colors[]=
{
    clrRed,clrBlue,clrGreen,clrChocolate,clrMagenta,clrDodgerBlue,clrGoldenrod,
    clrIndigo,clrLightBlue,clrAliceBlue,clrMoccasin,clrWhiteSmoke,clrCyan,clrMediumPur
};
//--- массив для хранения стилей отрисовки линии
ENUM_LINE_STYLE styles[]={STYLE_SOLID,STYLE_DASH,STYLE_DOT,STYLE_DASHDOT,STYLE_DASHDO
//+-----+
//| Custom indicator initialization function |
//+-----+
int OnInit()
{
    //--- indicator buffers mapping
    SetIndexBuffer(0,ColorHistogram_2Buffer1,INDICATOR_DATA);
    SetIndexBuffer(1,ColorHistogram_2Buffer2,INDICATOR_DATA);
    SetIndexBuffer(2,ColorHistogram_2Colors,INDICATOR_COLOR_INDEX);
    //--- установим пустое значение
    PlotIndexSetDouble(0,PLOT_EMPTY_VALUE,0);
    //--- количество цветов для раскраски синусоиды
    color_sections=8; // см. комментарий к свойству #property indicator_color1
    //---
    return(0);
}
//+-----+
//| Custom indicator iteration function |
//+-----+
int OnCalculate(const int rates_total,
               const int prev_calculated,
               const datetime &time[],
               const double &open[],
               const double &high[],
               const double &low[],
               const double &close[],
               const long &tick_volume[],

```

```

        const long &volume[],
        const int &spread[])
    {
        static int ticks=0;
//--- считаем тики для изменения стиля, цвета и толщины линии
        ticks++;
//--- если накопилось критическое число тиков
        if(ticks>=N)
        {
            //--- меняем свойства линии
            ChangeLineAppearance();
            //--- меняем цвета, которыми рисуются гистограмма
            ChangeColors(colors,color_sections);
            //--- сбрасываем счетчик тиков в ноль
            ticks=0;
        }

//--- вычисления значений индикатора
        int start=0;
//--- для получения дня недели по времени открытия каждого бара
        MqlDateTime dt;
//--- если расчет уже производился на предыдущем запуске OnCalculate
        if(prev_calculated>0) start=prev_calculated-1; // установим начало расчетов с пред
//--- заполняем индикаторный буфер значениями
        for(int i=start;i<rates_total;i++)
        {
            TimeToStruct(time[i],dt);
            //--- значени
            ColorHistogram_2Buffer1[i]=high[i];
            ColorHistogram_2Buffer2[i]=low[i];
            //--- зададим индекс цвета по дню недели
            int day=dt.day_of_week;
            ColorHistogram_2Colors[i]=day;
        }
//--- вернем значение prev_calculated для следующего вызова функции
        return(rates_total);
    }
//+-----+
//|   изменяет цвет участков линии   |
//+-----+
void ChangeColors(color &cols[],int plot_colors)
{
//--- количество цветов
    int size=ArraySize(cols);
//---
    string comm=ChartGetString(0,CHART_COMMENT)+"\r\n\r\n";

//--- для каждого цветового индекса зададим новый цвет случайным образом

```

```

for(int plot_color_ind=0;plot_color_ind<plot_colors;plot_color_ind++)
{
    //--- получим случайное число
    int number=MathRand();
    //--- получим индекс в массиве col[] как остаток от целочисленного деления
    int i=number%size;
    //--- установим цвет для каждого индекса как свойство PLOT_LINE_COLOR
    PlotIndexSetInteger(0, // номер графического стиля
                        PLOT_LINE_COLOR, // идентификатор свойства
                        plot_color_ind, // индекс цвета, куда запишем цвет
                        cols[i]); // новый цвет

    //--- запишем цвета
    comm=comm+StringFormat("HistogramColorIndex[%d]=%s \r\n",plot_color_ind,ColorTo
    ChartSetString(0,CHART_COMMENT,comm);
}
//---
}
//+-----+
//| изменяет внешний вид отображаемой линии в индикаторе |
//+-----+
void ChangeLineAppearance()
{
    //--- строка для формирования информации о свойствах линии
    string comm="";
    //--- блок изменения толщины линии
    int number=MathRand();
    //--- получим толщину как остаток от целочисленного деления
    int width=number%5; // толщина задается от 0 до 4
    //--- установим цвет как свойство PLOT_LINE_WIDTH
    PlotIndexSetInteger(0,PLOT_LINE_WIDTH,width);
    //--- запишем толщину линии
    comm=comm+" Width="+IntegerToString(width);

    //--- блок изменения стиля линии
    number=MathRand();
    //--- делитель числа равен размеру массива styles
    int size=ArraySize(styles);
    //--- получим индекс для выбора нового стиля как остаток от целочисленного деления
    int style_index=number%size;
    //--- установим цвет как свойство PLOT_LINE_COLOR
    PlotIndexSetInteger(0,PLOT_LINE_STYLE,styles[style_index]);
    //--- запишем стиль линии
    comm=EnumToString(styles[style_index])+", "+comm;
    //--- выведем информацию на график через комментарий
    Comment(comm);
}

```

DRAW_COLOR_ARROW

```

DRAW_COLOR_ARROW
(
    Wingdings)
    DRAW_ARROW,
    indicator_color1.

    DRAW_ARROW - PlotIndexSetInteger().

```

PLOT_ARROW.

```

//--- зададим код символа из шрифта Wingdings для отрисовки в PLOT_ARROW
PlotIndexSetInteger(0,PLOT_ARROW,code);

```

PLOT_ARROW=159 () .

) .

```

//--- зададим смещение стрелок по вертикали в пикселях
PlotIndexSetInteger(0,PLOT_ARROW_SHIFT,shift);

```

PLOT_ARROW_SHIFT

DRAW_COLOR_ARROW

" "

```

//--- установим пустое значение
PlotIndexSetDouble(индекс_построения_DRAW_COLOR_ARROW,PLOT_EMPTY_VALUE,0);

```

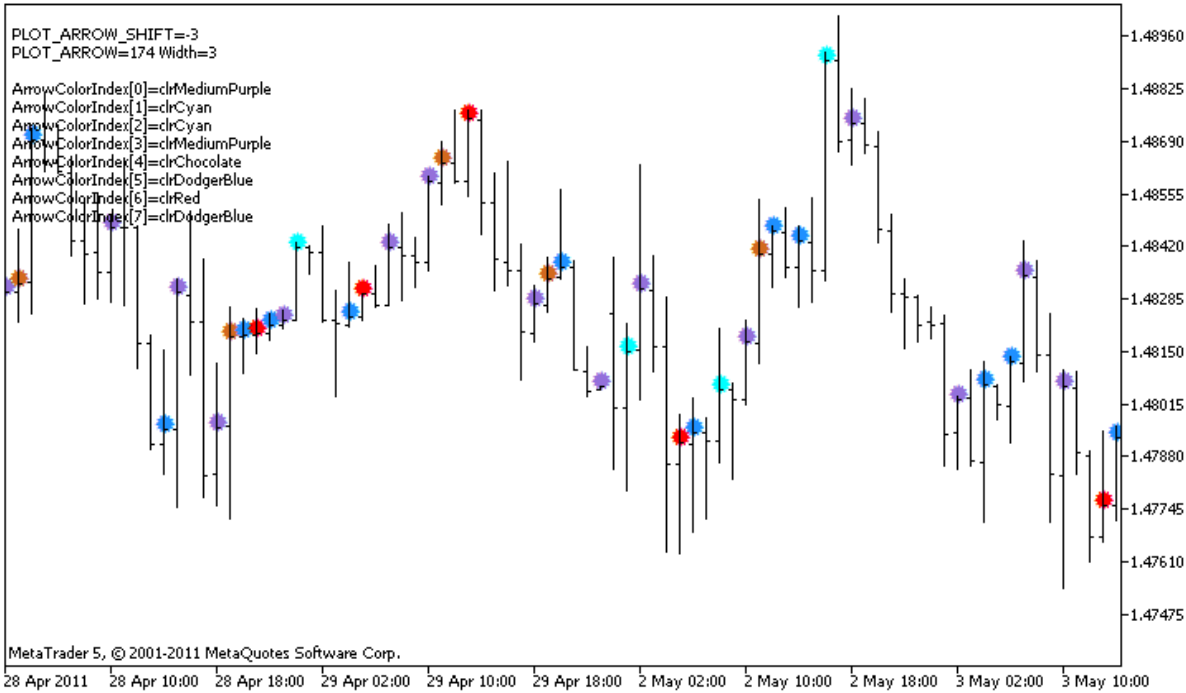
DRAW_COLOR_ARROW - 2:

• (PLOT_ARROW_SHIFT) ;

• () .

Close

N



plot1 DRAW_COLOR_ARROW

#property,

OnCalculate()

N

) .

(

"

"

8

#property,

OnCalculate()

14

colors[].

```
//+-----+
//|                                     DRAW_COLOR_ARROW.mq5 |
//|                                     Copyright 2011, MetaQuotes Software Corp. |
//|                                     http://www.mql5.com |
//+-----+
```

```

#property copyright "Copyright 2011, MetaQuotes Software Corp."
#property link      "http://www.mql5.com"
#property version   "1.00"

#property description "Индикатор для демонстрации DRAW_COLOR_ARROW"
#property description "Рисует на графике разным цветом стрелки, задаваемые символами"
#property description "Цвет, размер, смещение и код символа стрелки меняется"
#property description "случайным образом через каждые N тиков"
#property description "Параметр code задает базовое значение: код=159 (кружок)"

#property indicator_chart_window
#property indicator_buffers 2
#property indicator_plots 1
//--- plot ColorArrow
#property indicator_label1 "ColorArrow"
#property indicator_type1 DRAW_COLOR_ARROW
//--- зададим 8 цветов для раскраски гистограммы по дням недели (они хранятся в специ
#property indicator_color1 clrRed,clrBlue,clrSeaGreen,clrGold,clrDarkOrange,clrMagen
#property indicator_style1 STYLE_SOLID
#property indicator_width1 1

//--- input параметры
input int      N=5;           // кол-во тиков для изменения
input ushort   code=159;      // код символа для отрисовки в DRAW_ARROW
int          color_sections;
//--- индикаторный буфер для построения
double        ColorArrowBuffer[];
//--- буфер для хранения индексов цвета
double        ColorArrowColors[];
//--- массив для хранения цветов содержит 14 элементов
color colors[]=
{
    clrRed,clrBlue,clrGreen,clrChocolate,clrMagenta,clrDodgerBlue,clrGoldenrod,
    clrIndigo,clrLightBlue,clrAliceBlue,clrMoccasin,clrWhiteSmoke,clrCyan,clrMediumPur
};
//+-----+
//| Custom indicator initialization function |
//+-----+
int OnInit()
{
    //--- indicator buffers mapping
    SetIndexBuffer(0,ColorArrowBuffer,INDICATOR_DATA);
    SetIndexBuffer(1,ColorArrowColors,INDICATOR_COLOR_INDEX);
    //--- зададим код символа для отрисовки в PLOT_ARROW
    PlotIndexSetInteger(0,PLOT_ARROW,code);
    //--- зададим смещение стрелок по вертикали в пикселях
    PlotIndexSetInteger(0,PLOT_ARROW_SHIFT,5);
    //--- установим в качестве пустого значения 0

```

```

    PlotIndexSetDouble(0,PLOT_EMPTY_VALUE,0);
//---- количество цветов для раскраски синусоиды
    color_sections=8;    // см. комментарий к свойству #property indicator_color1
//---
    return(0);
}
//+-----+
//| Custom indicator iteration function |
//+-----+
int OnCalculate(const int rates_total,
               const int prev_calculated,
               const datetime &time[],
               const double &open[],
               const double &high[],
               const double &low[],
               const double &close[],
               const long &tick_volume[],
               const long &volume[],
               const int &spread[])
{
    static int ticks=0;
//--- считаем тики для изменения цвета, размера, смещения и кода стрелки
    ticks++;
//--- если накопилось критическое число тиков
    if(ticks>=N)
    {
        //--- меняем свойства стрелок
        ChangeLineAppearance();
        //--- меняем цвета, которыми рисуются гистограмма
        ChangeColors(colors,color_sections);
        //--- сбрасываем счетчик тиков в ноль
        ticks=0;
    }

//--- блок расчета значений индикатора
    int start=1;
    if(prev_calculated>0) start=prev_calculated-1;
//--- цикл расчета
    for(int i=1;i<rates_total;i++)
    {
        //--- если текущая цена Close больше предыдущей, ставим стрелку
        if(close[i]>close[i-1])
            ColorArrowBuffer[i]=close[i];
        //--- в противном случае указываем нулевое значение
        else
            ColorArrowBuffer[i]=0;
        //--- цвет стрелки
        int index=i%color_sections;

```

```

        ColorArrowColors[i]=index;
    }
    //--- return value of prev_calculated for next call
    return(rates_total);
}
//+-----+
//|  изменяет цвет участков линии |
//+-----+
void ChangeColors(color &cols[],int plot_colors)
{
    //--- количество цветов
    int size=ArraySize(cols);
    //---
    string comm=ChartGetString(0,CHART_COMMENT)+"\r\n\r\n";

    //--- для каждого цветового индекса зададим новый цвет случайным образом
    for(int plot_color_ind=0;plot_color_ind<plot_colors;plot_color_ind++)
    {
        //--- получим случайное число
        int number=MathRand();
        //--- получим индекс в массиве col[] как остаток от целочисленного деления
        int i=number%size;
        //--- установим цвет для каждого индекса как свойство PLOT_LINE_COLOR
        PlotIndexSetInteger(0, // номер графического стиля
                           PLOT_LINE_COLOR, // идентификатор свойства
                           plot_color_ind, // индекс цвета, куда запишем цвет
                           cols[i]); // новый цвет

        //--- запишем цвета
        comm=comm+StringFormat("ArrowColorIndex[%d]=%s \r\n",plot_color_ind,ColorToString(cols[i]));
        ChartSetString(0,CHART_COMMENT,comm);
    }
    //---
}
//+-----+
//|  изменяет внешний вид отображаемой линии в индикаторе |
//+-----+
void ChangeLineAppearance()
{
    //--- строка для формирования информации о свойствах линии
    string comm="";
    //--- блок изменения толщины линии
    int number=MathRand();
    //--- получим толщину как остаток от целочисленного деления
    int width=number%5; // толщина задается от 0 до 4
    //--- установим цвет как свойство PLOT_LINE_WIDTH
    PlotIndexSetInteger(0,PLOT_LINE_WIDTH,width);
    //--- запишем толщину линии
    comm=comm+" Width="+IntegerToString(width);
}

```

```
//--- блок изменения кода стрелки (PLOT_ARROW)
number=MathRand();
//--- получим остаток от целочисленного деления для вычисления нового кода стрелки (o
int code_add=number%20;
//--- установим новый код символа как сумму code+code_add
PlotIndexSetInteger(0,PLOT_ARROW,code+code_add);
//--- запишем код символа PLOT_ARROW
comm="\r\n"+"PLOT_ARROW="+IntegerToString(code+code_add)+comm;

//--- блок изменения смещения стрелок по вертикали в пикселях
number=MathRand();
//--- получим смещение как остаток от целочисленного деления
int shift=20-number%41;
//--- установим новое смещение
PlotIndexSetInteger(0,PLOT_ARROW_SHIFT,shift);
//--- запишем смещение PLOT_ARROW_SHIFT
comm="\r\n"+"PLOT_ARROW_SHIFT="+IntegerToString(shift)+comm;

//--- выведем информацию на график через комментарий
Comment(comm);
}
```

DRAW_COLOR_ZIGZAG

DRAW_COLOR_ZIGZAG

[DRAW_ZIGZAG](#),[DRAW_ZIGZAG](#) - _____[PlotIndexSetInteger\(\)](#).[PLOT_EMPTY_VALUE](#):

```
//--- значение 0 (пустое значение) не будет участвовать в отрисовке
PlotIndexSetDouble(индекс_построения_DRAW_COLOR_ZIGZAG,PLOT_EMPTY_VALUE,0);
```

DRAW_COLOR_ZIGZAG - 3:

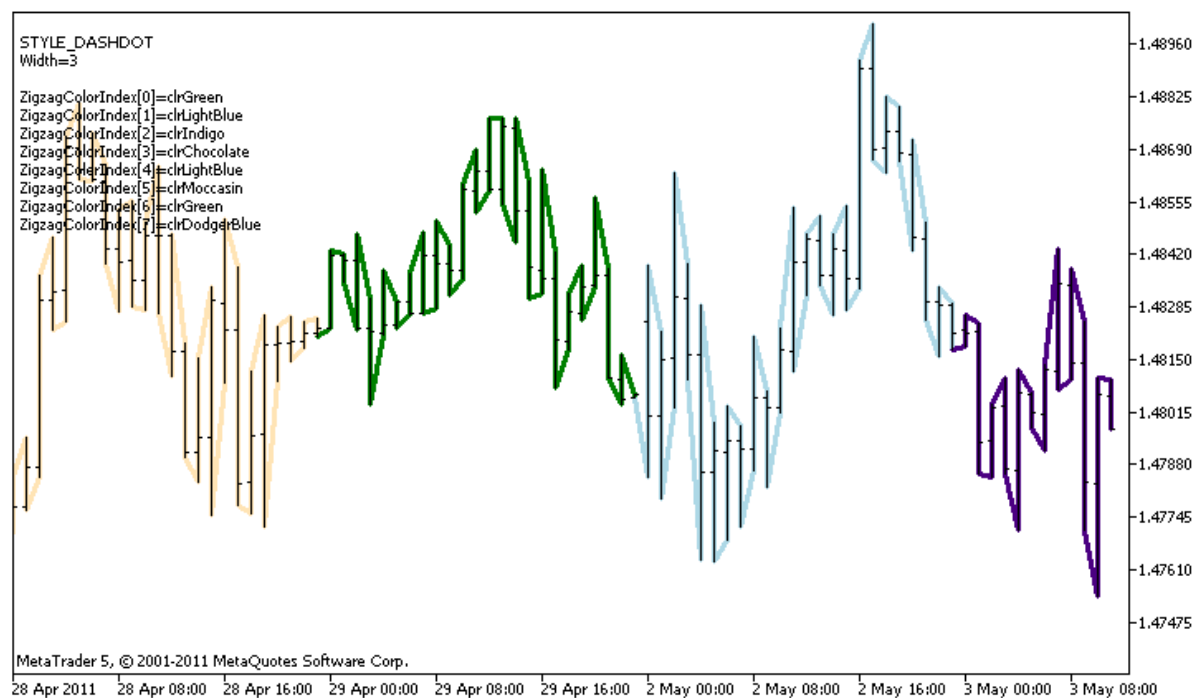
•

•

() .

High Low.

N



plot1

DRAW_COLOR_ZIGZAG

8

#property,

OnCalculate()

14

colors[].

N

).

```
//+-----+
//|                                     DRAW_COLOR_ZIGZAG.mq5 |
//|                                     Copyright 2011, MetaQuotes Software Corp. |
//|                                     http://www.mql5.com |
//+-----+

#property copyright "Copyright 2011, MetaQuotes Software Corp."
#property link      "http://www.mql5.com"
#property version   "1.00"

#property description "Индикатор для демонстрации DRAW_COLOR_ZIGZAG"
#property description "Рисует ломанную линию цветными отрезками, цвет зависит от номе"
#property description "Цвет, толщина и стиль отрезков меняется случайным"
#property description "образом через каждые N тиков"

#property indicator_chart_window
#property indicator_buffers 3
#property indicator_plots 1
//--- plot Color_Zigzag
#property indicator_label1 "Color_Zigzag"
#property indicator_type1 DRAW_COLOR_ZIGZAG
```

```

//--- зададим 8 цветов для раскраски секций (они хранятся в специальном массиве)
#property indicator_color1 clrRed,clrBlue,clrGreen,clrYellow,clrMagenta,clrCyan,clrL
#property indicator_style1 STYLE_SOLID
#property indicator_width1 1
//--- input параметр
input int      N=5;                // кол-во тиков для изменения
int           color_sections;
//--- буферы значений концов отрезков
double        Color_ZigzagBuffer1[];
double        Color_ZigzagBuffer2[];
//--- буфер индексов цвета для концов отрезков
double        Color_ZigzagColors[];
//--- массив для хранения цветов содержит 14 элементов
color colors[]=
{
    clrRed,clrBlue,clrGreen,clrChocolate,clrMagenta,clrDodgerBlue,clrGoldenrod,
    clrIndigo,clrLightBlue,clrAliceBlue,clrMoccasin,clrWhiteSmoke,clrCyan,clrMediumPur
};
//--- массив для хранения стилей отрисовки линии
ENUM_LINE_STYLE styles[]={STYLE_SOLID,STYLE_DASH,STYLE_DOT,STYLE_DASHDOT,STYLE_DASHDO
//+-----+
//| Custom indicator initialization function |
//+-----+
int OnInit()
{
    //--- indicator buffers mapping
    SetIndexBuffer(0,Color_ZigzagBuffer1,INDICATOR_DATA);
    SetIndexBuffer(1,Color_ZigzagBuffer2,INDICATOR_DATA);
    SetIndexBuffer(2,Color_ZigzagColors,INDICATOR_COLOR_INDEX);
    //---- количество цветов для раскраски зигзага
    color_sections=8;    // см. комментарий к свойству #property indicator_color1
    //---
    return(0);
}
//+-----+
//| Custom indicator iteration function |
//+-----+
int OnCalculate(const int rates_total,
                const int prev_calculated,
                const datetime &time[],
                const double &open[],
                const double &high[],
                const double &low[],
                const double &close[],
                const long &tick_volume[],
                const long &volume[],
                const int &spread[])
{

```



```

static int ticks=0;
//--- считаем тики для изменения стиля, цвета и толщины линии
ticks++;
//--- если накопилось достаточное число тиков
if(ticks>=N)
{
    //--- меняем свойства линии
    ChangeLineAppearance();
    //--- меняем цвета, которыми рисуются секции
    ChangeColors(colors,color_sections);
    //--- сбрасываем счетчик тиков в ноль
    ticks=0;
}

//--- структура времени понадобится для получения дня недели каждого бара
MqlDateTime dt;

//--- позиция начала расчетов
int start=0;
//--- если индикатор рассчитывался на предыдущем тике, то начинаю расчет с предпослед
if(prev_calculated!=0) start=prev_calculated-1;
//--- цикл расчетов
for(int i=start;i<rates_total;i++)
{
    //--- запишем время открытия бара в структуру
    TimeToStruct(time[i],dt);

    //--- если номер бара четный
    if(i%2==0)
    {
        //--- пишем в 1-ый буфер High, во 2-ой Low
        Color_ZigzagBuffer1[i]=high[i];
        Color_ZigzagBuffer2[i]=low[i];
        //--- цвет отрезка
        Color_ZigzagColors[i]=dt.day_of_year%color_sections;
    }
    //--- номер бара нечетный
    else
    {
        //--- заполняем бар в обратном порядке
        Color_ZigzagBuffer1[i]=low[i];
        Color_ZigzagBuffer2[i]=high[i];
        //--- цвет отрезка
        Color_ZigzagColors[i]=dt.day_of_year%color_sections;
    }
}

//--- return value of prev_calculated for next call
return(rates_total);

```

```

    }
//+-----+
//|  изменяет цвет отрезков зигзага
//+-----+
void ChangeColors(color &cols[],int plot_colors)
{
//--- количество цветов
    int size=ArraySize(cols);
//---
    string comm=ChartGetString(0,CHART_COMMENT)+"\r\n\r\n";

//--- для каждого цветового индекса зададим новый цвет случайным образом
    for(int plot_color_ind=0;plot_color_ind<plot_colors;plot_color_ind++)
    {
        //--- получим случайное число
        int number=MathRand();
        //--- получим индекс в массиве col[] как остаток от целочисленного деления
        int i=number%size;
        //--- установим цвет для каждого индекса как свойство PLOT_LINE_COLOR
        PlotIndexSetInteger(0, // номер графического стиля
                            PLOT_LINE_COLOR, // идентификатор свойства
                            plot_color_ind, // индекс цвета, куда запишем цвет
                            cols[i]); // новый цвет

        //--- запишем цвета
        comm=comm+StringFormat("ZigzagColorIndex[%d]=%s \r\n",plot_color_ind,ColorToString(cols[i]));
        ChartSetString(0,CHART_COMMENT,comm);
    }
//---
}
//+-----+
//|  изменяет внешний вид отрезков в зигзаге
//+-----+
void ChangeLineAppearance()
{
//--- строка для формирования информации о свойствах Color_ZigZag
    string comm="";
//--- блок изменения толщины линии
    int number=MathRand();
//--- получим толщину как остаток от целочисленного деления
    int width=number%5; // толщина задается от 0 до 4
//--- установим цвет как свойство PLOT_LINE_WIDTH
    PlotIndexSetInteger(0,PLOT_LINE_WIDTH,width);
//--- запишем толщину линии
    comm=comm+"\r\nWidth="+IntegerToString(width);

//--- блок изменения стиля линии
    number=MathRand();
//--- делитель числа равен размеру массива styles

```

```
int size=ArraySize(styles);  
//--- получим индекс для выбора нового стиля как остаток от целочисленного деления  
int style_index=number%size;  
//--- установим цвет как свойство PLOT_LINE_COLOR  
PlotIndexSetInteger(0,PLOT_LINE_STYLE,styles[style_index]);  
//--- запишем стиль линии  
comm="\r\n"+EnumToString(styles[style_index])+" "+comm;  
//--- выведем информацию на график через комментарий  
Comment(comm);  
}
```

DRAW_COLOR_BARS

DRAW_COLOR_BARS

Low Close.

DRAW_BARS

Open, High,

PlotIndexSetInteger().PLOT_EMPTY_VALUE:

```
//--- значение 0 (пустое значение) не будет участвовать в отрисовке
PlotIndexSetDouble(индекс_построения_DRAW_COLOR_BARS, PLOT_EMPTY_VALUE, 0);
```

DRAW_COLOR_BARS - 5:

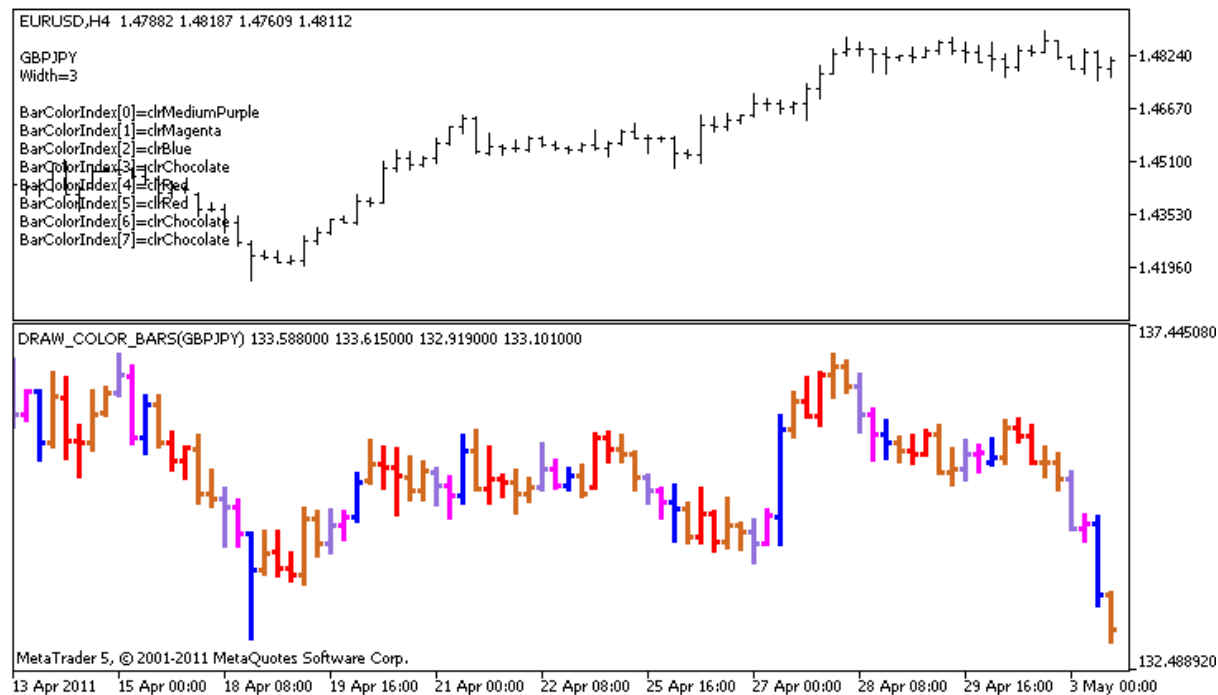
- Open, High, Low Close;
- () .

: Open, High, Low, Close

N N

(" "

) .



plot1

DRAW_COLOR_BARS

8

#property,

OnCalculate()

14

colors[].

```
//+-----+
//|                                     DRAW_COLOR_BARS.mq5 |
//|                                     Copyright 2011, MetaQuotes Software Corp. |
//|                                     http://www.mql5.com |
//+-----+

#property copyright "Copyright 2011, MetaQuotes Software Corp."
#property link      "http://www.mql5.com"
#property version   "1.00"

#property description "Индикатор для демонстрации DRAW_COLOR_BARS"
#property description "Рисует в отдельном окне разным цветом бары по выбранному символу"
#property description "Цвет и толщина баров, а также символ, меняются случайным образом"
#property description "образом через каждые N тиков"

#property indicator_separate_window
#property indicator_buffers 5
#property indicator_plots 1
//--- plot ColorBars
#property indicator_label1 "ColorBars"
#property indicator_type1 DRAW_COLOR_BARS
//--- зададим 8 цветов для раскраски баров (они хранятся в специальном массиве)
#property indicator_color1 clrRed, clrBlue, clrGreen, clrYellow, clrMagenta, clrCyan, clrL
#property indicator_style1 STYLE_SOLID
#property indicator_width1 1
```

```

//--- input параметры
input int      N=5;           // количество тиков для смены вида
input int      bars=500;      // сколько баров показывать
input bool     messages=false; // вывод сообщений в лог "Эксперты"
//--- индикаторные буферы
double        ColorBarsBuffer1[];
double        ColorBarsBuffer2[];
double        ColorBarsBuffer3[];
double        ColorBarsBuffer4[];
double        ColorBarsColors[];
//--- имя символа
string symbol;
int          bars_colors;
//--- массив для хранения цветов содержит 14 элементов
color colors[]=
{
    clrRed,clrBlue,clrGreen,clrChocolate,clrMagenta,clrDodgerBlue,clrGoldenrod,
    clrIndigo,clrLightBlue,clrAliceBlue,clrMoccasin,clrMagenta,clrCyan,clrMediumPurple
};
//+-----+
//| Custom indicator initialization function |
//+-----+
int OnInit()
{
    //--- indicator buffers mapping
    SetIndexBuffer(0,ColorBarsBuffer1,INDICATOR_DATA);
    SetIndexBuffer(1,ColorBarsBuffer2,INDICATOR_DATA);
    SetIndexBuffer(2,ColorBarsBuffer3,INDICATOR_DATA);
    SetIndexBuffer(3,ColorBarsBuffer4,INDICATOR_DATA);
    SetIndexBuffer(4,ColorBarsColors,INDICATOR_COLOR_INDEX);
    //---- количество цветов для раскраски баров
    bars_colors=8; // см. комментарий к свойству #property indicator_color1
    //---
    return(0);
}
//+-----+
//| Custom indicator iteration function |
//+-----+
int OnCalculate(const int rates_total,
               const int prev_calculated,
               const datetime &time[],
               const double &open[],
               const double &high[],
               const double &low[],
               const double &close[],
               const long &tick_volume[],
               const long &volume[],
               const int &spread[])

```

```

{
    static int ticks=0;
    //--- считаем тики для изменения стиля, цвета и толщины бара
    ticks++;
    //--- если накопилось достаточное число тиков
    if(ticks>=N)
    {
        //--- выберем новый символ из окна "Обзор рынка"
        symbol=GetRandomSymbolName();
        //--- меняем свойства линии
        ChangeLineAppearance();
        //--- меняем цвета, которыми рисуются бары
        ChangeColors(colors,bars_colors);
        int tries=0;
        //--- сделаем 5 попыток заполнить буфера ценами из symbol
        while(!CopyFromSymbolToBuffers(symbol,rates_total,bars_colors) && tries<5)
        {
            //--- счетчик вызовов функции CopyFromSymbolToBuffers()
            tries++;
        }
        //--- сбрасываем счетчик тиков в ноль
        ticks=0;
    }
    //--- return value of prev_calculated for next call
    return(rates_total);
}

//+-----+
//|   заполняем индикаторные буфера ценами   |
//+-----+
bool CopyFromSymbolToBuffers(string name,int total,int bar_colors)
{
    //--- в массив rates[] будем копировать цены Open, High, Low и Close
    MqlRates rates[];
    //--- счетчик попыток
    int attempts=0;
    //--- сколько скопировано
    int copied=0;
    //--- делаем 25 попыток получить таймсерия по нужному символу
    while(attempts<25 && (copied=CopyRates(name,_Period,0,bars,rates))<0)
    {
        Sleep(100);
        attempts++;
        if(messages) PrintFormat("%s CopyRates(%s) attempts=%d",__FUNCTION__,name,attempts);
    }
    //--- если не удалось скопировать достаточное количество баров
    if(copied!=bars)
    {
        //--- сформируем строку сообщения
    }
}

```

```

        string comm=StringFormat("Для символа %s удалось получить только %d баров из %d",
                                   name,
                                   copied,
                                   bars
                                   );

        //--- выведем сообщение в комментарий на главное окно графика
        Comment(comm);
        //--- выводим сообщения
        if(messages) Print(comm);
        return(false);
    }
    else
    {
        //--- установим отображение символа
        PlotIndexSetString(0,PLOT_LABEL,name+" Open;" +name+" High;" +name+" Low;" +name+"
        IndicatorSetString(INDICATOR_SHORTNAME,"DRAW_COLOR_BARS(" +name+" )");
    }
//--- инициализируем буфера пустыми значениями
ArrayInitialize(ColorBarsBuffer1,0.0);
ArrayInitialize(ColorBarsBuffer2,0.0);
ArrayInitialize(ColorBarsBuffer3,0.0);
ArrayInitialize(ColorBarsBuffer4,0.0);

//--- копируем цены в буферы
for(int i=0;i<copied;i++)
{
    //--- вычислим соответствующий индекс для буферов
    int buffer_index=total-copied+i;
    //--- записываем цены в буфера
    ColorBarsBuffer1[buffer_index]=rates[i].open;
    ColorBarsBuffer2[buffer_index]=rates[i].high;
    ColorBarsBuffer3[buffer_index]=rates[i].low;
    ColorBarsBuffer4[buffer_index]=rates[i].close;
    //---
    ColorBarsColors[buffer_index]=i%bar_colors;
}
return(true);
}

//+-----+
//| возвращает случайным образом символ из Market Watch |
//+-----+
string GetRandomSymbolName()
{
    //--- количество символов, показываемых в окне "Обзор рынка"
    int symbols=SymbolsTotal(true);
    //--- позиция символа в списке - случайное число от 0 до symbols
    int number=MathRand()%symbols;
    //--- вернем имя символа по указанной позиции

```



```

    return SymbolName(number,true);
}

//+-----+
//|  изменяет цвет отрезков зигзага
//+-----+
void ChangeColors(color  &cols[],int plot_colors)
{
//--- количество цветов
    int size=ArraySize(cols);
//---
    string comm=ChartGetString(0,CHART_COMMENT)+"\r\n\r\n";

//--- для каждого цветового индекса зададим новый цвет случайным образом
    for(int plot_color_ind=0;plot_color_ind<plot_colors;plot_color_ind++)
    {
        //--- получим случайное число
        int number=MathRand();
        //--- получим индекс в массиве col[] как остаток от целочисленного деления
        int i=number%size;
        //--- установим цвет для каждого индекса как свойство PLOT_LINE_COLOR
        PlotIndexSetInteger(0,          // номер графического стиля
                           PLOT_LINE_COLOR,  // идентификатор свойства
                           plot_color_ind,    // индекс цвета, куда запишем цвет
                           cols[i]);          // новый цвет

        //--- запишем цвета
        comm=comm+StringFormat("BarColorIndex[%d]=%s \r\n",plot_color_ind,ColorToString
                               ChartSetString(0,CHART_COMMENT,comm);
    }
//---
}

//+-----+
//|  изменяет внешний вид баров
//+-----+
void ChangeLineAppearance()
{
//--- строка для формирования информации о свойствах баров
    string comm="";

//--- блок изменения толщины баров
    int number=MathRand();
//--- получим толщину как остаток от целочисленного деления
    int width=number%5;    // толщина задается от 0 до 4
//--- установим цвет как свойство PLOT_LINE_WIDTH
    PlotIndexSetInteger(0,PLOT_LINE_WIDTH,width);
//--- запишем толщину линии
    comm=comm+"\r\nWidth="+IntegerToString(width);

//--- запишем имя символа

```

```
comm="\r\n"+symbol+comm;  
  
//--- выведем информацию на график через комментарий  
Comment(comm);  
}
```

DRAW_COLOR_CANDLES

DRAW_COLOR_CANDLES

DRAW CANDLES

Open, High, Low Close.

PlotIndexSetInteger().

PLOT EMPTY VALUE:

```
//--- значение 0 (пустое значение) не будет участвовать в отрисовке
PlotIndexSetDouble(индекс построения DRAW COLOR CANDLES, PLOT_EMPTY_VALUE, 0);
```

DRAW_COLOR_CANDLES - 5:

Open, High, Low Close;

() .

: Open, High, Low, Close

N

N



plot1

[#property,](#)[OnCalculate\(_\)](#)

```
//+-----+
//|                                     DRAW_COLOR_CANDLES.mq5 |
//|                                     Copyright 2011, MetaQuotes Software Corp. |
//|                                     http://www.mql5.com |
//+-----+

#property copyright "Copyright 2011, MetaQuotes Software Corp."
#property link      "http://www.mql5.com"
#property version   "1.00"

#property description "Индикатор для демонстрации DRAW_COLOR_CANDLES."
#property description "Рисует в отдельном окне разным цветом свечи по случайно выбран"
#property description " "
#property description "Цвет и толщина свечей, а также символ, меняются"
#property description "случайным образом через каждые N тиков."

#property indicator_separate_window
#property indicator_buffers 5
#property indicator_plots 1
//--- plot ColorCandles
#property indicator_label1 "ColorCandles"
#property indicator_type1 DRAW_COLOR_CANDLES
//--- зададим 8 цветов для раскраски свечей (они хранятся в специальном массиве)
#property indicator_color1 clrRed,clrBlue,clrGreen,clrYellow,clrMagenta,clrCyan,clrL
#property indicator_style1 STYLE_SOLID
```

```

#property indicator_width1 1

//--- input параметры
input int      N=5;           // количество тиков для смены вида
input int      bars=500;      // сколько свечей показывать
input bool     messages=false; // вывод сообщений в лог "Эксперты"
//--- индикаторные буферы
double         ColorCandlesBuffer1[];
double         ColorCandlesBuffer2[];
double         ColorCandlesBuffer3[];
double         ColorCandlesBuffer4[];
double         ColorCandlesColors[];
int            candles_colors;
//--- имя символа
string symbol;
//--- массив для хранения цветов содержит 14 элементов
color colors[]=
{
    clrRed,clrBlue,clrGreen,clrChocolate,clrMagenta,clrDodgerBlue,clrGoldenrod,
    clrIndigo,clrLightBlue,clrAliceBlue,clrMoccasin,clrMagenta,clrCyan,clrMediumPurple
};
//+-----+
//| Custom indicator initialization function |
//+-----+
int OnInit()
{
    //--- если bars слишком мало - досрочно завершаем работу
    if(bars<50)
    {
        Comment("Укажите большее количество баров! Работа индикатора прекращена");
        return(-1);
    }
    //--- indicator buffers mapping
    SetIndexBuffer(0,ColorCandlesBuffer1,INDICATOR_DATA);
    SetIndexBuffer(1,ColorCandlesBuffer2,INDICATOR_DATA);
    SetIndexBuffer(2,ColorCandlesBuffer3,INDICATOR_DATA);
    SetIndexBuffer(3,ColorCandlesBuffer4,INDICATOR_DATA);
    SetIndexBuffer(4,ColorCandlesColors,INDICATOR_COLOR_INDEX);
    //--- пустое значение
    PlotIndexSetDouble(0,PLOT_EMPTY_VALUE,0);
    //--- имя символа, по которому рисуются бары
    symbol=_Symbol;
    //--- установим отображение символа
    PlotIndexSetString(0,PLOT_LABEL,symbol+" Open;"+symbol+" High;"+symbol+" Low;"+symbol+" Close;");
    IndicatorSetString(INDICATOR_SHORTNAME,"DRAW_COLOR_CANDLES("+symbol+")");
    //---- количество цветов для раскраски свечей
    candles_colors=8; // см. комментарий к свойству #property indicator_color1
    //---

```

```

    return(0);
}

//+-----+
//| Custom indicator iteration function |
//+-----+

int OnCalculate(const int rates_total,
               const int prev_calculated,
               const datetime &time[],
               const double &open[],
               const double &high[],
               const double &low[],
               const double &close[],
               const long &tick_volume[],
               const long &volume[],
               const int &spread[])
{
    static int ticks=INT_MAX-100;
    //--- считаем тики для изменения стиля и цвета
    ticks++;
    //--- если накопилось достаточное число тиков
    if(ticks>=N)
    {
        //--- выберем новый символ из окна "Обзор рынка"
        symbol=GetRandomSymbolName();
        //--- сменим вид
        ChangeLineAppearance();
        //--- меняем цвета, которыми рисуются бары
        ChangeColors(colors,candles_colors);

        int tries=0;
        //--- сделаем 5 попыток заполнить буфера plot1 ценами из symbol
        while(!CopyFromSymbolToBuffers(symbol,rates_total,0,
            ColorCandlesBuffer1,ColorCandlesBuffer2,ColorCandlesBuffer3,
            ColorCandlesBuffer4,ColorCandlesColors,candles_colors)
            && tries<5)
        {
            //--- счетчик вызовов функции CopyFromSymbolToBuffers()
            tries++;
        }
        //--- сбрасываем счетчик тиков в ноль
        ticks=0;
    }
    //--- return value of prev_calculated for next call
    return(rates_total);
}

//+-----+
//| заполняет указанную свечу |
//+-----+

```

```

bool CopyFromSymbolToBuffers(string name,
                             int total,
                             int plot_index,
                             double &buff1[],
                             double &buff2[],
                             double &buff3[],
                             double &buff4[],
                             double &col_buffer[],
                             int cndl_colors
                             )
{
    //--- в массив rates[] будем копировать цены Open, High, Low и Close
    MqlRates rates[];
    //--- счетчик попыток
    int attempts=0;
    //--- сколько скопировано
    int copied=0;
    //--- делаем 25 попыток получить таймсерия по нужному символу
    while(attempts<25 && (copied=CopyRates(name,_Period,0,bars,rates))<0)
    {
        Sleep(100);
        attempts++;
        if(messages) PrintFormat("%s CopyRates(%s) attempts=%d",__FUNCTION__,name,attempts);
    }
    //--- если не удалось скопировать достаточное количество баров
    if(copied!=bars)
    {
        //--- сформируем строку сообщения
        string comm=StringFormat("Для символа %s удалось получить только %d баров из %d",
                                name,
                                copied,
                                bars
                                );
        //--- выведем сообщение в комментарий на главное окно графика
        Comment(comm);
        //--- выводим сообщения
        if(messages) Print(comm);
        return(false);
    }
    else
    {
        //--- установим отображение символа
        PlotIndexSetString(plot_index,PLOT_LABEL,name+" Open;" +name+" High;" +name+" Low
        IndicatorSetString(INDICATOR_SHORTNAME,"DRAW_COLOR_CANDLES (" +symbol+" )");
    }
    //--- инициализируем буфера пустыми значениями
    ArrayInitialize(buff1,0.0);
    ArrayInitialize(buff2,0.0);

```

```

    ArrayInitialize(buff3,0.0);
    ArrayInitialize(buff4,0.0);
//--- на каждом тике копируем цены в буферы
    for(int i=0;i<copied;i++)
    {
        //--- вычислим соответствующий индекс для буферов
        int buffer_index=total-copied+i;
        //--- записываем цены в буфера
        buff1[buffer_index]=rates[i].open;
        buff2[buffer_index]=rates[i].high;
        buff3[buffer_index]=rates[i].low;
        buff4[buffer_index]=rates[i].close;
        //--- зададим цвет свечи
        int color_index=i%cndl_colors;
        col_buffer[buffer_index]=color_index;
    }
    return(true);
}

//+-----+
//| возвращает случайным образом символ из Market Watch |
//+-----+
string GetRandomSymbolName()
{
    //--- количество символов, показываемых в окне "Обзор рынка"
    int symbols=SymbolsTotal(true);
    //--- позиция символа в списке - случайное число от 0 до symbols
    int number=MathRand()%symbols;
    //--- вернем имя символа по указанной позиции
    return SymbolName(number,true);
}

//+-----+
//| изменяет цвет отрезков свечей |
//+-----+
void ChangeColors(color &cols[],int plot_colors)
{
    //--- количество цветов
    int size=ArraySize(cols);
    //---
    string comm=ChartGetString(0,CHART_COMMENT)+"\r\n\r\n";

    //--- для каждого цветового индекса зададим новый цвет случайным образом
    for(int plot_color_ind=0;plot_color_ind<plot_colors;plot_color_ind++)
    {
        //--- получим случайное число
        int number=MathRand();
        //--- получим индекс в массиве col[] как остаток от целочисленного деления
        int i=number%size;
        //--- установим цвет для каждого индекса как свойство PLOT_LINE_COLOR

```



```

        PlotIndexSetInteger(0,          // номер графического стиля
                             PLOT_LINE_COLOR, // идентификатор свойства
                             plot_color_ind,  // индекс цвета, куда запишем цвет
                             cols[i]);        // новый цвет

    //--- запишем цвета
    comm=comm+StringFormat("CandleColorIndex[%d]=%s \r\n",plot_color_ind,ColorToStr
    ChartSetString(0,CHART_COMMENT,comm);
    }
//---
    }
//+-----+
//|  изменяет внешний вид свечей  |
//+-----+
void ChangeLineAppearance()
{
    //--- строка для формирования информации о свойствах свечей
    string comm="";
    //--- запишем имя символа
    comm="\r\n"+symbol+comm;
    //--- выведем информацию на график через комментарий
    Comment(comm);
}

```

指标属性和函数间的连接

每个自定义指标都有很多属性，有些是强制的，有些是开始描述的仓位，具有如下属性：

- 窗口指示到绘图指示 – indicator_separate_window 或者 indicator_chart_window；
- 指标缓冲区数量 – indicator_buffers；
- 指标绘图数量 – indicator_plots。

也可以通过 [预处理器](#) 指令或者建立自定义指标的函数建立其他属性。这些属性和类似函数在如下表格中描述。

指标子窗口属性指令	IndicatorSet...() 型函数	调整过的子窗口属性指令
indicator_height	IndicatorSetInteger (INDICATOR_INDICATOR_HEIGHT , nHeight)	
indicator_minimum	IndicatorSetDouble (INDICATOR_MINIMUM , dMaxValue)	纵轴最小值Minimal value of the vertical axis
indicator_maximum	IndicatorSetDouble (INDICATOR_MAXIMUM , dMinValue)	纵轴最大值
indicator_levelN	IndicatorSetDouble (INDICATOR_LEVELVALUE , N-1, nLevelValue)	N水平面纵轴值
no preprocessor directive	IndicatorSetString (INDICATOR_LEVELTEXT , N-1, sLevelName)	显示的水平面名称
indicator_levelcolor	IndicatorSetInteger (INDICATOR_LEVELCOLOR , N-1, nLevelColor)	N水平面颜色
indicator_levelwidth	IndicatorSetInteger (INDICATOR_LEVELWIDTH , N-1, nLevelWidth)	N平面的线宽
indicator_levelstyle	IndicatorSetInteger (INDICATOR_LEVELSTYLE , N-1, nLevelStyle)	N平面的线型
绘画属性指令	PlotIndexSet...() 型函数	调整过的绘画属性指令
indicator_labelN	PlotIndexSetString (N-1, PLOT_LABEL ,sLabel)	图N缩略名。当鼠标光标放在上面时显示在数据窗口和弹出提示框中
indicator_colorN	PlotIndexSetInteger (N-1, PLOT_LINE_COLOR , nColor)	图N线的颜色
indicator_styleN	PlotIndexSetInteger (N-1, PLOT_LINE_STYLE ,	图N的线型

	nType)	
indicator__typeN	PlotIndexSetInteger (N-1, PLOT_DRAW_TYPE , nType)	图N的线种
indicator__widthN	PlotIndexSetInteger (N-1, PLOT_LINE_WIDTH , nWidth)	图N的线宽
常用指标属性	IndicatorSet...() 型函数	描述
无预处理器指令	IndicatorSetString (INDICATOR_SHORTNAME , sShortName)	设置显示在指标列表（在程序端按Ctrl+I打开）中的指标的便捷短名称。
无预处理器指令	IndicatorSetInteger (INDICATOR_DIGITS , nDigits)	设置指标值显示所需精确性-小数位的数量。
无预处理器指令	IndicatorSetInteger (INDICATOR_LEVELS , nLevels)	设置指标窗口平面数
indicator__applied__price	无函数，属性仅可以通过预处理器指令设置。	用于指标计算的默认价格类型。只有使用OnCalculate() 时被指定。 属性值也可以在“参数”标签中“ 应用到 ”指标属性对话框中设置。

记录水平计算和预处理器项目绘图以1开始，而相同属性的技术使用函数以0开始，例如，少于1的指示值为#property。

有许多指令，没有类似函数：

指令	描述
indicator__chart__window	指标显示在主窗口中
indicator__separate__window	指标显示在独立子窗口中
indicator__buffers	所需指标缓冲区的数量
indicator__plots	指标中的 平面图 数量

SetIndexBuffer

该函数在**全局**水平使用一维动态 **双精度** 数组捆绑指定指标缓冲区。

```
bool SetIndexBuffer(
    int          index,          // 缓冲指数
    double       buffer[],       // 数组
    ENUM_INDEXBUFFER_TYPE data_type // 要存储的东西
);
```

参量

index

[in] 指标缓冲区的数字，编号从0开始，该数字在 `#property indicator_buffers` 中减少申报价格。

buffer[]

[in] 在自定义指标程序中表明一组数组。

data_type

[in] 数据存储类型在指标数组里。默认是 `INDICATOR_DATA`（计算指标的值），也可以使用 `INDICATOR_COLOR_INDEX` 值；在这种情况下缓冲区为先前指标缓冲区存储颜色指数。可以在 `#property indicator_colorN` 水平上指定64种颜色 - `INDICATOR_CALCULATIONS` 值表示缓冲器在指标媒介计算中使用，而不是用来绘画的。

返回值

如果成功，返回 `true`，否则 - `false`。

注释

在捆绑后，动态数组 `buffer[]` 将在共同数组中编入索引，尽管 **时间序列** 索引已经在界限数组中预先安装，如果想要转变指标数组元素的接入命令，在使用 `SetIndexBuffer()` 函数后捆绑数组使用 `ArraySetAsSeries()` 函数。注释表明通过 `SetIndexBuffer()`。函数不能改变建立指标缓冲区中的动态数组的大小，对于指标缓冲器来说，所有大小转变的操作都可以通过程序的子系统来执行。

示例：

```
//+-----+
//|                                     TestCopyBuffer1.mq5 |
//|                                     Copyright 2009, MetaQuotes Software Corp. |
//|                                     http://www.mql5.com |
//+-----+

#property copyright "2009, MetaQuotes Software Corp."
#property link      "http://www.mql5.com"
#property version   "1.00"

#property indicator_separate_window
#property indicator_buffers 1
#property indicator_plots   1
//---- 图 MA
#property indicator_label1  "MA"
#property indicator_type1   DRAW_LINE
```

```

#property indicator_color1 clrRed
#property indicator_style1 STYLE_SOLID
#property indicator_width1 1
//--- 输入参数
input bool          AsSeries=true;
input int           period=15;
input ENUM_MA_METHOD smootMode=MODE_EMA;
input ENUM_APPLIED_PRICE price=PRICE_CLOSE;
input int           shift=0;
//--- 指标缓冲
double             MABuffer[];
int                ma_handle;
//+-----+
//| 自定义指标初始化函数          |
//+-----+
int OnInit()
{
//--- 指标缓冲绘图
if(AsSeries) ArraySetAsSeries(MABuffer,true);
Print("Indicator buffer is timeseries = ",ArrayGetAsSeries(MABuffer));
SetIndexBuffer(0,MABuffer,INDICATOR_DATA);
Print("Indicator buffer after SetIndexBuffer() is timeseries = ",
      ArrayGetAsSeries(MABuffer));

//--- 更待指标缓冲访问单元命令
ArraySetAsSeries(MABuffer,AsSeries);

IndicatorSetString(INDICATOR_SHORTNAME,"MA("+period+")"+AsSeries);
//---
ma_handle=iMA(Symbol(),0,period,shift,smootMode,price);
return(0);
}
//+-----+
//| 自定义指标重复函数          |
//+-----+
int OnCalculate(const int rates_total,
               const int prev_calculated,
               const datetime &time[],
               const double &open[],
               const double &high[],
               const double &low[],
               const double &close[],
               const long &tick_volume[],
               const long &volume[],
               const int &spread[])
{
//--- 复制缓冲区 MABuffer 中的移动平均值
int copied=CopyBuffer(ma_handle,0,0,rates_total,MABuffer);

```

```
Print("MABuffer[0] = ",MABuffer[0]); // 取决于 AsSeries 值
                                     // 将收到旧值
                                     // 或者为当前未完成的字节

//--- 为下一次调用返回 prev_calculated 值
return(rates_total);
}
//+-----+
```

另见

[自定义指标属性](#), [访问时间序列和指标](#)

IndicatorSetDouble

函数建立了类似指标属性值，指标属性应该是双精度类型，有2个变量函数可以使用。

调用属性的指示标识符。

```
bool IndicatorSetDouble(  
    int      prop_id,           // 标识符  
    double   prop_value        // 将被设置的值  
);
```

调用指示标识符和属性修饰语。

```
bool IndicatorSetDouble(  
    int      prop_id,           // 标识符  
    int      prop_modifier,     // 修饰符  
    double   prop_value        // 将被设置的值  
)
```

参量

prop_id

[in] 值可以是 [ENUM_CUSTOMIND_PROPERTY_DOUBLE](#) 值中的一个。

prop_modifier

[in] 规定属性的修饰语，只有水平指示需要修饰语。

prop_value

[in] 属性值。

返回值

如果成功，返回 [true](#)，否则 - [false](#)。

IndicatorSetInteger

函数建立类似指标属性值，属性必须是整型或者颜色型的，有2个变量函数可以使用。

调用属性的指示标识符。

```
bool IndicatorSetInteger(  
    int prop_id,           // 标识符  
    int prop_value         // 将被设置的值  
);
```

调用指示标识符和属性修饰语。

```
bool IndicatorSetInteger(  
    int prop_id,           // 标识符  
    int prop_modifier,     // 修饰符  
    int prop_value         // 将被设置的值  
);
```

参量

prop_id

[in] 值可以是 [ENUM_CUSTOMIND_PROPERTY_INTEGER](#) 值中一个。

prop_modifier

[in] 指定属性修饰语，值有水平属性需要修饰语。

prop_value

[in] 属性值。

返回值

如果成功，返回 [true](#)，否则 - [false](#)。

IndicatorSetString

函数建立类似指标属性值，指标属性必须是字符串型。有2个变量函数可以使用。

调用属性的指示标识符

```
bool IndicatorSetString(  
    int      prop_id,           // 标识符  
    string   prop_value        // 将被设置的值  
);
```

调用指示标识符和属性修饰语

```
bool IndicatorSetString(  
    int      prop_id,           // 标识符  
    int      prop_modifier,     // 修饰符  
    string   prop_value        // 将被设置的值  
)
```

参量

prop_id

[in] 值可以是 [ENUM_CUSTOMIND_PROPERTY_STRING](#) 值中一个。

prop_modifier

[in] 指定属性标识符，只有水平属性需要修饰语。

prop_value

[in] 属性值。

返回值

如果成功，返回 [true](#)，否则 - [false](#)。

PlotIndexSetDouble

该函数建立一定特征的相应值，指标性质应该是双精度型。

```
bool PlotIndexSetDouble(  
    int      plot_index,    // 图样式指数  
    int      prop_id,       // 属性标识符  
    double   prop_value     // 将被设置的值  
);
```

参量

plot_index

[in] [图示](#) 的索引

prop_id

[in] 该值可以是 [ENUM_PLOT_PROPERTY_DOUBLE](#) 值中的一个。

prop_value

[in] 属性值。

返回值

如果成功，返回 [true](#)，否则 [false](#)。

PlotIndexSetInteger

该函数建立一定特征的相应值，指标性质应该是整型，图表型，布尔型或者颜色型。有两种变量函数。

调用指定的属性标识符。

```
bool PlotIndexSetInteger (
    int plot_index,      // 图样式指数
    int prop_id,         // 属性标识符
    int prop_value       // 将被设置的值
);
```

调用指定的属性标识符和修饰符。

```
bool PlotIndexSetInteger (
    int plot_index,      // 图样式指数
    int prop_id,         // 属性标识符
    int prop_modifier,   // 属性修饰符
    int prop_value       // 将被设置的值
)
```

参量

plot_index

[in] [图示](#) 的索引

prop_id

[in] 该值可以是 [ENUM_PLOT_PROPERTY_INTEGER](#) 值中的一个。

prop_modifier

[in] 规定属性的修饰符，只有水平属性需要修饰符。

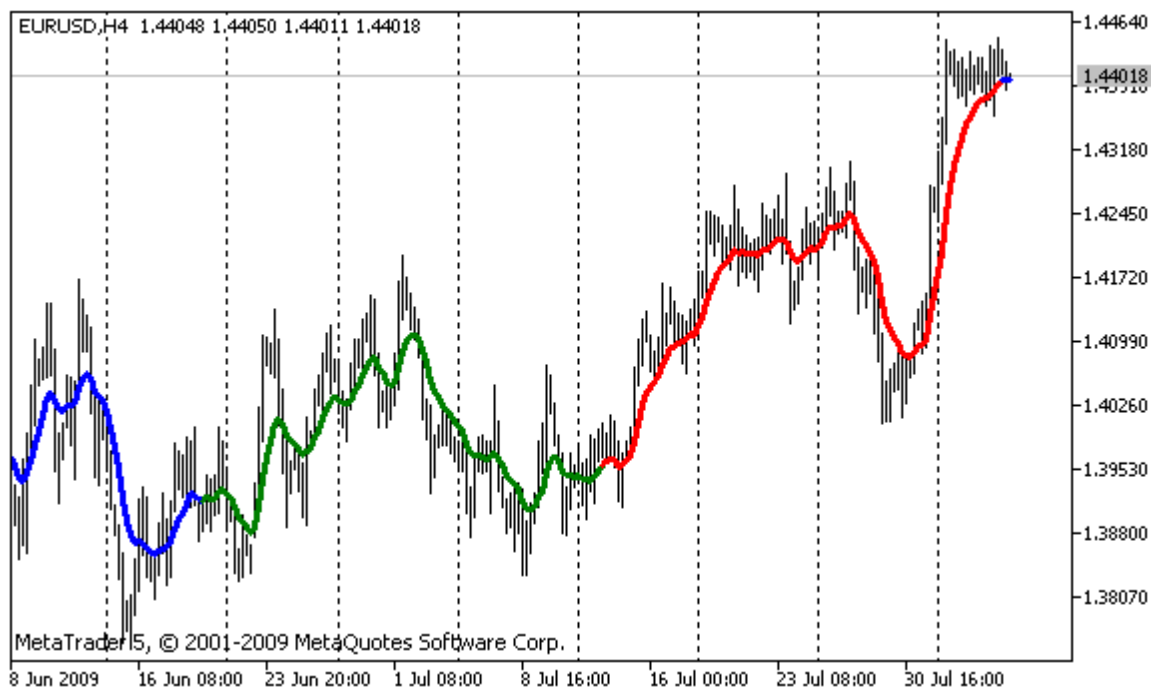
prop_value

[in] 属性值。

返回值

如果成功，返回 [true](#)，否则 [false](#)。

示例：这是画有三色线的指标。每五个节点改变一次颜色。



```
#property indicator_chart_window
#property indicator_buffers 2
#property indicator_plots 1
//---- 图线颜色
#property indicator_label1 "ColorLine"
#property indicator_type1 DRAW_COLOR_LINE
#property indicator_color1 clrRed,clrGreen,clrBlue
#property indicator_style1 STYLE_SOLID
#property indicator_width1 3
//--- 指标缓冲区
double ColorLineBuffer[];
double ColorBuffer[];
int MA_handle;
//+-----+
//| 自定义指标初始化函数 |
//+-----+
void OnInit()
{
//--- 指标缓冲绘图
SetIndexBuffer(0,ColorLineBuffer,INDICATOR_DATA);
SetIndexBuffer(1,ColorBuffer,INDICATOR_COLOR_INDEX);
//--- 获得MA处理器
MA_handle=iMA(Symbol(),0,10,0,MODE_EMA,PRICE_CLOSE);
//---
}
//+-----+
//| 获得颜色指数 |
//+-----+
```

```

int getIndexOfColor(int i)
{
    int j=i%300;
    if(j<100) return(0); // 第一指数
    if(j<200) return(1); // 第二指数
    return(2); // 第三指数
}

//+-----+
//| 自定义指标反复函数 |
//+-----+

int OnCalculate(const int rates_total,
                const int prev_calculated,
                const datetime &time[],
                const double &open[],
                const double &high[],
                const double &low[],
                const double &close[],
                const long &tick_volume[],
                const long &volume[],
                const int &spread[])
{
    //---
    static int ticks=0,modified=0;
    int limit;
    //--- 第一计算或者柱数被改变
    if(prev_calculated==0)
    {
        //--- 复制MA值到指标缓冲区 Col or LineBuffer
        int copied=CopyBuffer(MA_handle,0,0,rates_total,ColorLineBuffer);
        if(copied<=0) return(0); // 复制失败- 丢弃
        //--- 现在为每个柱设置线的颜色
        for(int i=0;i<rates_total;i++)
            ColorBuffer[i]=getIndexOfColor(i);
    }
    else
    {
        //--- 复制MA值到指标缓冲区 Col or LineBuffer
        int copied=CopyBuffer(MA_handle,0,0,rates_total,ColorLineBuffer);
        if(copied<=0) return(0);

        ticks++; // 数订单号
        if(ticks>=5) // 改变配色方案的时候
        {
            ticks=0; // 复位计数器
            modified++; // 颜色更改的计数器
            if(modified>=3) modified=0; // 复位计数器
            ResetLastError();
            switch(modified)

```

```

{
    case 0:// 第一配色方案
        PlotIndexSetInteger(0,PLOT_LINE_COLOR,0,clrRed);
        PlotIndexSetInteger(0,PLOT_LINE_COLOR,1,clrBlue);
        PlotIndexSetInteger(0,PLOT_LINE_COLOR,2,clrGreen);
        Print("Color scheme "+modified);
        break;
    case 1:// 第二配色方案
        PlotIndexSetInteger(0,PLOT_LINE_COLOR,0,clrYellow);
        PlotIndexSetInteger(0,PLOT_LINE_COLOR,1,clrPink);
        PlotIndexSetInteger(0,PLOT_LINE_COLOR,2,clrLightSlateGray);
        Print("Color scheme "+modified);
        break;
    default:// 第三配色方案
        PlotIndexSetInteger(0,PLOT_LINE_COLOR,0,clrLightGoldenrod);
        PlotIndexSetInteger(0,PLOT_LINE_COLOR,1,clrOrchid);
        PlotIndexSetInteger(0,PLOT_LINE_COLOR,2,clrLimeGreen);
        Print("Color scheme "+modified);
}
}
else
{
    //--- 设置初始位置
    limit=prev_calculated-1;
    //--- 现在为每柱设置线的颜色
    for(int i=limit;i<rates_total;i++)
        ColorBuffer[i]=getIndexOfColor(i);
}
}
//--- 为下一次调用返回prev_calculated值
return(rates_total);
}
//+-----+

```

PlotIndexSetString

The function sets the value of the corresponding property of the corresponding indicator line. The indicator property must be of the string type.

```
bool PlotIndexSetString(  
    int     plot_index,    // 图样式指数  
    int     prop_id,       // 属性标识符  
    string  prop_value     // 将被设置的值  
);
```

参量

plot_index

[in] [图示索引](#)

prop_id

[in] 该值可以是[ENUM_PLOT_PROPERTY_STRING](#) 列举值中的一个。

prop_value

[in] 属性值。

返回值

如果成功，返回 [true](#)，否则 [false](#)。

PlotIndexGetInteger

该函数建立一定特征的相应值，指标性质应该是整型，颜色型，布尔型或者字符型。有两种变量函数。

调用指定属性标识符。

```
int PlotIndexGetInteger(  
    int plot_index,      // 图样式指数  
    int prop_id,         // 属性标识符  
);
```

调用指定属性标识符和修饰符。

```
int PlotIndexGetInteger(  
    int plot_index,      // 绘图指数  
    int prop_id,         // 属性标识符  
    int prop_modifier    // 属性修饰符  
);
```

参量

plot_index

[in] [图解](#) 索引

prop_id

[in] 该值可以是 [ENUM_PLOT_PROPERTY_INTEGER](#) 列举值中的一个。

prop_modifier

[in] 规定属性修饰语，只有水平属性需要修饰语。

注释

函数用来提取适当指示线路的绘画设置，该函数与[PlotIndexSetInteger](#)函数串联，通过复制绘图性质从一条线绘制到另一条线里。

示例：指标的色彩平台以周为单位。每天的颜色都在程序中体现。



```
#property indicator_separate_window
#property indicator_buffers 5
#property indicator_plots 1
//---- 蜡烛颜色图
#property indicator_label1 "ColorCandles"
#property indicator_type1 DRAW_COLOR_CANDLES
#property indicator_style1 STYLE_SOLID
#property indicator_width1 1
//--- 指标缓冲区
double OpenBuffer[];
double HighBuffer[];
double LowBuffer[];
double CloseBuffer[];
double ColorCandlesColors[];
color ColorOfDay[6]={CLR_NONE,clrMediumSlateBlue,
                    clrDarkGoldenrod,clrForestGreen,clrBlueViolet,clrRed};

//+-----+
//| 自定义指标初始化函数 |
//+-----+
void OnInit()
{
//--- 指标缓冲区绘图
SetIndexBuffer(0,OpenBuffer,INDICATOR_DATA);
SetIndexBuffer(1,HighBuffer,INDICATOR_DATA);
SetIndexBuffer(2,LowBuffer,INDICATOR_DATA);
SetIndexBuffer(3,CloseBuffer,INDICATOR_DATA);
SetIndexBuffer(4,ColorCandlesColors,INDICATOR_COLOR_INDEX);
//--- 颜色缓存区设置颜色数
```

```

    PlotIndexSetInteger(0,PLOT_COLOR_INDEXES,6);
//--- 为颜色缓存设置颜色
    for(int i=1;i<6;i++)
        PlotIndexSetInteger(0,PLOT_LINE_COLOR,i,ColorOfDay[i]);
//--- 设置精确度
    IndicatorSetInteger(INDICATOR_DIGITS,_Digits);
    printf("We have %u colors of days",PlotIndexGetInteger(0,PLOT_COLOR_INDEXES));
//---
}
//+-----+
//| 自定义指标重复函数 |
//+-----+
int OnCalculate(const int rates_total,
               const int prev_calculated,
               const datetime &time[],
               const double &open[],
               const double &high[],
               const double &low[],
               const double &close[],
               const long &tick_volume[],
               const long &volume[],
               const int &spread[])
{
//---
    int i;
    MqlDateTime t;
//----
    if(prev_calculated==0) i=0;
    else i=prev_calculated-1;
//----
    while(i<rates_total)
    {
        OpenBuffer[i]=open[i];
        HighBuffer[i]=high[i];
        LowBuffer[i]=low[i];
        CloseBuffer[i]=close[i];
        //--- 为每个蜡烛设置颜色
        TimeToStruct(time[i],t);
        ColorCandlesColors[i]=t.day_of_week;
        //---
        i++;
    }
//--- 为下一次调用返回prev_calculated值
    return(rates_total);
}
//+-----+

```

物件函数

这是用于用有关特定图表的图解物件工作的函数组。

函数	功能
ObjectCreate	创建指定图表中指定类型物件
ObjectName	返回指定图表中相关类型物件名称（指定图表子窗口）
ObjectDelete	从指定图表移除指定名物件（从指定图表子窗口）
ObjectsDeleteAll	从指定图表移除指定类型所有物件（从指定图表子窗口）
ObjectFind	依据名称搜索指定ID物件
ObjectGetTimeByValue	对指定物件价格值返回时间值
ObjectGetValueByTime	对指定时间返回物件价格值
ObjectMove	改变指定物件定位点坐标
ObjectsTotal	返回指定图表中指定类型物件数（指定图表子窗口）
ObjectGetDouble	返回相应物件属性的双精度值
ObjectGetInteger	返回相应物件属性的整数型值
ObjectGetString	返回相应物件属性的字符串值
ObjectSetDouble	设置相应物件属性值
ObjectSetInteger	设置相应物件属性值
ObjectSetString	设置相应物件属性值

每个图表物件都应该有唯一的一个 [图表名称](#)，包括子窗口。图表物件名称的改变生成2个事件：物件删除旧名称事件，创建物件新名称事件。

在新建物件或者修改 [物件属性](#) 之后，推荐调用 [ChartRedraw\(\)](#) 函数，命令客户端强制绘图（[可见物件](#)仍涵盖在内）。

ObjectCreate

在制定图表子窗口中，函数创建指定名称，类型和原坐标物件，在创建期间可以指定到30个坐标。

```
bool ObjectCreate(
    long      chart_id,      // 图表标识符
    string     name,         // 物件名称
    int        type,         // 物件类型
    int        nwin,         // 窗口索引
    datetime   time1,        // 第一定位点的时间
    double     price1,       // 第一定位点的价格
    ...
    datetime   timeN=0,      // 第N个定位点的时间
    double     priceN=0,     //
    ...
    datetime   time30=0,     // 第30个定位点的时间
    double     price30=0     // 第30个定位点的价格
);
```

参量

chart_id

[in] 图表标识符。0代表当前图表。

name

[in] 物件名称。名称在图表中是唯一的，包括子窗口。

type

[in] 物件类型。值可以是 [ENUM_OBJECT](#) 值中的一个。

nwin

[in] 图表子窗口数量。0代表主图表窗口，指定子窗口一定存在，否则函数返回错误值。

time1

[in] 第一定位点的时间坐标。

price1

[in] 第一定位点的价格坐标。

timeN=0

[in] N点时间坐标。

priceN=0

[in] N点的价格坐标。

time30=0

[in] 第三十定位点的时间坐标。

price30=0

[in] 第三十定位点的价格坐标。

返回值

返回真值或者错误值取决于物件是否新建，为了获取更多[错误](#)调用 [GetLastError\(\)](#)。如果物件已经新建，函数会改变本身的坐标。

注释

图表子窗口的数量（如果图表中有附带指标的子窗口）以1开始，图表主窗口总是以0开始检索。

大量定位点（到30）提供更多使用，于此同时限制图表中的30个可能的定位点，调用的大量参量不能超过64。

当物件重命名，两个事件同时形成。这些事件可以在EA交易或者 [OnChartEvent\(\)](#) 函数指标中处理：

- 旧名称物件删除事件；
- 新名称物件新建事件。

ObjectName

函数返回指定图表中相关物件名称，在制定子窗口中的指定类型。

```
string ObjectName(  
    long   chart_id,      // 图表标识符  
    int     pos,          // 物件列表中的数量  
    int     nwin=-1,      // 窗口索引  
    int     type=-1       // 物件类型  
);
```

参量

chart_id

[in] 图表标识符。0表示当前图表。

pos

[in] 根据指定过滤器物件的序列号确定子窗口的类型和号码。

nwin=-1

[in] 图表子窗口的号码。0代表主图表窗口，-1代表所有图表子窗口，包括主窗口。

type=-1

[in] 物件类型。值可以是 [ENUM_OBJECT](#) 值中一个，-1表示全部类型。

返回值

如果成功返回物件名称。

注释

当物件重命名，两个事件同时形成。这些事件可以在EA交易或者 [OnChartEvent\(\)](#) 函数指标中处理：

- 旧名称物件删除事件；
- 新名称物件新建事件。

ObjectDelete

函数从指定图表中的指定名称里删除物件。

```
bool ObjectDelete(  
    long    chart_id,    // 图表标识符  
    string  name         // 物件名称  
);
```

参量

chart_id

[in] 图表标识符。0表示当前图表。

name

[in] 删除物件的名称。

返回值

如果成功移动返回真值，否则返回错误值，阅读更多关于 [错误](#) 信息，调用 [GetLastError\(\)](#)。

注释

当物件重命名，两个事件同时形成。这些事件可以在EA交易或者 [OnChartEvent\(\)](#) 函数指标中处理：

- 旧名称物件删除事件；
- 新名称物件新建事件。

ObjectsDeleteAll

从指定图表中删除所有物件，指定图表子窗口，指定类型。

```
int ObjectsDeleteAll(  
    long  chart_id,      // 图表标识符  
    int   nwin=-1,       // 窗口索引  
    int   type=-1        // 物件类型  
);
```

参量

chart_id

[in] 图表标识符。0代表当前图表。

nwin=-1

[in] 图表子窗口号码。0代表主要图表窗口，-1代表图表子窗口，包括主窗口。

type=-1

[in] 物件类型。值可以是 [ENUM_OBJECT](#) 值中一个，-1表示全部类型。

返回值

返回删除物件的数量。阅读更多关于 [错误](#) 信息，调用 [GetLastError\(\)](#)。

ObjectFind

在图表指定ID中函数搜索指定名称的物件。

```
int ObjectFind(  
    long    chart_id,    // 图表标识符  
    string  name         // 物件名称  
);
```

参量

chart_id

[in] 图表标识符，0表示当前图表。

name

[in] 搜索物件名称。

返回值

如果成功函数返回子窗口的号码（0表示图表的主窗口），发现物件。如果没有找到物件，函数返回负值，阅读更多关于[错误](#)，调用 [GetLastError\(\)](#)。

注释

当物件重命名，两个事件同时形成。这些事件可以在EA交易或者 [OnChartEvent\(\)](#) 函数指标中处理：

- 旧名称物件删除事件；
- 新名称物件新建事件。

ObjectGetTimeByValue

指定物件的指定价格值，函数返回时间值。

```
datetime ObjectGetTimeByValue(  
    long    chart_id,      // 图表标识符  
    string  name,          // 物件名称  
    double  value,         // 价格  
    int     line_id        // 线  
);
```

参量

chart_id

[in] 图表标识符。0表示当前图表。

name

[in] 物件名称。

value

[in] 价格值。

line_id

[in] 线路标识符。

返回值

为指定物件的指定价格值设定时间值。

注释

在价格坐标中的物件可以有許多值（椭圆，三角，凹形槽等），这种情况下，指定线形ID。

ObjectGetValueByTime

函数返回为指定物件指定时间值设定的价格值。

```
double ObjectGetValueByTime(  
    long      chart_id,      // 图表标识符  
    string     name,         // 物件名称  
    datetime  time,         // 时间  
    int       line_id        // 线  
);
```

参量

chart_id

[in] 图表标识符。0代表当前图表。

name

[in] 物件名称。

time

[in] 时间值。

line_id

[in] 线形ID。

返回值

指定物件的指定时间值的价格值。

注释

在时间坐标中的物件可以有許多值（椭圆，三角，凹形槽等），指定线形ID。

ObjectMove

函数物件定位点的指定坐标。

```
bool  ObjectMove(  
    long      chart_id,      // 图表标识符  
    string     name,         // 物件名称  
    int        point_index,   // 定位点数  
    datetime   time,         // 时间  
    double     price          // 价格  
);
```

参量

chart_id

[in] 图表标识符。0代表当前图表。

name

[in] 物件名称。

point_index

[in] 定位点检索。定位点数量取决于物件类型。

time

[in] 所选定位点的时间坐标。

price

[in] 所选定位点的价格坐标。

返回值

如果成功返回真值，否则返回错误值，阅读更多关于 [错误](#) 信息，调用 [GetLastError\(\)](#)。

ObjectsTotal

在指定类型的指定子窗口的指定图表物件中，函数返回物件数量。

```
int  ObjectsTotal(  
    long  chart_id,      // 图表标识符  
    int   nwin=-1,       // 窗口索引  
    int   type=-1        // 物件类型  
);
```

参量

chart_id

[in] 图表标识符。0表示当前图表。

nwin=-1

[in] 图表子窗口的数量。0代表主图表窗口，-1代表图表的所有子窗口，包括主窗口。

type=-1

[in] 物件类型。值可以是 [ENUM_OBJECT](#) 值中一个，-1表示全部类型。

返回值

物件数量。

ObjectSetDouble

函数建立类似物件属性的值。物件数据应该是 [双精度](#) 类型，有2中类型函数可供使用。

设置属性值，无修饰符

```
bool ObjectSetDouble(
    long    chart_id,      // 图表标识符
    string  name,          // 物件名称
    int     prop_id,       // 属性
    double  prop_value     // 值
);
```

设置表明修饰符属性值

```
bool ObjectSetDouble(
    long    chart_id,      // 图表标识符
    string  name,          // 物件名称
    int     prop_id,       // 属性
    int     prop_modifier, // 修饰符
    double  prop_value     // 值
);
```

参量

chart_id

[in] 图表标识符。0表示当前图表。

name

[in] 物件名称。

prop_id

[in] 物件属性ID。值可以是 [ENUM_OBJECT_PROPERTY_DOUBLE](#) 值中一个。

prop_modifier

[in] 指定属性修饰语，表示 [斐波纳契工具](#) 水平线数量和安德鲁分叉图解物件，水平线编号从0开始。

prop_value

[in] 属性值。

返回值

如果改变图解物件属性的命令成功发送到图表该函数返回true。否则返回false。调用函数[GetLastError\(\)](#) 读取更多关于[错误](#)信息。

新建斐波纳契物件和添加新水平线示例

```
//| 脚本程序启动函数 |
//+-----+
void OnStart()
{
//--- 辅助数组
    double high[],low[],price1,price2;
```

```

    datetime time[],time1,time2;
//--- 复制开盘价 - 最新的100柱就足够
    int copied=CopyHigh(Symbol(),0,0,100,high);
    if(copied<=0)
    {
        Print("Failed to copy the values of the High price series");
        return;
    }
//--- 复制收盘价 - 最新100柱就足够
    copied=CopyLow(Symbol(),0,0,100,low);
    if(copied<=0)
    {
        Print("Failed to copy the values of the Low price series");
        return;
    }
//--- 复制最新100柱的开盘时间
    copied=CopyTime(Symbol(),0,0,100,time);
    if(copied<=0)
    {
        Print("Failed to copy the values of the price series of Time");
        return;
    }
//--- 如访问时间序列一样组织访问复制的数据-反向
    ArraySetAsSeries(high,true);
    ArraySetAsSeries(low,true);
    ArraySetAsSeries(time,true);

//--- 斐波纳契物件的第一定位点的坐标
    price1=high[70];
    time1=time[70];
//--- 斐波纳契物件的第二定位点的坐标
    price2=low[50];
    time2=time[50];

//--- 创建斐波纳契物件的时间
    bool created=ObjectCreate(0,"Fibo",OBJ_FIBO,0,time1,price1,time2,price2);
    if(created) // 如果物件成功创建
    {
        //--- 设置斐波纳契色阶调整颜色
        ObjectSetInteger(0,"Fibo",OBJPROP_LEVELCOLOR,Blue);
        //--- 另外,有多少斐波纳契色阶?
        int levels=ObjectGetInteger(0,"Fibo",OBJPROP_LEVELS);
        Print("Fibo levels before = ",levels);
        //--- 输出到日志 => 水平号: 水平值描述
        for(int i=0;i<levels;i++)
        {
            Print(i," : ",ObjectGetDouble(0,"Fibo",OBJPROP_LEVELVALUE,i),
                " ",ObjectGetString(0,"Fibo",OBJPROP_LEVELTEXT,i));
        }
    }

```

```

    }
    //--- 增加每单元的水平面数量
    bool modified=ObjectSetInteger(0,"Fibo",OBJPROP_LEVELS,levels+1);
    if(!modified) // 更改水平面数量失败
    {
        Print("Failed to change the number of levels of Fibo, error ",GetLastError())
    }
    //--- 只是通知
    Print("Fibo levels after = ",ObjectGetInteger(0,"Fibo",OBJPROP_LEVELS));
    //--- 为新创建的水平面设置一个值
    bool added=ObjectSetDouble(0,"Fibo",OBJPROP_LEVELVALUE,levels,133);
    if(added) // 为水平面管理并设置一个值
    {
        Print("Successfully set one more Fibo level");
        //--- 还不要忘记设置水平面描述
        ObjectSetString(0,"Fibo",OBJPROP_LEVELTEXT,levels,"my level");
        ChartRedraw(0);
        //--- 在斐波纳契物件中获得水平面数量的实际值
        levels=ObjectGetInteger(0,"Fibo",OBJPROP_LEVELS);
        Print("Fibo levels after adding = ",levels);
        //--- 再次输出所有水平-只是为了证实
        for(int i=0;i<levels;i++)
        {
            Print(i,":",ObjectGetDouble(0,"Fibo",OBJPROP_LEVELVALUE,i),
                " ",ObjectGetString(0,"Fibo",OBJPROP_LEVELTEXT,i));
        }
    }
    else // 如果在斐波纳契物件中增加水平面数量则会失败
    {
        Print("Failed to set one more Fibo level. Error ",GetLastError());
    }
}
}

```

另见

[物件类型](#), [物件属性](#)

ObjectSetInteger

函数建立类似物件属性值，物件属性一定是[日期时间](#)，[整型](#)，[颜色](#)，[布尔或者字符](#) 类型。有2个变量函数可以使用。

设置属性值，无修饰符

```
bool ObjectSetInteger(
    long    chart_id,      // 图表标识符
    string  name,          // 物件名称
    int     prop_id,       // 属性
    long    prop_value     // 值
);
```

设置表明修饰符属性值

```
bool ObjectSetInteger(
    long    chart_id,      // 图表标识符
    string  name,          // 物件名称
    int     prop_id,       // 属性
    int     prop_modifier, // 修饰符
    long    prop_value     // 值
);
```

参量

chart_id

[in] 图表标识符，0代表当前图表。

name

[in] 物件名称。

prop_id

[in] 物件属性ID。值可以是 [ENUM_OBJECT_PROPERTY_INTEGER](#) 值中的一个。

prop_modifier

[in] 指定属性修饰语。表示 [斐波纳契工具](#) 水平线数量和安德鲁分叉图解物件，水平线编号从0开始。

prop_value

[in] 属性值。

返回值

如果改变图解物件属性的命令成功发送到图表该函数返回true。否则返回false。调用函数[GetLastError\(\)](#) 读取更多关于[错误](#)信息。

另见

[物件类型](#)， [物件属性](#)

ObjectSetString

函数建立类似于物件属性的值，物件属性必须是 [字符串](#) 类型。有2个变量函数可以使用。

设置属性值，无修饰符

```
bool ObjectSetString(
    long    chart_id,      // 图表标识符
    string  name,          // 物件名称
    int     prop_id,       // 属性
    string  prop_value     // 值
);
```

设置表明修饰符属性值

```
bool ObjectSetString(
    long    chart_id,      // 图表标识符
    string  name,          // 物件名称
    int     prop_id,       // 属性
    int     prop_modifier, // 修饰符
    string  prop_value     // 值
);
```

参量

chart_id

[in] 图表标识符。0代表当前图表。

name

[in] 物件名称。

prop_id

[in] 物件属性ID。值可以是 [ENUM_OBJECT_PROPERTY_STRING](#) 值中一个。

prop_modifier

[in] 指定属性修饰语，表示[斐波纳契工具](#) 水平线数量和安德鲁分叉图解物件，水平线编号从0开始。

prop_value

[in] 属性值。

返回值

如果改变图解物件属性的命令成功发送到图表该函数返回true。否则返回false。调用函数[GetLastError\(\)](#) 读取更多关于[错误](#)信息。

注释

当物件重命名，两个事件同时形成。这些事件可以在EA交易或者 [OnChartEvent\(\)](#) 函数指标中处理：

- 旧名称物件删除事件；
- 新名称物件新建事件。

ObjectGetDouble

函数返回类似物件属性的返回值，物件属性必须是[双精度](#)类型。有2个变量函数可以使用。

1. 即时返回属性值。

```
double ObjectGetDouble(
    long    chart_id,          // 图表标识符
    string  name,              // 物件名称
    int     prop_id,           // 属性标识符
    int     prop_modifier=0    // 属性修饰符，如果需要的话
);
```

2. 返回 true 或者 false，取决于函数是否成功。 如果成功，属性值通过上一参量以引用的方式传递安置接收变量。

```
bool ObjectGetDouble(
    long    chart_id,          // 图表标识符
    string  name,              // 物件名称
    int     prop_id,           // 属性标识符
    int     prop_modifier,     // 属性修饰符
    double& double_var         // 这里接受属性值
);
```

参量

chart_id

[in] 图表标识符。0代表当前图表。

name

[in] 物件名称。

prop_id

[in] 物件属性ID。值可以是 [ENUM_OBJECT_PROPERTY_DOUBLE](#) 值中一个。

prop_modifier

[in] 指定属性修饰语，第一变体，默认修饰语的值是0,大多属性不要求修饰语。表示[斐波纳契工具](#) 水平线数量和安德鲁分叉图解物件，水平线编号从0开始。

double_var

[out] 双精度类型变量接收需求属性值。

返回值

第一次调用变体的双精度类型返回值。

第二变体的函数返回真值，如果该属性是被维护的并且值可以放置到double_var 中，否则返回错误值。为了阅读更多关于[错误](#)，调用[GetLastError\(\)](#)。

ObjectGetInteger

函数返回类似物件属性值，物件属性必须是[日期时间](#)，[整型](#)，[颜色](#)，[布尔或者字符](#) 类型。有2个变量函数可以使用。

1. 即时返回属性值。

```
long ObjectGetInteger(
    long    chart_id,          // 图表标识符
    string  name,              // 物件名称
    int     prop_id,           // 属性标识符
    int     prop_modifier=0    // 属性修饰符，如果需要的话
);
```

2. 返回 true 或者 false，取决于函数是否成功。 如果成功，属性值通过上一参量以引用的方式传递安置接收变量。

```
bool ObjectGetInteger(
    long    chart_id,          // 图表标识符
    string  name,              // 物件名称
    int     prop_id,           // 属性标识符
    int     prop_modifier,     // 属性修饰符
    long&   long_var           // 这里接受属性值
);
```

参量

chart_id

[in] 图表标识符。0代表当前图表。

name

[in] 物件名称。

prop_id

[in] 物件属性ID。值可以是 [ENUM_OBJECT_PROPERTY_INTEGER](#) 值中的一个。

prop_modifier

[in] 指定属性修饰语，第一变体，默认修饰语的值是0。大多属性不要求修饰语。表示[斐波纳契工具](#) 水平线数量和安德鲁分叉图解物件，水平线编号从0开始。

long_var

[out] 长型变量接收要求属性值。

返回值

第一次调用变体的长类型返回值。

第二变体的函数返回真值，如果该属性是被维护的并且值可以放置到long_var 中，否则返回错误值。为了阅读更多关于 [错误](#)，调用 [GetLastError\(\)](#)。

ObjectGetString

函数返回类似物件属性值，物件属性必须是 [字符串](#) 类型，有2个变量函数可以使用。

1. 即时返回属性值。

```
string ObjectGetString(
    long    chart_id,          // 图表标识符
    string  name,              // 物件名称
    int     prop_id,           // 属性标识符
    int     prop_modifier=0    // 属性修饰符，如果需要的话
);
```

2. 返回 true 或者 false，取决于函数是否成功。 如果成功，属性值通过上一参量以引用的方式传递安置接收变量。

```
bool ObjectGetString(
    long    chart_id,          // 图表标识符
    string  name,              // 物件名称
    int     prop_id,           // 属性标识符
    int     prop_modifier,     // 属性修饰符
    string& string_var         // 这里接受属性值
);
```

参量

chart_id

[in] 图表标识符。0代表当前图表。

name

[in] 物件名称。

prop_id

[in] 物件属性ID。值可以是 [ENUM_OBJECT_PROPERTY_STRING](#) 值中的一个。

prop_modifier

[in] 指定属性修饰语，第一变体，默认修饰语的值是0,大多属性不要求修饰语。表示[斐波纳契工具](#) 水平线数量和Andrew's pitchfork图解物件，水平线编号从0开始。

string_var

[out] 字符串类型变量接收要求属性值。

返回值

第一次调用版本的字符串值。

第二变体的函数返回真值，如果该属性是被维护的并且值可以放置到string_var 中，否则返回错误值。为了阅读更多关于 [错误](#)，调用 [GetLastError\(\)](#)。

注释

当物件重命名，两个事件同时形成。这些事件可以在EA交易或者 [OnChartEvent\(\)](#) 函数指标中处理：

- 旧名称物件删除事件；

- 新名称物件新建事件。

技术指标函数

所有函数，像iMA, iAC, iMACD, ilchimoku等，在客户端全局缓存区建立类似技术指标的副本，如果类似参量的指标副本已经存在，新的副本就不会建立，只是参考存在的副本增添项目。

这些函数返回适当的指标副本处理，使用该处理系统，可以通过类似指标收到计算数据。类似缓冲区数据（技术指标包括内部缓冲区的计算数据1到5的不同取决于指标）可以使用 [CopyBuffer\(\)](#) 函数被MQL5程序复制。

不能参考新建的指标数据，因为指标计算值需要时间，因此最好在OnInit() 中建立指标处理器，函数 [iCustom\(\)](#) 创建类似自定义指标，返回的处理器就能成功建立。自定义指标包含最高512指标缓冲区，[CopyBuffer\(\)](#) 函数中包括目录，使用获得的处理器。

使用[IndicatorCreate\(\)](#) 函数创建任意计算指标是一种通用的办法，该函数以输入参量形式接收如下数据：

- 交易品种名称；
- 时间表；
- 创建的指标类型；
- 指标输入参量数量；
- MqlParam类型数组包含所有必要的输入参量。

计算机内存可以通过删除不使用的指标释放，使用[IndicatorRelease\(\)](#) 函数，传递指标处理器。

注释。在MQL5程序中反复以相同参量调用指标函数不能导致参考计算器的增加；计算器值能被1建立一次，推荐使用 [OnInit\(\)](#) 函数获得指标处理器，在其余函数中使用这些处理器，当MQL5程序初始化时，指标开始减少。

所有指标函数都有至少2个参量-交易品种和周期。交易品种 [NULL](#) 值代表当前交易品种，周期0值代表当前时间表。

函数	返回指标处理器：
iAC	加速震荡指标
iAD	累积/分配
iADX	平均定向指数
iADXWilder	威尔达平均定向指数
iAlligator	鳄鱼指标
iAMA	适合的移动平均数
iAO	动量震荡指标
iATR	平均真实区域
iBearsPower	熊市
iBands	布林带
iBullsPower	牛市
iCCI	商品通道指数
iChaikin	蔡金摆动指标
iCustom	自定义指标

<u>iDEMA</u>	双指数移动平均线
<u>iDeMarker</u>	指标
<u>iEnvelopes</u>	轨道线指标
<u>iForce</u>	强力指数
<u>iFractals</u>	分形学
<u>iFRAMA</u>	分形学适合移动平均数
<u>iGator</u>	鳄鱼振荡器
<u>ilchimoku</u>	一目均衡图
<u>iBWMFI</u>	威廉姆斯的市场便利指标
<u>iMomentum</u>	动量指标
<u>iMFI</u>	货币流量指标
<u>iMA</u>	移动平均数
<u>iOsMA</u>	移动平均振荡指标 (MACD柱状图)
<u>iMACD</u>	移动平均聚散指标
<u>iOBV</u>	平衡交易量
<u>iSAR</u>	抛物转向系统
<u>iRSI</u>	相对强弱指标
<u>iRVI</u>	相对活力指标
<u>iStdDev</u>	标准偏差
<u>iStochastic</u>	随机摆动指标
<u>iTEMA</u>	三倍指数移动平均数
<u>iTriX</u>	三倍指数移动平均数振荡指标
<u>iWPR</u>	威廉指数
<u>iVIDyA</u>	动态平均数便利指标
<u>iVolumes</u>	成交量

iAC

函数在客户端的全局缓存器建立 加速振荡指标并返回处理器，只有一个缓冲区。

```
int iAC(
    string      symbol,      // 交易品种名称
    ENUM_TIMEFRAMES period   // 周期
);
```

参量

symbol

[in] 证券交易品种名称，该数据应该用来计算指标。 [NULL](#) 值代表当前交易品种。

period

[in] 周期值可以是 [ENUM_TIMEFRAMES](#) 值中一个，0代表当前时间表。

返回值

返回特殊技术指标处理器，失败返回 [INVALID_HANDLE](#)。计算机内存从不使用的指标中释放，使用指标处理程序传递到的函数 [IndicatorRelease\(\)](#)。

:

```
//+-----+
//|                                     Demo_iAC.mq5 |
//|                                     Copyright 2011, MetaQuotes Software Corp. |
//|                                     http://www.mql5.com |
//+-----+
#property copyright "Copyright 2011, MetaQuotes Software Corp."
#property link      "http://www.mql5.com"
#property version   "1.00"
#property description "Индикатор демонстрирует как нужно получать данные"
#property description "индикаторных буферов для технического индикатора iAC."
#property description "Символ и таймфрейм, на котором рассчитывается индикатор,"
#property description "задаются параметрами symbol и period."
#property description "Способ создания хэндла задается параметром 'type' (тип функции"

#property indicator_separate_window
#property indicator_buffers 2
#property indicator_plots 1
//--- построение iAC
#property indicator_label1  "iAC"
#property indicator_type1   DRAW_COLOR_HISTOGRAM
#property indicator_color1  clrGreen, clrRed
#property indicator_style1  STYLE_SOLID
#property indicator_width1  1
//+-----+
//| перечисление способов создания хэндла |
//+-----+
enum Creation
```

```

{
    Call_iAC,                // использовать iAC
    Call_IndicatorCreate     // использовать IndicatorCreate
};

//--- входные параметры
input Creation              type=Call_iAC;           // тип функции
input string                symbol=" ";              // символ
input ENUM_TIMEFRAMES      period=PERIOD_CURRENT;   // таймфрейм
//--- индикаторные буферы
double iACBuffer[];
double iACColors[];
//--- переменная для хранения хэндла индикатора iAC
int    handle;
//--- переменная для хранения
string name=symbol;
//--- имя индикатора на графике
string short_name;
//--- будем хранить количество значений в индикаторе Accelerator Oscillator
int    bars_calculated=0;
//+-----+
//| Custom indicator initialization function |
//+-----+
int OnInit()
{
    //--- привязка массивов к индикаторным буферам
    SetIndexBuffer(0,iACBuffer,INDICATOR_DATA);
    SetIndexBuffer(1,iACColors,INDICATOR_COLOR_INDEX);
    //--- определимся с символом, на котором строится индикатор
    name=symbol;
    //--- удалим пробелы слева и справа
    StringTrimRight(name);
    StringTrimLeft(name);
    //--- если после этого длина строки name нулевая
    if(StringLen(name)==0)
    {
        //--- возьмем символ с графика, на котором запущен индикатор
        name=_Symbol;
    }
    //--- создадим хэндл индикатора
    if(type==Call_iAC)
        handle=iAC(name,period);
    else
        handle=IndicatorCreate(name,period,IND_AC);
    //--- если не удалось создать хэндл
    if(handle==INVALID_HANDLE)
    {
        //--- сообщим о неудаче и выведем номер ошибки
        PrintFormat("Не удалось создать хэндл индикатора iAC для пары %s/%s, код ошибки

```

```

        name,
        EnumToString(period),
        GetLastError());

    //--- работа индикатора завершается досрочно при возврате отрицательного значения
    return(-1);
}

//--- покажем на какой паре символ/таймфрейм рассчитан индикатор Accelerator Oscillator
short_name=StringFormat("iAC(%s/%s)",name,EnumToString(period));
IndicatorSetString(INDICATOR_SHORTNAME,short_name);
//--- нормальное выполнение инициализации индикатора
return(0);
}

//+-----+
//| Custom indicator iteration function |
//+-----+

int OnCalculate(const int rates_total,
               const int prev_calculated,
               const datetime &time[],
               const double &open[],
               const double &high[],
               const double &low[],
               const double &close[],
               const long &tick_volume[],
               const long &volume[],
               const int &spread[])
{
    //--- количество копируемых значений из индикатора iAC
    int values_to_copy;
    //--- узнаем количество рассчитанных значений в индикаторе
    int calculated=BarsCalculated(handle);
    if(calculated<=0)
    {
        PrintFormat("BarsCalculated() вернул %d, код ошибки %d",calculated,GetLastError());
        return(0);
    }
    //--- если это первый запуск вычислений нашего индикатора или изменилось количество значений
    //--- или если необходимо рассчитать индикатор для двух или более баров (значит что-то изменилось)
    if(prev_calculated==0 || calculated!=bars_calculated || rates_total>prev_calculated)
    {
        //--- если массив iACBuffer больше, чем значений в индикаторе iAC на паре symbol
        //--- в противном случае копировать будем меньше, чем размер индикаторных буферов
        if(calculated>rates_total) values_to_copy=rates_total;
        else
            values_to_copy=calculated;
    }
    else
    {
        //--- значит наш индикатор рассчитывается не в первый раз и с момента последнего запуска
        //--- для расчета добавилось не более одного бара

```

```

        values_to_copy=(rates_total-prev_calculated)+1;
    }
//--- заполняем массивы iACBuffer и iACColors значениями из индикатора Accelerator Os
//--- если FillArraysFromBuffer вернула false, значит данные не готовы - завершаем ра
    if(!FillArraysFromBuffer(iACBuffer,iACColors,handle,values_to_copy)) return(0);
//--- сформируем сообщение
    string comm=StringFormat("%s ==> Обновлено значений в индикаторе %s: %d",
                              TimeToString(TimeCurrent(),TIME_DATE|TIME_SECONDS),
                              short_name,
                              values_to_copy);
//--- выведем на график служебное сообщение
    Comment(comm);
//--- заппомним количество значений в индикаторе Accelerator Oscillator
    bars_calculated=calculated;
//--- вернем значение prev_calculated для следующего вызова
    return(rates_total);
}
//+-----+
//| Заполняем индикаторные буферы из индикатора iAC |
//+-----+
bool FillArraysFromBuffer(double &values[], // индикаторный буфер значений Асс
                          double &color_indexes[], // цветовой буфер (для хранения ин
                          int ind_handle, // хэндл индикатора iAC
                          int amount // количество копируемых значений
                          )
{
//--- сбросим код ошибки
    ResetLastError();
//--- заполняем часть массива iACBuffer значениями из индикаторного буфера под индекс
    if(CopyBuffer(ind_handle,0,0,amount,values)<0)
    {
        //--- если копирование не удалось, сообщим код ошибки
        PrintFormat("Не удалось скопировать данные из индикатора iAC, код ошибки %d",Ge
        //--- завершим с нулевым результатом - это означает, что индикатор будет считат
        return(false);
    }
//--- теперь копируем индексы цветов
    if(CopyBuffer(ind_handle,1,0,amount,color_indexes)<0)
    {
        //--- если копирование не удалось, сообщим код ошибки
        PrintFormat("Не удалось скопировать значения цветов из индикатора iAC, код ошиб
        //--- завершим с нулевым результатом - это означает, что индикатор будет считат
        return(false);
    }
//--- все получилось
    return(true);
}
//+-----+

```

```
///| Indicator deinitialization function |  
//+-----+  
void OnDeinit(const int reason)  
{  
//--- почистим график при удалении индикатора  
    Comment("");  
}
```

iAD

函数返回累积/分配指标处理器。只有一个缓冲区。

```
int iAD(
    string          symbol,          // 交易品种名称
    ENUM_TIMEFRAMES period,          // 周期
    ENUM_APPLIED_VOLUME applied_volume // 用于计算的交易量类型
);
```

参量

symbol

[in] 证券交易品种名称，数据用来计算指标。[NULL](#) 值表示当前交易品种。

period

[in] 周期值可以是 [ENUM_TIMEFRAMES](#) 值中一个，0代表当前时间表。

applied_volume

[in] 使用的交易量。可以是 [ENUM_APPLIED_VOLUME](#) 值中的一个。

返回值

返回特殊技术指标处理器，失败返回 [INVALID_HANDLE](#)。计算机内存从不使用的指标中释放，使用指标处理程序传递到的函数 [IndicatorRelease\(\)](#)。

:

```
//+-----+
//|                                     Demo_iAD.mq5 |
//|                                     Copyright 2011, MetaQuotes Software Corp. |
//|                                     http://www.mql5.com |
//+-----+

#property copyright "Copyright 2011, MetaQuotes Software Corp."
#property link      "http://www.mql5.com"
#property version   "1.00"
#property description "Индикатор демонстрирует как нужно получать данные"
#property description "индикаторных буферов для технического индикатора iAD."
#property description "Символ и таймфрейм, на котором рассчитывается индикатор,"
#property description "задаются параметрами symbol и period."
#property description "Способ создания хэндла задается параметром 'type' (тип функции"

#property indicator_separate_window
#property indicator_buffers 1
#property indicator_plots 1
//--- plot iAD
#property indicator_label1 "iAD"
#property indicator_type1  DRAW_LINE
#property indicator_color1  clrLightSeaGreen
#property indicator_style1  STYLE_SOLID
#property indicator_width1  1
```

```

//+-----+
//|  перечисление способов создания хэндла |
//+-----+
enum Creation
{
    Call_iAD,           // использовать iAD
    Call_IndicatorCreate // использовать IndicatorCreate
};
//--- входные параметры
input Creation      type=Call_iAD;           // тип функции
input ENUM_APPLIED_VOLUME volumes;           // используемый объем
input string        symbol=" ";              // символ
input ENUM_TIMEFRAMES period=PERIOD_CURRENT; // таймфрейм
//--- индикаторный буфер
double iADBuffer[];
//--- переменная для хранения хэндла индикатора iAD
int handle;
//--- переменная для хранения
string name=symbol;
//--- имя индикатора на графике
string short_name;
//--- будем хранить количество значений в индикаторе Accumulation/Distribution
int bars_calculated=0;
//+-----+
//| Custom indicator initialization function |
//+-----+
int OnInit()
{
    //--- привязка массива к индикаторному буферу
    SetIndexBuffer(0,iADBuffer,INDICATOR_DATA);
    //--- определимся с символом, на котором строится индикатор
    name=symbol;
    //--- удалим пробелы слева и справа
    StringTrimRight(name);
    StringTrimLeft(name);
    //--- если после этого длина строки name нулевая
    if(StringLen(name)==0)
    {
        //--- возьмем символ с графика, на котором запущен индикатор
        name=_Symbol;
    }
    //--- создадим хэндл индикатора
    if(type==Call_iAD)
        handle=iAD(name,period,volumes);
    else
    {
        //--- заполним структуру значениями параметров индикатора
        MqlParam pars[1];
    }
}

```

```

    pars[0].type=TYPE_INT;
    pars[0].integer_value=volumes;
    handle=IndicatorCreate(name,period,IND_AD,1,pars);
}

//--- если не удалось создать хэндл
if(handle==INVALID_HANDLE)
{
    //--- сообщим о неудаче и выведем номер ошибки
    PrintFormat("Не удалось создать хэндл индикатора iAD для пары %s/%s, код ошибки
                name,
                EnumToString(period),
                GetLastError());

    //--- работа индикатора завершается досрочно при возврате отрицательного значения
    return(-1);
}

//--- покажем на какой паре символ/таймфрейм рассчитан индикатор Accumulation/Distribution
short_name=StringFormat("iAD(%s/%s)",name,EnumToString(period));
IndicatorSetString(INDICATOR_SHORTNAME,short_name);
//--- нормальное выполнение инициализации индикатора
return(0);
}

//+-----+
//| Custom indicator iteration function |
//+-----+

int OnCalculate(const int rates_total,
                const int prev_calculated,
                const datetime &time[],
                const double &open[],
                const double &high[],
                const double &low[],
                const double &close[],
                const long &tick_volume[],
                const long &volume[],
                const int &spread[])
{
    //--- количество копируемых значений из индикатора iAD
    int values_to_copy;
    //--- узнаем количество рассчитанных значений в индикаторе
    int calculated=BarsCalculated(handle);
    if(calculated<=0)
    {
        PrintFormat("BarsCalculated() вернул %d, код ошибки %d",calculated,GetLastError());
        return(0);
    }

    //--- если это первый запуск вычислений нашего индикатора или изменилось количество э
    //--- или если необходимо рассчитать индикатор для двух или более баров (значит что-т
    if(prev_calculated==0 || calculated!=bars_calculated || rates_total>prev_calculate
    {

```



```

    //--- если массив iADBuffer больше, чем значений в индикаторе iAD на паре symbo
    //--- в противном случае копировать будем меньше, чем размер индикаторных буфер
    if(calculated>rates_total) values_to_copy=rates_total;
    else
        values_to_copy=calculated;
    }
else
{
    //--- значит наш индикатор рассчитывается не в первый раз и с момента последнег
    //--- для расчета добавилось не более одного бара
    values_to_copy=(rates_total-prev_calculated)+1;
}

//--- заполняем массив iADBuffer значениями из индикатора Accumulation/Distribution
//--- если FillArraysFromBuffer вернула false, значит данные не готовы - завершаем ра
    if(!FillArrayFromBuffer(iADBuffer,handle,values_to_copy)) return(0);
//--- сформируем сообщение
    string comm=StringFormat("%s ==> Обновлено значений в индикаторе %s: %d",
                               TimeToString(TimeCurrent(),TIME_DATE|TIME_SECONDS),
                               short_name,
                               values_to_copy);
//--- выведем на график служебное сообщение
    Comment(comm);
//--- заппомним количество значений в индикаторе Accumulation/Distribution
    bars_calculated=calculated;
//--- вернем значение prev_calculated для следующего вызова
    return(rates_total);
}

//+-----+
//| Заполняем индикаторные буферы из индикатора iAD |
//+-----+
bool FillArrayFromBuffer(double &values[], // индикаторный буфер линии Accumulation
                        int ind_handle, // хэндл индикатора iAD
                        int amount // количество копируемых значений
                        )
{
    //--- сбросим код ошибки
    ResetLastError();
    //--- заполняем часть массива iADBuffer значениями из индикаторного буфера под индекс
    if(CopyBuffer(ind_handle,0,0,amount,values)<0)
    {
        //--- если копирование не удалось, сообщим код ошибки
        PrintFormat("Не удалось скопировать данные из индикатора iAD, код ошибки %d",Ge
        //--- завершим с нулевым результатом - это означает, что индикатор будет считат
        return(false);
    }
    //--- все получилось
    return(true);
}

//+-----+

```

```
///| Indicator deinitialization function |  
//+-----+  
void OnDeinit(const int reason)  
{  
//--- почистим график при удалении индикатора  
    Comment("");  
}
```

iADX

函数返回平均定向移动指数指标处理器。

```
int iADX(
    string      symbol,      // 交易品种名称
    ENUM_TIMEFRAMES period,  // 周期
    int         adx_period   // 平均周期
);
```

参量

symbol

[in] 证券交易品种名称，数据用来计算指标。 [NULL](#) 值代表当前交易品种。

period

[in] 周期值可以是 [ENUM_TIMEFRAMES](#) 值中的一个，0代表当前时间表。

adx_period

[in] 周期计算指数。

返回值

返回特殊技术指标处理器，失败返回 [INVALID_HANDLE](#)。计算机内存从不使用的指标中释放，使用指标处理程序传递到的函数 [IndicatorRelease\(\)](#)。

注释

缓冲区代码如下：0 - MAIN_ LINE，1 - PLUSDI_ LINE，2 - MINUSDI_ LINE。

:

```
//+-----+
//|                                     Demo_iADX.mq5 |
//|                                     Copyright 2011, MetaQuotes Software Corp. |
//|                                     http://www.mql5.com |
//+-----+
#property copyright "Copyright 2011, MetaQuotes Software Corp."
#property link      "http://www.mql5.com"
#property version   "1.00"
#property description "Индикатор демонстрирует как нужно получать данные"
#property description "индикаторных буферов для технического индикатора iADX."
#property description "Символ и таймфрейм, на котором рассчитывается индикатор,"
#property description "задаются параметрами symbol и period."
#property description "Способ создания хэндла задается параметром 'type' (тип функции"

#property indicator_separate_window
#property indicator_buffers 3
#property indicator_plots   3
//--- plot ADX
#property indicator_label1  "ADX"
#property indicator_type1   DRAW_LINE
```

```

#property indicator_color1 clrLightSeaGreen
#property indicator_style1 STYLE_SOLID
#property indicator_width1 1
//--- plot DI_plus
#property indicator_label2 "DI_plus"
#property indicator_type2 DRAW_LINE
#property indicator_color2 clrYellowGreen
#property indicator_style2 STYLE_SOLID
#property indicator_width2 1
//--- plot DI_minus
#property indicator_label3 "DI_minus"
#property indicator_type3 DRAW_LINE
#property indicator_color3 clrWheat
#property indicator_style3 STYLE_SOLID
#property indicator_width3 1
//+-----+
//| перечисление способов создания хэнгла |
//+-----+
enum Creation
{
    Call_iADX,          // использовать iADX
    Call_IndicatorCreate // использовать IndicatorCreate
};
//--- входные параметры
input Creation      type=Call_iADX;          // тип функции
input int           adx_period=14;           // период расчета
input string        symbol=" ";              // символ
input ENUM_TIMEFRAMES period=PERIOD_CURRENT; // таймфрейм
//--- индикаторные буферы
double ADXBuffer[];
double DI_plusBuffer[];
double DI_minusBuffer[];
//--- переменная для хранения хэнгла индикатора iADX
int handle;
//--- переменная для хранения
string name=symbol;
//--- имя индикатора на графике
string short_name;
//--- будем хранить количество значений в индикаторе Average Directional Movement Ind
int bars_calculated=0;
//+-----+
//| Custom indicator initialization function |
//+-----+
int OnInit()
{
    //--- привязка массивов к индикаторным буферам
    SetIndexBuffer(0,ADXBuffer,INDICATOR_DATA);
    SetIndexBuffer(1,DI_plusBuffer,INDICATOR_DATA);

```

```

    SetIndexBuffer(2,DI_minusBuffer,INDICATOR_DATA);
//--- определимся с символом, на котором строится индикатор
    name=symbol;
//--- удалим пробелы слева и справа
    StringTrimRight(name);
    StringTrimLeft(name);
//--- если после этого длина строки name нулевая
    if(StringLen(name)==0)
    {
        //--- возьмем символ с графика, на котором запущен индикатор
        name=_Symbol;
    }
//--- создадим хэндл индикатора
    if(type==Call_iADX)
        handle=iADX(name,period,adx_period);
    else
    {
        //--- заполним структуру значениями параметров индикатора
        MqlParam pars[1];
        pars[0].type=TYPE_INT;
        pars[0].integer_value=adx_period;
        handle=IndicatorCreate(name,period,IND_ADX,1,pars);
    }
//--- если не удалось создать хэндл
    if(handle==INVALID_HANDLE)
    {
        //--- сообщим о неудаче и выведем номер ошибки
        PrintFormat("Не удалось создать хэндл индикатора iADX для пары %s/%s, код ошибки",
                    name,
                    EnumToString(period),
                    GetLastError());
        //--- работа индикатора завершается досрочно при возврате отрицательного значения
        return(-1);
    }
//--- покажем на какой паре символ/таймфрейм рассчитан индикатор Average Directional
    short_name=StringFormat("iADX(%s/%s period=%d)",name,EnumToString(period),adx_period);
    IndicatorSetString(INDICATOR_SHORTNAME,short_name);
//--- нормальное выполнение инициализации индикатора
    return(0);
}

//+-----+
//| Custom indicator iteration function |
//+-----+

int OnCalculate(const int rates_total,
                const int prev_calculated,
                const datetime &time[],
                const double &open[],
                const double &high[],

```

```

        const double &low[],
        const double &close[],
        const long &tick_volume[],
        const long &volume[],
        const int &spread[])

{
//--- количество копируемых значений из индикатора iADX
    int values_to_copy;
//--- узнаем количество рассчитанных значений в индикаторе
    int calculated=BarsCalculated(handle);
    if(calculated<=0)
    {
        PrintFormat("BarsCalculated() вернул %d, код ошибки %d",calculated,GetLastError);
        return(0);
    }
//--- если это первый запуск вычислений нашего индикатора или изменилось количество з
//--- или если необходимо рассчитать индикатор для двух или более баров (значит что-т
    if(prev_calculated==0 || calculated!=bars_calculated || rates_total>prev_calculate
    {
        //--- если массив ADXBuffer больше, чем значений в индикаторе iADX на паре symb
        //--- в противном случае копировать будем меньше, чем размер индикаторных буфер
        if(calculated>rates_total) values_to_copy=rates_total;
        else
            values_to_copy=calculated;
    }
    else
    {
        //--- значит наш индикатор рассчитывается не в первый раз и с момента последнег
        //--- для расчета добавилось не более одного бара
        values_to_copy=(rates_total-prev_calculated)+1;
    }
//--- заполняем массив значениями из индикатора Average Directional Movement Index
//--- если FillArraysFromBuffer вернула false, значит данные не готовы - завершаем ра
    if(!FillArraysFromBuffers(ADXBuffer,DI_plusBuffer,DI_minusBuffer,handle,values_to_
//--- сформируем сообщение
    string comm=StringFormat("%s ==> Обновлено значений в индикаторе %s: %d",
                               TimeToString(TimeCurrent(),TIME_DATE|TIME_SECONDS),
                               short_name,
                               values_to_copy);
//--- выведем на график служебное сообщение
    Comment(comm);
//--- запомним количество значений в индикаторе Average Directional Movement Index
    bars_calculated=calculated;
//--- вернем значение prev_calculated для следующего вызова
    return(rates_total);
}

//+-----+
//| Заполняем индикаторные буферы из индикатора iADX |
//+-----+

```

```

bool FillArraysFromBuffers(double &adx_values[],      // индикаторный буфер линии ADX
                           double &DIplus_values[],  // индикаторный буфер для DI+
                           double &DIminus_values[], // индикаторный буфер для DI-
                           int ind_handle,           // хэндл индикатора iADXWilder
                           int amount                // количество копируемых значений
                           )
{
    //--- сбросим код ошибки
    ResetLastError();
    //--- заполняем часть массива ADXBuffer значениями из индикаторного буфера под индекс
    if(CopyBuffer(ind_handle,0,0,amount,adx_values)<0)
    {
        //--- если копирование не удалось, сообщим код ошибки
        PrintFormat("Не удалось скопировать данные из индикатора iADX, код ошибки %d",G
        //--- завершим с нулевым результатом - это означает, что индикатор будет считат
        return(false);
    }

    //--- заполняем часть массива DI_plusBuffer значениями из индикаторного буфера под ин
    if(CopyBuffer(ind_handle,1,0,amount,DIplus_values)<0)
    {
        //--- если копирование не удалось, сообщим код ошибки
        PrintFormat("Не удалось скопировать данные из индикатора iADX, код ошибки %d",G
        //--- завершим с нулевым результатом - это означает, что индикатор будет считат
        return(false);
    }

    //--- заполняем часть массива DI_plusBuffer значениями из индикаторного буфера под ин
    if(CopyBuffer(ind_handle,2,0,amount,DIminus_values)<0)
    {
        //--- если копирование не удалось, сообщим код ошибки
        PrintFormat("Не удалось скопировать данные из индикатора iADX, код ошибки %d",G
        //--- завершим с нулевым результатом - это означает, что индикатор будет считат
        return(false);
    }

    //--- все получилось
    return(true);
}

//+-----+
//| Indicator deinitialization function |
//+-----+
void OnDeinit(const int reason)
{
    //--- почистим график при удалении индикатора
    Comment("");
}

```

iADXWilder

函数返回威尔达平均定向移动指数处理器。

```
int iADXWilder(
    string      symbol,      // 交易品种名称
    ENUM_TIMEFRAMES period,  // 周期
    int         adx_period   // 平均周期
);
```

参量

symbol

[in] 证券交易品种名称，数据用来计算指标。 [NULL](#) 值代表当前交易品种。

period

[in] 周期值可以是 [ENUM_TIMEFRAMES](#) 值中的一个，0代表当前时间表。

adx_period

[in] 周期计算指数。

返回值

返回特殊技术指标处理器，失败返回 [INVALID_HANDLE](#)。计算机内存从不使用的指标中释放，使用指标处理程序传递到的函数 [IndicatorRelease\(\)](#)。

注释

缓冲区代码如下：0 - MAIN_ LINE，1 - PLUSDI_ LINE，2 - MINUSDI_ LINE。

:

```
//+-----+
//|                                     Demo_iAlligator.mq5 |
//|                                     Copyright 2011, MetaQuotes Software Corp. |
//|                                     http://www.mql5.com |
//+-----+

#property copyright "Copyright 2011, MetaQuotes Software Corp."
#property link      "http://www.mql5.com"
#property version   "1.00"
#property description "Индикатор демонстрирует как нужно получать данные"
#property description "индикаторных буферов для технического индикатора iAlligator."
#property description "Символ и таймфрейм, на котором рассчитывается индикатор,"
#property description "задаются параметрами symbol и period."
#property description "Способ создания хэндла задается параметром 'type' (тип функции"
#property description "Все остальные параметры как в стандартном Аллигаторе."

#property indicator_chart_window
#property indicator_buffers 3
#property indicator_plots   3
//--- plot Jaws
#property indicator_label1  "Jaws"
```



```

#property indicator_type1   DRAW_LINE
#property indicator_color1  clrBlue
#property indicator_style1  STYLE_SOLID
#property indicator_width1  1
//--- plot Teeth
#property indicator_label2  "Teeth"
#property indicator_type2   DRAW_LINE
#property indicator_color2  clrRed
#property indicator_style2  STYLE_SOLID
#property indicator_width2  1
//--- plot Lips
#property indicator_label3  "Lips"
#property indicator_type3   DRAW_LINE
#property indicator_color3  clrLime
#property indicator_style3  STYLE_SOLID
#property indicator_width3  1
//+-----+
//|  перечисление способов создания хэндла  |
//+-----+
enum Creation
{
    Call_iAlligator,      // использовать iAlligator
    Call_IndicatorCreate  // использовать IndicatorCreate
};
//--- входные параметры
input Creation          type=Call_iAlligator;  // тип функции
input string            symbol=" ";            // символ
input ENUM_TIMEFRAMES  period=PERIOD_CURRENT; // таймфрейм
input int               jaw_period=13;         // период для линии Челюстей
input int               jaw_shift=8;           // смещение линии Челюстей
input int               teeth_period=8;        // период для линии Зубов
input int               teeth_shift=5;         // смещение линии Челюстей
input int               lips_period=5;         // период для динии Губ
input int               lips_shift=3;          // смещение линии Губ
input ENUM_MA_METHOD    MA_method=MODE_SMA;   // метод усреднения линий Аллигатор
input ENUM_APPLIED_PRICE applied_price=PRICE_MEDIAN; // тип цены, от которой строится
//--- индикаторные буферы
double      JawsBuffer[];
double      TeethBuffer[];
double      LipsBuffer[];
//--- переменная для хранения хэндла индикатора iAlligator
int         handle;
//--- переменная для хранения
string name=symbol;
//--- имя индикатора на графике
string short_name;
//--- будем хранить количество значений в индикаторе Alligator
int      bars_calculated=0;

```

```

//+-----+
//| Custom indicator initialization function |
//+-----+
int OnInit()
{
//--- привязка массивов к индикаторным буферам
    SetIndexBuffer(0,JawsBuffer,INDICATOR_DATA);
    SetIndexBuffer(1,TeethBuffer,INDICATOR_DATA);
    SetIndexBuffer(2,LipsBuffer,INDICATOR_DATA);
//--- зададим смещение для каждой линии
    PlotIndexSetInteger(0,PLOT_SHIFT,jaw_shift);
    PlotIndexSetInteger(1,PLOT_SHIFT,teeth_shift);
    PlotIndexSetInteger(2,PLOT_SHIFT,lips_shift);
//--- определимся с символом, на котором строится индикатор
    name=symbol;
//--- удалим пробелы слева и справа
    StringTrimRight(name);
    StringTrimLeft(name);
//--- если после этого длина строки name нулевая
    if(StringLen(name)==0)
    {
        //--- возьмем символ с графика, на котором запущен индикатор
        name=_Symbol;
    }
//--- создадим хэндл индикатора
    if(type==Call_iAlligator)
        handle=iAlligator(name,period,jaw_period,jaw_shift,teeth_period,
                           teeth_shift,lips_period,lips_shift,MA_method,applied_price);
    else
    {
        //--- заполним структуру значениями параметров индикатора
        MqlParam pars[8];
        //--- периоды и смещения линий Аллигатора
        pars[0].type=TYPE_INT;
        pars[0].integer_value=jaw_period;
        pars[1].type=TYPE_INT;
        pars[1].integer_value=jaw_shift;
        pars[2].type=TYPE_INT;
        pars[2].integer_value=teeth_period;
        pars[3].type=TYPE_INT;
        pars[3].integer_value=teeth_shift;
        pars[4].type=TYPE_INT;
        pars[4].integer_value=lips_period;
        pars[5].type=TYPE_INT;
        pars[5].integer_value=lips_shift;
//--- тип сглаживания
        pars[6].type=TYPE_INT;
        pars[6].integer_value=MA_method;
    }
}

```

```

//--- тип цены
pars[7].type=TYPE_INT;
pars[7].integer_value=applied_price;
//--- создадим хэндл
handle=IndicatorCreate(name,period,IND_ALLIGATOR,8,pars);
}
//--- если не удалось создать хэндл
if(handle==INVALID_HANDLE)
{
    //--- сообщим о неудаче и выведем номер ошибки
    PrintFormat("Не удалось создать хэндл индикатора iAlligator для пары %s/%s, код
                name,
                EnumToString(period),
                GetLastError());
    //--- работа индикатора завершается досрочно при возврате отрицательного значения
    return(-1);
}
//--- покажем на какой паре символ/таймфрейм рассчитан индикатор Alligator
short_name=StringFormat("iAlligator(%s/%s, %d,%d,%d,%d,%d,%d)",name,EnumToString(p
                jaw_period,jaw_shift,teeth_period,teeth_shift,lips_period,
    IndicatorSetString(INDICATOR_SHORTNAME,short_name);
//--- нормальное выполнение инициализации индикатора
return(0);
}
//+-----+
//| Custom indicator iteration function |
//+-----+
int OnCalculate(const int rates_total,
                const int prev_calculated,
                const datetime &time[],
                const double &open[],
                const double &high[],
                const double &low[],
                const double &close[],
                const long &tick_volume[],
                const long &volume[],
                const int &spread[])
{
    //--- количество копируемых значений из индикатора iAlligator
    int values_to_copy;
    //--- узнаем количество рассчитанных значений в индикаторе
    int calculated=BarsCalculated(handle);
    if(calculated<=0)
    {
        PrintFormat("BarsCalculated() вернул %d, код ошибки %d",calculated,GetLastError
        return(0);
    }
    //--- если это первый запуск вычислений нашего индикатора или изменилось количество з

```

```

//--- или если необходимо рассчитать индикатор для двух или более баров (значит что-т
if(prev_calculated==0 || calculated!=bars_calculated || rates_total>prev_calculate
{
    //--- если массив JawsBuffer больше, чем значений в индикаторе iAlligator на па
    //--- в противном случае копировать будем меньше, чем размер индикаторных буфер
    if(calculated>rates_total) values_to_copy=rates_total;
    else
        values_to_copy=calculated;
}
else
{
    //--- значит наш индикатор рассчитывается не в первый раз и с момента последнег
    //--- для расчета добавилось не более одного бара
    values_to_copy=(rates_total-prev_calculated)+1;
}
//--- заполняем массивы значениями из индикатора Alligator
//--- если FillArraysFromBuffer вернула false, значит данные не готовы - завершаем ра
if(!FillArraysFromBuffers(JawsBuffer,jaw_shift,TeethBuffer,teeth_shift,LipsBuffer,
//--- сформируем сообщение
string comm=StringFormat("%s ==> Обновлено значений в индикаторе %s: %d",
                        TimeToString(TimeCurrent(),TIME_DATE|TIME_SECONDS),
                        short_name,
                        values_to_copy);
//--- выведем на график служебное сообщение
Comment(comm);
//--- запомним количество значений в индикаторе Alligator
bars_calculated=calculated;
//--- вернем значение prev_calculated для следующего вызова
return(rates_total);
}
//+-----+
//| Заполняем индикаторные буферы из индикатора iAlligator |
//+-----+
bool FillArraysFromBuffers(double &jaws_buffer[], // индикаторный буфер для линии Че
                        int j_shift, // смещение линии Челюстей
                        double &teeth_buffer[], // индикаторный буфер для линии Зу
                        int t_shift, // смещение линии Зубов
                        double &lips_buffer[], // индикаторный буфер для линии Гу
                        int l_shift, // смещение линии Губ
                        int ind_handle, // хэндл индикатора iAlligator
                        int amount // количество копируемых значений
                        )
{
    //--- сбросим код ошибки
    ResetLastError();
    //--- заполняем часть массива JawsBuffer значениями из индикаторного буфера под индек
    if(CopyBuffer(ind_handle,0,-j_shift,amount,jaws_buffer)<0)
    {
        //--- если копирование не удалось, сообщим код ошибки

```

```

        PrintFormat("Не удалось скопировать данные из индикатора iAlligator, код ошибки");
        //--- завершим с нулевым результатом - это означает, что индикатор будет считат
        return(false);
    }

    //--- заполняем часть массива TeethBuffer значениями из индикаторного буфера под инде
    if(CopyBuffer(ind_handle,1,-t_shift,amount,teeth_buffer)<0)
    {
        //--- если копирование не удалось, сообщим код ошибки
        PrintFormat("Не удалось скопировать данные из индикатора iAlligator, код ошибки");
        //--- завершим с нулевым результатом - это означает, что индикатор будет считат
        return(false);
    }

    //--- заполняем часть массива LipsBuffer значениями из индикаторного буфера под индекс
    if(CopyBuffer(ind_handle,2,-l_shift,amount,lips_buffer)<0)
    {
        //--- если копирование не удалось, сообщим код ошибки
        PrintFormat("Не удалось скопировать данные из индикатора iAlligator, код ошибки");
        //--- завершим с нулевым результатом - это означает, что индикатор будет считат
        return(false);
    }

    //--- все получилось
    return(true);
}

//+-----+
//| Indicator deinitialization function |
//+-----+
void OnDeinit(const int reason)
{
    //--- почистим график при удалении индикатора
    Comment("");
}

```

iAlligator

函数返回鳄鱼指标处理器。

```
int iAlligator(
    string          symbol,           // 交易品种名称
    ENUM_TIMEFRAMES period,          // 周期
    int             jaw_period,       // 咽喉计算周期
    int             jaw_shift,        // 咽喉平移
    int             teeth_period,     // 牙齿计算周期
    int             teeth_shift,      // 牙齿平移
    int             lips_period,      // 唇部计算周期
    int             lips_shift,       // 唇部平移
    ENUM_MA_METHOD  ma_method,       // 平滑类型
    ENUM_APPLIED_PRICE applied_price // 价格或者处理器类型
);
```

参量

symbol

[in] 证券交易品种名称，数据用来计算指标。 [NULL](#) 值代表当前交易品种。

period

[in] 周期值可以是 [ENUM_TIMEFRAMES](#) 值中的一个，0代表当前时间表。

jaw_period

[in] 蓝线的平均周期(鳄鱼颌骨)

jaw_shift

[in] 关于价格图表的蓝线转换。

teeth_period

[in] 红线的平均周期(鳄鱼牙) 。

teeth_shift

[in] 关于价格图表的红线转换。

lips_period

[in] 绿线的平均周期 (鳄鱼唇部) 。

lips_shift

[in] 关于价格图表的绿线转换。

ma_method

[in] 求平均值的方式，可以是 [ENUM_MA_METHOD](#) 值中任意值。

applied_price

[in] 使用价格。可以是任意 [ENUM_APPLIED_PRICE](#) 价格常量或者另外指标处理器。

返回值

返回特殊技术指标处理器，失败返回 [INVALID_HANDLE](#)。计算机内存从不使用的指标中释放，使用指标处理

程序传递到的函数 [IndicatorRelease\(\)](#) 。

注释

缓冲区代码如下：0 - GATORJAW_ LINE , 1 - GATORTEETH_ LINE , 2 - GATORLIPS_ LINE。

:

```
//+-----+
//|                                     Demo_iAlligator.mq5 |
//|                                     Copyright 2011, MetaQuotes Software Corp. |
//|                                     http://www.mql5.com |
//+-----+

#property copyright "Copyright 2011, MetaQuotes Software Corp."
#property link      "http://www.mql5.com"
#property version   "1.00"
#property description "Индикатор демонстрирует как нужно получать данные"
#property description "индикаторных буферов для технического индикатора iAlligator."
#property description "Символ и таймфрейм, на котором рассчитывается индикатор,"
#property description "задаются параметрами symbol и period."
#property description "Способ создания хэндла задается параметром 'type' (тип функции"
#property description "Все остальные параметры как в стандартном Аллигаторе."

#property indicator_chart_window
#property indicator_buffers 3
#property indicator_plots 3
//--- plot Jaws
#property indicator_label1  "Jaws"
#property indicator_type1   DRAW_LINE
#property indicator_color1  clrBlue
#property indicator_style1  STYLE_SOLID
#property indicator_width1  1
//--- plot Teeth
#property indicator_label2  "Teeth"
#property indicator_type2   DRAW_LINE
#property indicator_color2  clrRed
#property indicator_style2  STYLE_SOLID
#property indicator_width2  1
//--- plot Lips
#property indicator_label3  "Lips"
#property indicator_type3   DRAW_LINE
#property indicator_color3  clrLime
#property indicator_style3  STYLE_SOLID
#property indicator_width3  1
//+-----+
//|  перечисление способов создания хэндла |
//+-----+

enum Creation
{
    Call_iAlligator, // использовать iAlligator
```

```

    Call_IndicatorCreate    // использовать IndicatorCreate
};
//--- входные параметры
input Creation            type=Call_iAlligator;    // тип функции
input string              symbol=" ";              // символ
input ENUM_TIMEFRAMES     period=PERIOD_CURRENT;  // таймфрейм
input int                 jaw_period=13;           // период для линии Челюстей
input int                 jaw_shift=8;             // смещение линии Челюстей
input int                 teeth_period=8;          // период для линии Зубов
input int                 teeth_shift=5;           // смещение линии Челюстей
input int                 lips_period=5;           // период для динии Губ
input int                 lips_shift=3;            // смещение линии Губ
input ENUM_MA_METHOD       MA_method=MODE_SMMA;    // метод усреднения линий Аллигатор
input ENUM_APPLIED_PRICE   applied_price=PRICE_MEDIAN; // тип цены, от которой строится
//--- индикаторные буферы
double                    JawsBuffer[];
double                    TeethBuffer[];
double                    LipsBuffer[];
//--- переменная для хранения хэндла индикатора iAlligator
int    handle;
//--- переменная для хранения
string name=symbol;
//--- имя индикатора на графике
string short_name;
//--- будем хранить количество значений в индикаторе Alligator
int    bars_calculated=0;
//+-----+
//| Custom indicator initialization function |
//+-----+
int OnInit()
{
    //--- привязка массивов к индикаторным буферам
    SetIndexBuffer(0,JawsBuffer,INDICATOR_DATA);
    SetIndexBuffer(1,TeethBuffer,INDICATOR_DATA);
    SetIndexBuffer(2,LipsBuffer,INDICATOR_DATA);
    //--- зададим смещение для каждой линии
    PlotIndexSetInteger(0,PLOT_SHIFT,jaw_shift);
    PlotIndexSetInteger(1,PLOT_SHIFT,teeth_shift);
    PlotIndexSetInteger(2,PLOT_SHIFT,lips_shift);
    //--- определимся с символом, на котором строится индикатор
    name=symbol;
    //--- удалим пробелы слева и справа
    StringTrimRight(name);
    StringTrimLeft(name);
    //--- если после этого длина строки name нулевая
    if(StringLen(name)==0)
    {
        //--- возьмем символ с графика, на котором запущен индикатор

```



```

        name=_Symbol;
    }
//--- создадим хэндл индикатора
    if(type==Call_iAlligator)
        handle=iAlligator(name,period,jaw_period,jaw_shift,teeth_period,
                           teeth_shift,lips_period,lips_shift,MA_method,applied_price);
    else
    {
        //--- заполним структуру значениями параметров индикатора
        MqlParam pars[8];
        //--- периоды и смещения линий Аллигатора
        pars[0].type=TYPE_INT;
        pars[0].integer_value=jaw_period;
        pars[1].type=TYPE_INT;
        pars[1].integer_value=jaw_shift;
        pars[2].type=TYPE_INT;
        pars[2].integer_value=teeth_period;
        pars[3].type=TYPE_INT;
        pars[3].integer_value=teeth_shift;
        pars[4].type=TYPE_INT;
        pars[4].integer_value=lips_period;
        pars[5].type=TYPE_INT;
        pars[5].integer_value=lips_shift;
//--- тип сглаживания
        pars[6].type=TYPE_INT;
        pars[6].integer_value=MA_method;
//--- тип цены
        pars[7].type=TYPE_INT;
        pars[7].integer_value=applied_price;
//--- создадим хэндл
        handle=IndicatorCreate(name,period,IND_ALLIGATOR,8,pars);
    }
//--- если не удалось создать хэндл
    if(handle==INVALID_HANDLE)
    {
        //--- сообщим о неудаче и выведем номер ошибки
        PrintFormat("Не удалось создать хэндл индикатора iAlligator для пары %s/%s, код
                     name,
                     EnumToString(period),
                     GetLastError());
        //--- работа индикатора завершается досрочно при возврате отрицательного значения
        return(-1);
    }
//--- покажем на какой паре символ/таймфрейм рассчитан индикатор Alligator
    short_name=StringFormat("iAlligator(%s/%s, %d,%d,%d,%d,%d,%d)",name,EnumToString(p
        jaw_period,jaw_shift,teeth_period,teeth_shift,lips_period,
        IndicatorSetString(INDICATOR_SHORTNAME,short_name);
//--- нормальное выполнение инициализации индикатора

```

```

    return(0);
}

//+-----+
//| Custom indicator iteration function |
//+-----+

int OnCalculate(const int rates_total,
               const int prev_calculated,
               const datetime &time[],
               const double &open[],
               const double &high[],
               const double &low[],
               const double &close[],
               const long &tick_volume[],
               const long &volume[],
               const int &spread[])
{
    //--- количество копируемых значений из индикатора iAlligator
    int values_to_copy;
    //--- узнаем количество рассчитанных значений в индикаторе
    int calculated=BarsCalculated(handle);
    if(calculated<=0)
    {
        PrintFormat("BarsCalculated() вернул %d, код ошибки %d",calculated,GetLastError);
        return(0);
    }
    //--- если это первый запуск вычислений нашего индикатора или изменилось количество э
    //--- или если необходимо рассчитать индикатор для двух или более баров (значит что-т
    if(prev_calculated==0 || calculated!=bars_calculated || rates_total>prev_calculate
    {
        //--- если массив iADBuffer больше, чем значений в индикаторе iAD на паре symbo
        //--- в противном случае копировать будем меньше, чем размер индикаторных буфер
        if(calculated>rates_total) values_to_copy=rates_total;
        else
            values_to_copy=calculated;
    }
    else
    {
        //--- значит наш индикатор рассчитывается не в первый раз и с момента последнег
        //--- для расчета добавилось не более одного бара
        values_to_copy=(rates_total-prev_calculated)+1;
    }
    //--- заполняем массивы значениями из индикатора Alligator
    //--- если FillArraysFromBuffer вернула false, значит данные не готовы - завершаем ра
    if(!FillArraysFromBuffers(JawsBuffer,jaw_shift,TeethBuffer,teeth_shift,LipsBuffer,
    //--- сформируем сообщение
    string comm=StringFormat("%s ==> Обновлено значений в индикаторе %s: %d",
                            TimeToString(TimeCurrent(),TIME_DATE|TIME_SECONDS),
                            short_name,
                            values_to_copy);

```

```

//--- выведем на график служебное сообщение
    Comment(comm);
//--- заппомним количество значений в индикаторе Accelerator Oscillator
    bars_calculated=calculated;
//--- вернем значение prev_calculated для следующего вызова
    return(rates_total);
}

//+-----+
//| Заполняем индикаторные буферы из индикатора iAlligator |
//+-----+
bool FillArraysFromBuffers(double &jaws_buffer[], // индикаторный буфер для линии Че
                           int j_shift,          // смещение линии Челюстей
                           double &teeth_buffer[], // индикаторный буфер для линии Зу
                           int t_shift,          // смещение линии Зубов
                           double &lips_buffer[], // индикаторный буфер для линии Гу
                           int l_shift,          // смещение линии Губ
                           int ind_handle,       // хэндл индикатора iAlligator
                           int amount           // количество копируемых значений
                           )
{
//--- сбросим код ошибки
    ResetLastError();
//--- заполняем часть массива JawsBuffer значениями из индикаторного буфера под индексом
    if(CopyBuffer(ind_handle,0,-j_shift,amount,jaws_buffer)<0)
    {
        //--- если копирование не удалось, сообщим код ошибки
        PrintFormat("Не удалось скопировать данные из индикатора iAlligator, код ошибки %d", GetLastError());
        //--- завершим с нулевым результатом - это означает, что индикатор будет считаться неактивным
        return(false);
    }

//--- заполняем часть массива TeethBuffer значениями из индикаторного буфера под индексом
    if(CopyBuffer(ind_handle,1,-t_shift,amount,teeth_buffer)<0)
    {
        //--- если копирование не удалось, сообщим код ошибки
        PrintFormat("Не удалось скопировать данные из индикатора iAlligator, код ошибки %d", GetLastError());
        //--- завершим с нулевым результатом - это означает, что индикатор будет считаться неактивным
        return(false);
    }

//--- заполняем часть массива LipsBuffer значениями из индикаторного буфера под индексом
    if(CopyBuffer(ind_handle,2,-l_shift,amount,lips_buffer)<0)
    {
        //--- если копирование не удалось, сообщим код ошибки
        PrintFormat("Не удалось скопировать данные из индикатора iAlligator, код ошибки %d", GetLastError());
        //--- завершим с нулевым результатом - это означает, что индикатор будет считаться неактивным
        return(false);
    }
}

```

```
    }  
    //--- все получилось  
    return(true);  
}  
//+-----+  
//| Indicator deinitialization function |  
//+-----+  
void OnDeinit(const int reason)  
{  
    //--- почистим график при удалении индикатора  
    Comment("");  
}
```

iAMA

函数返回适当移动平均指标处理器。只有一个缓冲区。

```
int iAMA(
    string          symbol,          // 交易品种名称
    ENUM_TIMEFRAMES period,          // 周期
    int             ama_period,      // AMA平均周期
    int             fast_ma_period,  // 快速 MA 周期
    int             slow_ma_period,  // 慢速 MA 周期
    int             ama_shift,       // 指标平移
    ENUM_APPLIED_PRICE applied_price // 价格或者处理器类型
);
```

参量

symbol

[in] 证券交易品种名称，数据用来计算指标。 [NULL](#) 值代表当前交易品种。

period

[in] 周期值可以是 [ENUM_TIMEFRAMES](#) 值中的一个，0代表当前时间表。

ama_period

[in] 计算周期，计算效率系数。

fast_ma_period

[in] 在快速市场为平滑系数计算快速周期。

slow_ma_period

[in] 在缺乏趋势时为平滑系数计算的慢周期。

ama_shift

[in] 转换关于价格图表的指标。

applied_price

[in] 使用价格。可以是任意 [ENUM_APPLIED_PRICE](#) 价格常量或者另外指标处理器。

返回值

返回特殊技术指标处理器，失败返回 [INVALID_HANDLE](#)。计算机内存从不使用的指标中释放，使用指标处理程序传递到的函数 [IndicatorRelease\(\)](#)。

:

```
//+-----+
//|                                     Demo_iAMA.mq5 |
//|                                     Copyright 2011, MetaQuotes Software Corp. |
//|                                     http://www.mql5.com |
//+-----+
#property copyright "Copyright 2011, MetaQuotes Software Corp."
#property link      "http://www.mql5.com"
#property version   "1.00"
```

```

#property description "Индикатор демонстрирует как нужно получать данные"
#property description "Индикаторных буферов для технического индикатора iAMA."
#property description "Символ и таймфрейм, на котором рассчитывается индикатор,"
#property description "задаются параметрами symbol и period."
#property description "Способ создания хэндла задается параметром 'type' (тип функции)"
#property description "Все остальные параметры как в стандартной AMA."

#property indicator_chart_window
#property indicator_buffers 1
#property indicator_plots 1
//--- plot iAMA
#property indicator_label1 "iAMA"
#property indicator_type1 DRAW_LINE
#property indicator_color1 clrRed
#property indicator_style1 STYLE_SOLID
#property indicator_width1 1
//+-----+
//| перечисление способов создания хэндла |
//+-----+
enum Creation
{
    Call_iAMA,          // использовать iAMA
    Call_IndicatorCreate // использовать IndicatorCreate
};
//--- входные параметры
input Creation      type=Call_iAMA;          // тип функции
input string        symbol=" ";              // символ
input ENUM_TIMEFRAMES period=PERIOD_CURRENT; // таймфрейм
input int            ama_period=15;           // период для расчета
input int            fast_ma_period=2;        // период быстрой скользящей
input int            slow_ma_period=30;       // период медленной скользящей
input int            ama_shift=0;             // смещение по горизонтали
input ENUM_APPLIED_PRICE applied_price;       // тип цены
//--- индикаторный буфер
double iAMABuffer[];
//--- переменная для хранения хэндла индикатора iAMA
int handle;
//--- переменная для хранения
string name=symbol;
//--- имя индикатора на графике
string short_name;
//--- будем хранить количество значений в индикаторе Adaptive Moving Average
int bars_calculated=0;
//+-----+
//| Custom indicator initialization function |
//+-----+
int OnInit()
{

```

```

//--- привязка массива к индикаторному буферу
SetIndexBuffer(0,iAMABuffer,INDICATOR_DATA);
//--- зададим смещение
PlotIndexSetInteger(0,PLOT_SHIFT,ama_shift);
//--- определимся с символом, на котором строится индикатор
name=symbol;
//--- удалим пробелы слева и справа
StringTrimRight(name);
StringTrimLeft(name);
//--- если после этого длина строки name нулевая
if(StringLen(name)==0)
{
    //--- возьмем символ с графика, на котором запущен индикатор
    name=_Symbol;
}
//--- создадим хэндл индикатора
if(type==Call_iAMA)
    handle=iAMA(name,period,ama_period,fast_ma_period,slow_ma_period,ama_shift,applied_price);
else
{
    //--- заполним структуру значениями параметров индикатора
    MqlParam pars[5];
    pars[0].type=TYPE_INT;
    pars[0].integer_value=ama_period;
    pars[1].type=TYPE_INT;
    pars[1].integer_value=fast_ma_period;
    pars[2].type=TYPE_INT;
    pars[2].integer_value=slow_ma_period;
    pars[3].type=TYPE_INT;
    pars[3].integer_value=ama_shift;
    //--- тип цены
    pars[4].type=TYPE_INT;
    pars[4].integer_value=applied_price;
    handle=IndicatorCreate(name,period,IND_AMA,5,pars);
}
//--- если не удалось создать хэндл
if(handle==INVALID_HANDLE)
{
    //--- сообщим о неудаче и выведем номер ошибки
    PrintFormat("Не удалось создать хэндл индикатора iAMA для пары %s/%s, код ошибки: %d",
        name,
        EnumToString(period),
        GetLastError());
    //--- работа индикатора завершается досрочно при возврате отрицательного значения
    return(-1);
}
//--- покажем на какой паре символ/таймфрейм рассчитан индикатор Adaptive Moving Average
short_name=StringFormat("iAMA(%s/%s,%d,%d,%d,%d)",name,EnumToString(period),ama_period,fast_ma_period,slow_ma_period);

```

```

IndicatorSetString(INDICATOR_SHORTNAME,short_name);
//--- нормальное выполнение инициализации индикатора
return(0);
}
//+-----+
//| Custom indicator iteration function |
//+-----+
int OnCalculate(const int rates_total,
                const int prev_calculated,
                const datetime &time[],
                const double &open[],
                const double &high[],
                const double &low[],
                const double &close[],
                const long &tick_volume[],
                const long &volume[],
                const int &spread[])
{
//--- количество копируемых значений из индикатора iAlligator
int values_to_copy;
//--- узнаем количество рассчитанных значений в индикаторе
int calculated=BarsCalculated(handle);
if(calculated<=0)
{
    PrintFormat("BarsCalculated() вернул %d, код ошибки %d",calculated,GetLastError());
    return(0);
}
//--- если это первый запуск вычислений нашего индикатора или изменилось количество з
//--- или если необходимо рассчитать индикатор для двух или более баров (значит что-т
if(prev_calculated==0 || calculated!=bars_calculated || rates_total>prev_calculated)
{
    //--- если массив iADBuffer больше, чем значений в индикаторе iAD на паре symbo
    //--- в противном случае копировать будем меньше, чем размер индикаторных буфер
    if(calculated>rates_total) values_to_copy=rates_total;
    else
        values_to_copy=calculated;
}
else
{
    //--- значит наш индикатор рассчитывается не в первый раз и с момента последнег
    //--- для расчета добавилось не более одного бара
    values_to_copy=(rates_total-prev_calculated)+1;
}
//--- заполняем массивы значениями из индикатора Alligator
//--- если FillArraysFromBuffer вернула false, значит данные не готовы - завершаем ра
if(!FillArrayFromBuffer(iAMABuffer,ama_shift,handle,values_to_copy)) return(0);
//--- сформируем сообщение
string comm=StringFormat("%s ==> Обновлено значений в индикаторе %s: %d",
                          TimeToString(TimeCurrent(),TIME_DATE|TIME_SECONDS),

```



```

        short_name,
        values_to_copy);
//--- выведем на график служебное сообщение
    Comment(comm);
//--- заппомним количество значений в индикаторе Accelerator Oscillator
    bars_calculated=calculated;
//--- вернем значение prev_calculated для следующего вызова
    return(rates_total);
}
//+-----+
//| Заполняем индикаторные буферы из индикатора iAMA |
//+-----+
bool FillArrayFromBuffer(double &ama_buffer[], // индикаторный буфер линии AMA
                        int a_shift,           // смещение линии AMA
                        int ind_handle,        // хэндл индикатора iAMA
                        int amount            // количество копируемых значений
                        )
{
//--- сбросим код ошибки
    ResetLastError();
//--- заполняем часть массива iAMABuffer значениями из индикаторного буфера под индекс
    if(CopyBuffer(ind_handle,0,-a_shift,amount,ama_buffer)<0)
    {
        //--- если копирование не удалось, сообщим код ошибки
        PrintFormat("Не удалось скопировать данные из индикатора iAlligator, код ошибки");
        //--- завершим с нулевым результатом - это означает, что индикатор будет считат
        return(false);
    }
//--- все получилось
    return(true);
}
//+-----+
//| Indicator deinitialization function |
//+-----+
void OnDeinit(const int reason)
{
//--- почистим график при удалении индикатора
    Comment("");
}

```

iAO

函数返回动量震荡指标处理器。只有一个缓冲区。

```
int iAO(
    string      symbol,      // 交易品种名称
    ENUM_TIMEFRAMES period   // 周期
);
```

参量

symbol

[in] 证券交易品种名称，数据用来计算指标。 [NULL](#) 值代表当前交易品种。

period

[in] 周期值可以是 [ENUM_TIMEFRAMES](#) 值中的一个，0代表当前时间表。

返回值

返回特殊技术指标处理器，失败返回 [INVALID_HANDLE](#)。计算机内存从不使用的指标中释放，使用指标处理程序传递到的函数 [IndicatorRelease\(\)](#)。

:

```
//+-----+
//|                                     Demo_iAO.mq5 |
//|                                     Copyright 2011, MetaQuotes Software Corp. |
//|                                     http://www.mql5.com |
//+-----+
#property copyright "Copyright 2011, MetaQuotes Software Corp."
#property link      "http://www.mql5.com"
#property version   "1.00"
#property description "Индикатор демонстрирует как нужно получать данные"
#property description "индикаторных буферов для технического индикатора iAO."
#property description "Символ и таймфрейм, на котором рассчитывается индикатор,"
#property description "задаются параметрами symbol и period."
#property description "Способ создания хэндла задается параметром 'type' (тип функции"

#property indicator_separate_window
#property indicator_buffers 2
#property indicator_plots 1
//--- построение iAO
#property indicator_label1 "iAO"
#property indicator_type1  DRAW_COLOR_HISTOGRAM
#property indicator_color1 clrGreen,clrRed
#property indicator_style1 STYLE_SOLID
#property indicator_width1 1
//+-----+
//| перечисление способов создания хэндла |
//+-----+
enum Creation
```

```

{
    Call_iAO,                // использовать iAO
    Call_IndicatorCreate     // использовать IndicatorCreate
};

//--- входные параметры
input Creation              type=Call_iAO;           // тип функции
input string                symbol=" ";              // символ
input ENUM_TIMEFRAMES      period=PERIOD_CURRENT;   // таймфрейм
//--- индикаторные буферы
double iAOBuffer[];
double iAOCOLORS[];
//--- переменная для хранения хэндла индикатора iAO
int handle;
//--- переменная для хранения
string name=symbol;
//--- имя индикатора на графике
string short_name;
//--- будем хранить количество значений в индикаторе Awesome Oscillator
int bars_calculated=0;
//+-----+
//| Custom indicator initialization function |
//+-----+
int OnInit()
{
    //--- привязка массивов к индикаторным буферам
    SetIndexBuffer(0,iAOBuffer,INDICATOR_DATA);
    SetIndexBuffer(1,iAOCOLORS,INDICATOR_COLOR_INDEX);
    //--- определимся с символом, на котором строится индикатор
    name=symbol;
    //--- удалим пробелы слева и справа
    StringTrimRight(name);
    StringTrimLeft(name);
    //--- если после этого длина строки name нулевая
    if(StringLen(name)==0)
    {
        //--- возьмем символ с графика, на котором запущен индикатор
        name=_Symbol;
    }
    //--- создадим хэндл индикатора
    if(type==Call_iAO)
        handle=iAO(name,period);
    else
        handle=IndicatorCreate(name,period,IND_AO);
    //--- если не удалось создать хэндл
    if(handle==INVALID_HANDLE)
    {
        //--- сообщим о неудаче и выведем номер ошибки
        PrintFormat("Не удалось создать хэндл индикатора iAO для пары %s/%s, код ошибки

```

```

        name,
        EnumToString(period),
        GetLastError());

    //--- работа индикатора завершается досрочно при возврате отрицательного значения
    return(-1);
}

//--- покажем на какой паре символ/таймфрейм рассчитан индикатор Awesome Oscillator
short_name=StringFormat("iAO(%s/%s)",name,EnumToString(period));
IndicatorSetString(INDICATOR_SHORTNAME,short_name);
//--- нормальное выполнение инициализации индикатора
return(0);
}

//+-----+
//| Custom indicator iteration function |
//+-----+

int OnCalculate(const int rates_total,
               const int prev_calculated,
               const datetime &time[],
               const double &open[],
               const double &high[],
               const double &low[],
               const double &close[],
               const long &tick_volume[],
               const long &volume[],
               const int &spread[])
{
    //--- количество копируемых значений из индикатора iAO
    int values_to_copy;
    //--- узнаем количество рассчитанных значений в индикаторе
    int calculated=BarsCalculated(handle);
    if(calculated<=0)
    {
        PrintFormat("BarsCalculated() вернул %d, код ошибки %d",calculated,GetLastError());
        return(0);
    }
    //--- если это первый запуск вычислений нашего индикатора или изменилось количество з
    //--- или если необходимо рассчитать индикатор для двух или более баров (значит что-т
    if(prev_calculated==0 || calculated!=bars_calculated || rates_total>prev_calculated)
    {
        //--- если массив iAOWBuffer больше, чем значений в индикаторе iAO на паре symbo
        //--- в противном случае копировать будем меньше, чем размер индикаторных буфер
        if(calculated>rates_total) values_to_copy=rates_total;
        else
            values_to_copy=calculated;
    }
    else
    {
        //--- значит наш индикатор рассчитывается не в первый раз и с момента последнег
        //--- для расчета добавилось не более одного бара

```

```

        values_to_copy=(rates_total-prev_calculated)+1;
    }
//--- заполняем массивы iAObuffer и iAOColors значениями из индикатора Awesome Oscill
//--- если FillArraysFromBuffer вернула false, значит данные не готовы - завершаем ра
    if(!FillArraysFromBuffer(iAObuffer,iAOColors,handle,values_to_copy)) return(0);
//--- сформируем сообщение
    string comm=StringFormat("%s ==> Обновлено значений в индикаторе %s: %d",
                              TimeToString(TimeCurrent(),TIME_DATE|TIME_SECONDS),
                              short_name,
                              values_to_copy);
//--- выведем на график служебное сообщение
    Comment(comm);
//--- заппомним количество значений в индикаторе Accelerator Oscillator
    bars_calculated=calculated;
//--- вернем значение prev_calculated для следующего вызова
    return(rates_total);
}
//+-----+
//| Заполняем индикаторные буферы из индикатора iAO |
//+-----+
bool FillArraysFromBuffer(double &values[], // индикаторный буфер значений Awe
                          double &color_indexes[], // цветовой буфер (для хранения ин
                          int ind_handle, // хэндл индикатора iAO
                          int amount // количество копируемых значений
                          )
{
//--- сбросим код ошибки
    ResetLastError();
//--- заполняем часть массива iAObuffer значениями из индикаторного буфера под индекс
    if(CopyBuffer(ind_handle,0,0,amount,values)<0)
    {
        //--- если копирование не удалось, сообщим код ошибки
        PrintFormat("Не удалось скопировать данные из индикатора iAO, код ошибки %d",Ge
        //--- завершим с нулевым результатом - это означает, что индикатор будет считат
        return(false);
    }
//--- теперь копируем индексы цветов
    if(CopyBuffer(ind_handle,1,0,amount,color_indexes)<0)
    {
        //--- если копирование не удалось, сообщим код ошибки
        PrintFormat("Не удалось скопировать значения цветов из индикатора iAO, код ошиб
        //--- завершим с нулевым результатом - это означает, что индикатор будет считат
        return(false);
    }
//--- все получилось
    return(true);
}
//+-----+

```

```
///| Indicator deinitialization function |  
//+-----+  
void OnDeinit(const int reason)  
{  
//--- почистим график при удалении индикатора  
    Comment("");  
}
```

iATR

函数返回平均真实区域指标处理器。只有一个缓冲区。

```
int iATR(
    string      symbol,      // 交易品种名称
    ENUM_TIMEFRAMES period,  // 周期
    int         ma_period    // 平均周期
);
```

参量

symbol

[in] 证券交易品种名称，数据用来计算指标。 [NULL](#) 值代表当前交易品种。

period

[in] 周期值可以是 [ENUM_TIMEFRAMES](#) 值中的一个，0代表当前时间表。

ma_period

[in] 指标计算平均周期值。

返回值

返回特殊技术指标处理器，失败返回 [INVALID_HANDLE](#)。计算机内存从不使用的指标中释放，使用指标处理程序传递到的函数 [IndicatorRelease\(\)](#)。

:

```
//+-----+
//|                                     Demo_iATR.mq5 |
//|                                     Copyright 2011, MetaQuotes Software Corp. |
//|                                     http://www.mql5.com |
//+-----+

#property copyright "Copyright 2011, MetaQuotes Software Corp."
#property link      "http://www.mql5.com"
#property version   "1.00"
#property description "Индикатор демонстрирует как нужно получать данные"
#property description "индикаторных буферов для технического индикатора iATR."
#property description "Символ и таймфрейм, на котором рассчитывается индикатор,"
#property description "задаются параметрами symbol и period."
#property description "Способ создания хэндла задается параметром 'type' (тип функции"

#property indicator_separate_window
#property indicator_buffers 1
#property indicator_plots   1
//--- plot iATR
#property indicator_label1  "iATR"
#property indicator_type1   DRAW_LINE
#property indicator_color1  clrLightSeaGreen
#property indicator_style1  STYLE_SOLID
#property indicator_width1  1
```

```

//+-----+
//|  перечисление способов создания хэндла |
//+-----+
enum Creation
{
    Call_iATR, // использовать iATR
    Call_IndicatorCreate // использовать IndicatorCreate
};
//--- входные параметры
input int          atr_period=14;           // период для вычисления
input Creation     type=Call_iATR;          // тип функции
input string       symbol=" ";              // символ
input ENUM_TIMEFRAMES period=PERIOD_CURRENT; // таймфрейм
//--- индикаторный буфер
double            iATRBufter[];
//--- переменная для хранения хэндла индикатора iAC
int               handle;
//--- переменная для хранения
string name=symbol;
//--- имя индикатора на графике
string short_name;
//--- будем хранить количество значений в индикаторе Average True Range
int               bars_calculated=0;
//+-----+
//| Custom indicator initialization function |
//+-----+
int OnInit()
{
    //--- привязка массива к индикаторному буферу
    SetIndexBuffer(0,iATRBufter,INDICATOR_DATA);
    //--- определимся с символом, на котором строится индикатор
    name=symbol;
    //--- удалим пробелы слева и справа
    StringTrimRight(name);
    StringTrimLeft(name);
    //--- если после этого длина строки name нулевая
    if(StringLen(name)==0)
    {
        //--- возьмем символ с графика, на котором запущен индикатор
        name=_Symbol;
    }
    //--- создадим хэндл индикатора
    if(type==Call_iATR)
        handle=iATR(name,period,atr_period);
    else
    {
        //--- заполним структуру значениями параметров индикатора
        MqlParam pars[1];
    }
}

```



```

    pars[0].type=TYPE_INT;
    pars[0].integer_value=atr_period;
    handle=IndicatorCreate(name,period,IND_ATR,1,pars);
}

//--- если не удалось создать хэндл
if(handle==INVALID_HANDLE)
{
    //--- сообщим о неудаче и выведем номер ошибки
    PrintFormat("Не удалось создать хэндл индикатора iATR для пары %s/%s, код ошибки",
        name,
        EnumToString(period),
        GetLastError());

    //--- работа индикатора завершается досрочно при возврате отрицательного значения
    return(-1);
}

//--- покажем на какой паре символ/таймфрейм рассчитан индикатор Average True Range
short_name=StringFormat("iATR(%s/%s, period=%d)",name,EnumToString(period),atr_period);
IndicatorSetString(INDICATOR_SHORTNAME,short_name);
//--- нормальное выполнение инициализации индикатора
return(0);
}

//+-----+
//| Custom indicator iteration function |
//+-----+

int OnCalculate(const int rates_total,
                const int prev_calculated,
                const datetime &time[],
                const double &open[],
                const double &high[],
                const double &low[],
                const double &close[],
                const long &tick_volume[],
                const long &volume[],
                const int &spread[])
{
    //--- количество копируемых значений из индикатора iATR
    int values_to_copy;
    //--- узнаем количество рассчитанных значений в индикаторе
    int calculated=BarsCalculated(handle);
    if(calculated<=0)
    {
        PrintFormat("BarsCalculated() вернул %d, код ошибки %d",calculated,GetLastError());
        return(0);
    }

    //--- если это первый запуск вычислений нашего индикатора или изменилось количество э
    //--- или если необходимо рассчитать индикатор для двух или более баров (значит что-т
    if(prev_calculated==0 || calculated!=bars_calculated || rates_total>prev_calculated)
    {

```

```

    //--- если массив iATRBuffer больше, чем значений в индикаторе iATR на паре sym
    //--- в противном случае копировать будем меньше, чем размер индикаторных буфер
    if(calculated>rates_total) values_to_copy=rates_total;
    else
        values_to_copy=calculated;
    }
else
{
    //--- значит наш индикатор рассчитывается не в первый раз и с момента последнего
    //--- для расчета добавилось не более одного бара
    values_to_copy=(rates_total-prev_calculated)+1;
}

//--- заполняем массив iATRBuffer значениями из индикатора Average True Range
//--- если FillArrayFromBuffer вернула false, значит данные не готовы - завершаем работу
if(!FillArrayFromBuffer(iATRBuffer,handle,values_to_copy)) return(0);
//--- сформируем сообщение
string comm=StringFormat("%s ==> Обновлено значений в индикаторе %s: %d",
                          TimeToString(TimeCurrent(),TIME_DATE|TIME_SECONDS),
                          short_name,
                          values_to_copy);

//--- выведем на график служебное сообщение
Comment(comm);

//--- запоним количество значений в индикаторе Accelerator Oscillator
bars_calculated=calculated;

//--- вернем значение prev_calculated для следующего вызова
return(rates_total);
}

//+-----+
//| Заполняем индикаторный буфер из индикатора iATR |
//+-----+
bool FillArrayFromBuffer(double &values[], // индикаторный буфер значений ATR
                        int ind_handle,    // хэндл индикатора iATR
                        int amount         // количество копируемых значений
                        )
{
    //--- сбросим код ошибки
    ResetLastError();
    //--- заполняем часть массива iATRBuffer значениями из индикаторного буфера под индексом
    if(CopyBuffer(ind_handle,0,0,amount,values)<0)
    {
        //--- если копирование не удалось, сообщим код ошибки
        PrintFormat("Не удалось скопировать данные из индикатора iATR, код ошибки %d",GetLastError());
        //--- завершим с нулевым результатом - это означает, что индикатор будет считаться неактивным
        return(false);
    }
    //--- все получилось
    return(true);
}

//+-----+

```

```
///| Indicator deinitialization function |
//+-----+
void OnDeinit(const int reason)
{
//--- почистим график при удалении индикатора
    Comment("");
}
```

iBearsPower

函数返回熊市指标处理器。只有一个缓冲区。

```
int iBearsPower(
    string          symbol,          // 交易品种名称
    ENUM_TIMEFRAMES period,          // 周期
    int             ma_period,        // 平均周期
);
```

参量

symbol

[in] 证券交易品种名称，数据用来计算指标。 [NULL](#) 值代表当前交易品种。

period

[in] 周期值可以是 [ENUM_TIMEFRAMES](#) 值中的一个，0代表当前时间表。

ma_period

[in] 指标计算平均周期值。

返回值

返回特殊技术指标处理器，失败返回 [INVALID_HANDLE](#)。计算机内存从不使用的指标中释放，使用指标处理程序传递到的函数 [IndicatorRelease\(\)](#)。

:

```
//+-----+
//|                                     Demo_iBearsPower.mq5 |
//|                                     Copyright 2011, MetaQuotes Software Corp. |
//|                                     http://www.mql5.com |
//+-----+

#property copyright "Copyright 2011, MetaQuotes Software Corp."
#property link      "http://www.mql5.com"
#property version   "1.00"
#property description "Индикатор демонстрирует как нужно получать данные"
#property description "индикаторных буферов для технического индикатора iBearsPower."
#property description "Символ и таймфрейм, на котором рассчитывается индикатор,"
#property description "задаются параметрами symbol и period."
#property description "Способ создания хэндла задается параметром 'type' (тип функции"

#property indicator_separate_window
#property indicator_buffers 1
#property indicator_plots 1
//--- построение iBearsPower
#property indicator_label1 "iBearsPower"
#property indicator_type1 DRAW_HISTOGRAM
#property indicator_color1 clrSilver
#property indicator_style1 STYLE_SOLID
#property indicator_width1 1
```

```

//+-----+
//|  перечисление способов создания хэндла |
//+-----+
enum Creation
{
    Call_iBearsPower,      // использовать iBearsPower
    Call_IndicatorCreate    // использовать IndicatorCreate
};
//--- входные параметры
input Creation            type=Call_iBearsPower;  // тип функции
input int                 ma_period=13;          // период скользящей
input string              symbol=" ";            // символ
input ENUM_TIMEFRAMES     period=PERIOD_CURRENT; // таймфрейм
//--- индикаторный буфер
double                    iBearsPowerBuffer[];
//--- переменная для хранения хэндла индикатора iBearsPower
int                        handle;
//--- переменная для хранения
string name=symbol;
//--- имя индикатора на графике
string short_name;
//--- будем хранить количество значений в индикаторе Bears Power
int                        bars_calculated=0;
//+-----+
//| Custom indicator initialization function |
//+-----+
int OnInit()
{
    //--- привязка массива к индикаторному буферу
    SetIndexBuffer(0,iBearsPowerBuffer,INDICATOR_DATA);
    //--- определимся с символом, на котором строится индикатор
    name=symbol;
    //--- удалим пробелы слева и справа
    StringTrimRight(name);
    StringTrimLeft(name);
    //--- если после этого длина строки name нулевая
    if(StringLen(name)==0)
    {
        //--- возьмем символ с графика, на котором запущен индикатор
        name=_Symbol;
    }
    //--- создадим хэндл индикатора
    if(type==Call_iBearsPower)
        handle=iBearsPower(name,period,ma_period);
    else
    {
        //--- заполним структуру значениями параметров индикатора
        MqlParam pars[1];
    }
}

```

```

    //--- период скользящей
    pars[0].type=TYPE_INT;
    pars[0].integer_value=ma_period;
    handle=IndicatorCreate(name,period,IND_BEARS,1,pars);
}
//--- если не удалось создать хэндл
if(handle==INVALID_HANDLE)
{
    //--- сообщим о неудаче и выведем номер ошибки
    PrintFormat("Не удалось создать хэндл индикатора iBearsPower для пары %s/%s, ко
                name,
                EnumToString(period),
                GetLastError());
    //--- работа индикатора завершается досрочно при возврате отрицательного значен
    return(-1);
}
//--- покажем на какой паре символ/таймфрейм рассчитан индикатор Bears Power
short_name=StringFormat("iBearsPower(%s/%s, period=%d)",name,EnumToString(period),
IndicatorSetString(INDICATOR_SHORTNAME,short_name);
//--- нормальное выполнение инициализации индикатора
return(0);
}
//+-----+
//| Custom indicator iteration function |
//+-----+
int OnCalculate(const int rates_total,
                const int prev_calculated,
                const datetime &time[],
                const double &open[],
                const double &high[],
                const double &low[],
                const double &close[],
                const long &tick_volume[],
                const long &volume[],
                const int &spread[])
{
    //--- количество копируемых значений из индикатора iBearsPower
    int values_to_copy;
    //--- узнаем количество рассчитанных значений в индикаторе
    int calculated=BarsCalculated(handle);
    if(calculated<=0)
    {
        PrintFormat("BarsCalculated() вернул %d, код ошибки %d",calculated,GetLastError
        return(0);
    }
    //--- если это первый запуск вычислений нашего индикатора или изменилось количество э
    //--- или если необходимо рассчитать индикатор для двух или более баров (значит что-т
    if(prev_calculated==0 || calculated!=bars_calculated || rates_total>prev_calculate

```

```

    {
        //--- если массив iATRBuffer больше, чем значений в индикаторе iBearsPower на п
        //--- в противном случае копировать будем меньше, чем размер индикаторных буфер
        if(calculated>rates_total) values_to_copy=rates_total;
        else                        values_to_copy=calculated;
    }
else
{
    //--- значит наш индикатор рассчитывается не в первый раз и с момента последнег
    //--- для расчета добавилось не более одного бара
    values_to_copy=(rates_total-prev_calculated)+1;
}

//--- заполняем массив iBearsPowerBuffer значениями из индикатора Bears Power
//--- если FillArrayFromBuffer вернула false, значит данные не готовы - завершаем раб
if(!FillArrayFromBuffer(iBearsPowerBuffer,handle,values_to_copy)) return(0);
//--- сформируем сообщение
string comm=StringFormat("%s ==> Обновлено значений в индикаторе %s: %d",
                          TimeToString(TimeCurrent(),TIME_DATE|TIME_SECONDS),
                          short_name,
                          values_to_copy);

//--- выведем на график служебное сообщение
Comment(comm);

//--- запомним количество значений в индикаторе Bears Power
bars_calculated=calculated;
//--- вернем значение prev_calculated для следующего вызова
return(rates_total);
}

//+-----+
//| Заполняем индикаторный буфер из индикатора iBearsPower |
//+-----+
bool FillArrayFromBuffer(double &values[], // индикаторный буфер значений Bears Powe
                        int ind_handle,    // хэндл индикатора iBearsPower
                        int amount         // количество копируемых значений
                        )
{
    //--- сбросим код ошибки
    ResetLastError();
    //--- заполняем часть массива iBearsPowerBuffer значениями из индикаторного буфера по
    if(CopyBuffer(ind_handle,0,0,amount,values)<0)
    {
        //--- если копирование не удалось, сообщим код ошибки
        PrintFormat("Не удалось скопировать данные из индикатора iBearsPower, код ошибк
        //--- завершим с нулевым результатом - это означает, что индикатор будет считат
        return(false);
    }
    //--- все получилось
    return(true);
}

```

```
//+-----+  
//| Indicator deinitialization function |  
//+-----+  
void OnDeinit(const int reason)  
{  
//--- почистим график при удалении индикатора  
    Comment("");  
}
```


iBands

函数返回布林带指标处理器。

```
int iBands(
    string          symbol,           // 交易品种名称
    ENUM_TIMEFRAMES period,          // 周期
    int             bands_period,     // 均线计算周期
    int             bands_shift,      // 指标平移
    double          deviation,        // 标准差数
    ENUM_APPLIED_PRICE applied_price // 价格或处理器类型
);
```

参量

symbol

[in] 证券交易品种名称，数据用来计算指标。 [NULL](#) 值代表当前交易品种。

period

[in] 周期值可以是 [ENUM_TIMEFRAMES](#) 值中的一个，0代表当前时间表。

bands_period

[in] 指标主线的平均周期。

bands_shift

[in] 相关价格图表的指标转换。

deviation

[in] 偏离主线。

applied_price

[in] 使用价格。可以是任意 [ENUM_APPLIED_PRICE](#) 价格常量或者另外指标处理器。

返回值

返回特殊技术指标处理器，失败返回 [INVALID_HANDLE](#)。计算机内存从不使用的指标中释放，使用指标处理程序传递到的函数 [IndicatorRelease\(\)](#)。

注释

缓冲区代码如下：0 - BASE_ LINE，1 - UPPER_ BAND，2 - LOWER_ BAND。

:

```
//+-----+
//|                                     Demo_iBands.mq5 |
//|                                     Copyright 2011, MetaQuotes Software Corp. |
//|                                     http://www.mql5.com |
//+-----+
#property copyright "Copyright 2011, MetaQuotes Software Corp."
#property link      "http://www.mql5.com"
#property version   "1.00"
#property description "Индикатор демонстрирует как нужно получать данные"
```

```

#property description "индикаторных буферов для технического индикатора iBands."
#property description "Символ и таймфрейм, на котором рассчитывается индикатор,"
#property description "задаются параметрами symbol и period."
#property description "Способ создания хэндла задается параметром 'type' (тип функции)

#property indicator_chart_window
#property indicator_buffers 3
#property indicator_plots 3
//--- построение Upper
#property indicator_label1 "Upper"
#property indicator_type1 DRAW_LINE
#property indicator_color1 clrMediumSeaGreen
#property indicator_style1 STYLE_SOLID
#property indicator_width1 1
//--- построение Lower
#property indicator_label2 "Lower"
#property indicator_type2 DRAW_LINE
#property indicator_color2 clrMediumSeaGreen
#property indicator_style2 STYLE_SOLID
#property indicator_width2 1
//--- построение Middle
#property indicator_label3 "Middle"
#property indicator_type3 DRAW_LINE
#property indicator_color3 clrMediumSeaGreen
#property indicator_style3 STYLE_SOLID
#property indicator_width3 1
//+-----+
//| перечисление способов создания хэндла |
//+-----+
enum Creation
{
    Call_iBands,          // использовать iBands
    Call_IndicatorCreate  // использовать IndicatorCreate
};
//--- входные параметры
input Creation          type=Call_iBands;          // тип функции
input int               bands_period=20;           // период скользящей средней
input int               bands_shift=0;             // сдвиг
input double            deviation=2.0;             // кол-во стандартных отклонений
input ENUM_APPLIED_PRICE applied_price=PRICE_CLOSE; // тип цены
input string            symbol=" ";               // символ
input ENUM_TIMEFRAMES   period=PERIOD_CURRENT;    // таймфрейм
//--- индикаторные буферы
double      UpperBuffer[];
double      LowerBuffer[];
double      MiddleBuffer[];
//--- переменная для хранения хэндла индикатора iBands
int         handle;

```

```

//--- переменная для хранения
string name=symbol;
//--- имя индикатора на графике
string short_name;
//--- будем хранить количество значений в индикаторе Bollinger Bands
int bars_calculated=0;
//+-----+
//| Custom indicator initialization function |
//+-----+
int OnInit()
{
//--- привязка массивов к индикаторным буферам
SetIndexBuffer(0,UpperBuffer,INDICATOR_DATA);
SetIndexBuffer(1,LowerBuffer,INDICATOR_DATA);
SetIndexBuffer(2,MiddleBuffer,INDICATOR_DATA);
//--- зададим смещение для каждой линии
PlotIndexSetInteger(0,PLOT_SHIFT,bands_shift);
PlotIndexSetInteger(1,PLOT_SHIFT,bands_shift);
PlotIndexSetInteger(2,PLOT_SHIFT,bands_shift);
//--- определимся с символом, на котором строится индикатор
name=symbol;
//--- удалим пробелы слева и справа
StringTrimRight(name);
StringTrimLeft(name);
//--- если после этого длина строки name нулевая
if(StringLen(name)==0)
{
//--- возьмем символ с графика, на котором запущен индикатор
name=_Symbol;
}
//--- создадим хэндл индикатора
if(type==Call_iBands)
handle=iBands(name,period,bands_period,bands_shift,deviation,applied_price);
else
{
//--- заполним структуру значениями параметров индикатора
MqlParam pars[4];
//--- период скользящей
pars[0].type=TYPE_INT;
pars[0].integer_value=bands_period;
//--- смещение
pars[1].type=TYPE_INT;
pars[1].integer_value=bands_shift;
//--- количество стандартных отклонений
pars[2].type=TYPE_DOUBLE;
pars[2].double_value=deviation;
//--- тип цены
pars[3].type=TYPE_INT;

```

```

        pars[3].integer_value=applied_price;
        handle=IndicatorCreate(name,period,IND_BANDS,4,pars);
    }
//--- если не удалось создать хэндл
    if(handle==INVALID_HANDLE)
    {
        //--- сообщим о неудаче и выведем номер ошибки
        PrintFormat("Не удалось создать хэндл индикатора iBands для пары %s/%s, код оши
            name,
            EnumToString(period),
            GetLastError());
        //--- работа индикатора завершается досрочно при возврате отрицательного значен
        return(-1);
    }
//--- покажем на какой паре символ/таймфрейм рассчитан индикатор Bollinger Bands
    short_name=StringFormat("iBands(%s/%s, %d,%d,%G,%s)",name,EnumToString(period),
        bands_period,bands_shift,deviation,EnumToString(applied_pr
    IndicatorSetString(INDICATOR_SHORTNAME,short_name);
//--- нормальное выполнение инициализации индикатора
    return(0);
}
//+-----+
//| Custom indicator iteration function |
//+-----+
int OnCalculate(const int rates_total,
                const int prev_calculated,
                const datetime &time[],
                const double &open[],
                const double &high[],
                const double &low[],
                const double &close[],
                const long &tick_volume[],
                const long &volume[],
                const int &spread[])
{
//--- количество копируемых значений из индикатора iBands
    int values_to_copy;
//--- узнаем количество рассчитанных значений в индикаторе
    int calculated=BarsCalculated(handle);
    if(calculated<=0)
    {
        PrintFormat("BarsCalculated() вернул %d, код ошибки %d",calculated,GetLastError
        return(0);
    }
//--- если это первый запуск вычислений нашего индикатора или изменилось количество з
//--- или если необходимо рассчитать индикатор для двух или более баров (значит что-т
    if(prev_calculated==0 || calculated!=bars_calculated || rates_total>prev_calculate
    {

```

```

    //--- если размер индикаторных массивов больше, чем значений в индикаторе iBands
    //--- в противном случае копировать будем меньше, чем размер индикаторных буфер
    if(calculated>rates_total) values_to_copy=rates_total;
    else
        values_to_copy=calculated;
    }
else
{
    //--- значит наш индикатор рассчитывается не в первый раз и с момента последнего
    //--- для расчета добавилось не более одного бара
    values_to_copy=(rates_total-prev_calculated)+1;
}

//--- заполняем массив значениями из индикатора Bollinger Bands
//--- если FillArraysFromBuffer вернула false, значит данные не готовы - завершаем ра
    if(!FillArraysFromBuffers(MiddleBuffer,UpperBuffer,LowerBuffer,bands_shift,handle,
//--- сформируем сообщение
    string comm=StringFormat("%s ==> Обновлено значений в индикаторе %s: %d",
                                TimeToString(TimeCurrent(),TIME_DATE|TIME_SECONDS),
                                short_name,
                                values_to_copy);

//--- выведем на график служебное сообщение
    Comment(comm);

//--- запомним количество значений в индикаторе Bollinger Bands
    bars_calculated=calculated;

//--- вернем значение prev_calculated для следующего вызова
    return(rates_total);
}

//+-----+
//| Заполняем индикаторные буферы из индикатора iBands |
//+-----+

bool FillArraysFromBuffers(double &base_values[], // индикаторный буфер средней л
                        double &upper_values[], // индикаторный буфер верхней г
                        double &lower_values[], // индикаторный буфер верхней
                        int shift, // смещение
                        int ind_handle, // хэндл индикатора iBands
                        int amount // количество копируемых значен

    {
//--- сбросим код ошибки
    ResetLastError();

//--- заполняем часть массива MiddleBuffer значениями из индикаторного буфера под инд
    if(CopyBuffer(ind_handle,0,-shift,amount,base_values)<0)
    {
        //--- если копирование не удалось, сообщим код ошибки
        PrintFormat("Не удалось скопировать данные из индикатора iBands, код ошибки %d"
        //--- завершим с нулевым результатом - это означает, что индикатор будет считат
        return(false);
    }
}

```

```

//--- заполняем часть массива UpperBuffer значениями из индикаторного буфера под инде
    if(CopyBuffer(ind_handle,1,-shift,amount,upper_values)<0)
    {
        //--- если копирование не удалось, сообщим код ошибки
        PrintFormat("Не удалось скопировать данные из индикатора iBands, код ошибки %d"
        //--- завершим с нулевым результатом - это означает, что индикатор будет считат
        return(false);
    }

//--- заполняем часть массива LowerBuffer значениями из индикаторного буфера под инде
    if(CopyBuffer(ind_handle,2,-shift,amount,lower_values)<0)
    {
        //--- если копирование не удалось, сообщим код ошибки
        PrintFormat("Не удалось скопировать данные из индикатора iBands, код ошибки %d"
        //--- завершим с нулевым результатом - это означает, что индикатор будет считат
        return(false);
    }
//--- все получилось
    return(true);
}

//+-----+
//| Indicator deinitialization function |
//+-----+
void OnDeinit(const int reason)
{
    //--- почистим график при удалении индикатора
    Comment("");
}

```

iBullsPower

函数返回牛市指标处理器。只有一个缓冲区。

```
int iBullsPower(
    string          symbol,          // 交易品种名称
    ENUM_TIMEFRAMES period,          // 周期
    int             ma_period,        // 平均周期
);
```

参量

symbol

[in] 证券交易品种名称，数据用来计算指标。 [NULL](#) 值代表当前交易品种。

period

[in] 周期值可以是 [ENUM_TIMEFRAMES](#) 值中的一个，0代表当前时间表。

ma_period

[in] 指标计算平均周期值。

返回值

返回特殊技术指标处理器，失败返回 [INVALID_HANDLE](#)。计算机内存从不使用的指标中释放，使用指标处理程序传递到的函数 [IndicatorRelease\(\)](#)。

:

```
//+-----+
//|                                     Demo_iBullsPower.mq5 |
//|                                     Copyright 2011, MetaQuotes Software Corp. |
//|                                     http://www.mql5.com |
//+-----+

#property copyright "Copyright 2011, MetaQuotes Software Corp."
#property link      "http://www.mql5.com"
#property version   "1.00"
#property description "Индикатор демонстрирует как нужно получать данные"
#property description "индикаторных буферов для технического индикатора iBullsPower."
#property description "Символ и таймфрейм, на котором рассчитывается индикатор,"
#property description "задаются параметрами symbol и period."
#property description "Способ создания хэндла задается параметром 'type' (тип функции"

#property indicator_separate_window
#property indicator_buffers 1
#property indicator_plots 1
//--- построение iBullsPower
#property indicator_label1 "iBullsPower"
#property indicator_type1  DRAW_HISTOGRAM
#property indicator_color1 clrSilver
#property indicator_style1 STYLE_SOLID
#property indicator_width1 1
```

```

//+-----+
//|  перечисление способов создания хэндла |
//+-----+
enum Creation
{
    Call_iBullsPower,      // использовать iBullsPower
    Call_IndicatorCreate    // использовать IndicatorCreate
};
//--- входные параметры
input Creation            type=Call_iBullsPower;  // тип функции
input int                 ma_period=13;           // период скользящей
input string              symbol=" ";             // символ
input ENUM_TIMEFRAMES     period=PERIOD_CURRENT;  // таймфрейм
//--- индикаторный буфер
double                    iBullsPowerPowerBuffer[];
//--- переменная для хранения хэндла индикатора iBullsPower
int                        handle;
//--- переменная для хранения
string name=symbol;
//--- имя индикатора на графике
string short_name;
//--- будем хранить количество значений в индикаторе Bulls Power
int                        bars_calculated=0;
//+-----+
//| Custom indicator initialization function |
//+-----+
int OnInit()
{
    //--- привязка массива к индикаторному буферу
    SetIndexBuffer(0,iBullsPowerPowerBuffer,INDICATOR_DATA);
    //--- определимся с символом, на котором строится индикатор
    name=symbol;
    //--- удалим пробелы слева и справа
    StringTrimRight(name);
    StringTrimLeft(name);
    //--- если после этого длина строки name нулевая
    if(StringLen(name)==0)
    {
        //--- возьмем символ с графика, на котором запущен индикатор
        name=_Symbol;
    }
    //--- создадим хэндл индикатора
    if(type==Call_iBullsPower)
        handle=iBullsPower(name,period,ma_period);
    else
    {
        //--- заполним структуру значениями параметров индикатора
        MqlParam pars[1];

```



```

    //--- период скользящей
    pars[0].type=TYPE_INT;
    pars[0].integer_value=ma_period;
    handle=IndicatorCreate(name,period,IND_BULLS,1,pars);
}
//--- если не удалось создать хэндл
if(handle==INVALID_HANDLE)
{
    //--- сообщим о неудаче и выведем номер ошибки
    PrintFormat("Не удалось создать хэндл индикатора iBullsPower для пары %s/%s, ко
                name,
                EnumToString(period),
                GetLastError());
    //--- работа индикатора завершается досрочно при возврате отрицательного значен
    return(-1);
}
//--- покажем на какой паре символ/таймфрейм рассчитан индикатор Bulls Power
short_name=StringFormat("iBullsPower(%s/%s, period=%d)",name,EnumToString(period),
IndicatorSetString(INDICATOR_SHORTNAME,short_name);
//--- нормальное выполнение инициализации индикатора
return(0);
}
//+-----+
//| Custom indicator iteration function |
//+-----+
int OnCalculate(const int rates_total,
                const int prev_calculated,
                const datetime &time[],
                const double &open[],
                const double &high[],
                const double &low[],
                const double &close[],
                const long &tick_volume[],
                const long &volume[],
                const int &spread[])
{
    //--- количество копируемых значений из индикатора iBullsPower
    int values_to_copy;
    //--- узнаем количество рассчитанных значений в индикаторе
    int calculated=BarsCalculated(handle);
    if(calculated<=0)
    {
        PrintFormat("BarsCalculated() вернул %d, код ошибки %d",calculated,GetLastError
        return(0);
    }
    //--- если это первый запуск вычислений нашего индикатора или изменилось количество э
    //--- или если необходимо рассчитать индикатор для двух или более баров (значит что-т
    if(prev_calculated==0 || calculated!=bars_calculated || rates_total>prev_calculate

```

```

{
    //--- если массив iBullsPowerPowerBuffer больше, чем значений в индикаторе iBul
    //--- в противном случае копировать будем меньше, чем размер индикаторных буфер
    if(calculated>rates_total) values_to_copy=rates_total;
    else                        values_to_copy=calculated;
}
else
{
    //--- значит наш индикатор рассчитывается не в первый раз и с момента последнег
    //--- для расчета добавилось не более одного бара
    values_to_copy=(rates_total-prev_calculated)+1;
}

//--- заполняем массив iBullsPowerBuffer значениями из индикатора Bulls Power
//--- если FillArrayFromBuffer вернула false, значит данные не готовы - завершаем раб
    if(!FillArrayFromBuffer(iBullsPowerPowerBuffer,handle,values_to_copy)) return(0);
//--- сформируем сообщение
    string comm=StringFormat("%s ==> Обновлено значений в индикаторе %s: %d",
                             TimeToString(TimeCurrent(),TIME_DATE|TIME_SECONDS),
                             short_name,
                             values_to_copy);

//--- выведем на график служебное сообщение
    Comment(comm);
//--- запомним количество значений в индикаторе Bulls Power
    bars_calculated=calculated;
//--- вернем значение prev_calculated для следующего вызова
    return(rates_total);
}

//+-----+
//| Заполняем индикаторный буфер из индикатора iBullsPower |
//+-----+

bool FillArrayFromBuffer(double &values[], // индикаторный буфер значений Bulls Powe
                        int ind_handle,    // хэндл индикатора iBullsPower
                        int amount         // количество копируемых значений
                        )
{
    //--- сбросим код ошибки
    ResetLastError();
//--- заполняем часть массива iBullsPowerBuffer значениями из индикаторного буфера по
    if(CopyBuffer(ind_handle,0,0,amount,values)<0)
    {
        //--- если копирование не удалось, сообщим код ошибки
        PrintFormat("Не удалось скопировать данные из индикатора iBullsPower, код ошибк
        //--- завершим с нулевым результатом - это означает, что индикатор будет считат
        return(false);
    }
//--- все получилось
    return(true);
}

```

```
//+-----+
//| Indicator deinitialization function |
//+-----+
void OnDeinit(const int reason)
{
    //--- почистим график при удалении индикатора
    Comment("");
}
//+-----+
```

iCCI

函数返回顺势指标处理器。只有一个缓冲区。

```
int iCCI(
    string          symbol,          // 交易品种名称
    ENUM_TIMEFRAMES period,          // 周期
    int             ma_period,        // 平均周期
    ENUM_APPLIED_PRICE applied_price // 价格或处理器类型
);
```

参量

symbol

[in] 证券交易品种名称，数据用来计算指标。 [NULL](#) 值代表当前交易品种。

period

[in] 周期值可以是 [ENUM_TIMEFRAMES](#) 值中的一个，0代表当前时间表。

ma_period

[in] 指标计算平均周期值。

applied_price

[in] 使用价格。可以是任意 [ENUM_APPLIED_PRICE](#) 价格常量或者另外指标处理器。

返回值

返回特殊技术指标处理器，失败返回 [INVALID_HANDLE](#)。计算机内存从不使用的指标中释放，使用指标处理程序传递到的函数 [IndicatorRelease\(\)](#)。

:

```
//+-----+
//|                                     Demo_iCCI.mq5 |
//|                                     Copyright 2011, MetaQuotes Software Corp. |
//|                                     http://www.mql5.com |
//+-----+

#property copyright "Copyright 2011, MetaQuotes Software Corp."
#property link      "http://www.mql5.com"
#property version   "1.00"
#property description "Индикатор демонстрирует как нужно получать данные"
#property description "индикаторных буферов для технического индикатора iCCI."
#property description "Символ и таймфрейм, на котором рассчитывается индикатор,"
#property description "задаются параметрами symbol и period."
#property description "Способ создания хэндла задается параметром 'type' (тип функции"

#property indicator_separate_window
#property indicator_buffers 1
#property indicator_plots 1
//--- построение iCCI
#property indicator_label1 "iCCI"
#property indicator_type1  DRAW_LINE
```

```

#property indicator_color1 clrLightSeaGreen
#property indicator_style1 STYLE_SOLID
#property indicator_width1 1
//--- горизонтальные уровни в окне индикатора
#property indicator_level1 -100.0
#property indicator_level2 100.0
//+-----+
//| перечисление способов создания хэндла |
//+-----+
enum Creation
{
    Call_iCCI,          // использовать iCCI
    Call_IndicatorCreate // использовать IndicatorCreate
};
//--- входные параметры
input Creation      type=Call_iCCI;          // тип функции
input int           ma_period=14;            // период скользящей средней
input ENUM_APPLIED_PRICE applied_price=PRICE_TYPICAL; // тип цены
input string        symbol=" ";              // символ
input ENUM_TIMEFRAMES period=PERIOD_CURRENT; // таймфрейм
//--- индикаторный буфер
double iCCIBuffer[];
//--- переменная для хранения хэндла индикатора iCCI
int handle;
//--- переменная для хранения
string name=symbol;
//--- имя индикатора на графике
string short_name;
//--- будем хранить количество значений в индикаторе Commodity Channel Index
int bars_calculated=0;
//+-----+
//| Custom indicator initialization function |
//+-----+
int OnInit()
{
    //--- привязка массива к индикаторному буферу
    SetIndexBuffer(0,iCCIBuffer,INDICATOR_DATA);
    //--- определимся с символом, на котором строится индикатор
    name=symbol;
    //--- удалим пробелы слева и справа
    StringTrimRight(name);
    StringTrimLeft(name);
    //--- если после этого длина строки name нулевая
    if(StringLen(name)==0)
    {
        //--- возьмем символ с графика, на котором запущен индикатор
        name=_Symbol;
    }
}

```

```

//--- создадим хэндл индикатора
if(type==Call_iCCI)
    handle=iCCI(name,period,ma_period,applied_price);
else
{
    //--- заполним структуру значениями параметров индикатора
    MqlParam pars[2];
    //--- период средней
    pars[0].type=TYPE_INT;
    pars[0].integer_value=ma_period;
    //--- тип цены
    pars[1].type=TYPE_INT;
    pars[1].integer_value=applied_price;
    handle=IndicatorCreate(name,period,IND_CCI,2,pars);
}

//--- если не удалось создать хэндл
if(handle==INVALID_HANDLE)
{
    //--- сообщим о неудаче и выведем номер ошибки
    PrintFormat("Не удалось создать хэндл индикатора iCCI для пары %s/%s, код ошибки",
        name,
        EnumToString(period),
        GetLastError());

    //--- работа индикатора завершается досрочно при возврате отрицательного значения
    return(-1);
}

//--- покажем на какой паре символ/таймфрейм рассчитан индикатор Bollinger Bands
short_name=StringFormat("iCCI(%s/%s, %d, %s)",name,EnumToString(period),
    ma_period,EnumToString(applied_price));
IndicatorSetString(INDICATOR_SHORTNAME,short_name);
//--- нормальное выполнение инициализации индикатора
return(0);
}

//+-----+
//| Custom indicator iteration function |
//+-----+

int OnCalculate(const int rates_total,
    const int prev_calculated,
    const datetime &time[],
    const double &open[],
    const double &high[],
    const double &low[],
    const double &close[],
    const long &tick_volume[],
    const long &volume[],
    const int &spread[])
{
    //--- количество копируемых значений из индикатора iCCI

```

```

    int values_to_copy;
    //--- узнаем количество рассчитанных значений в индикаторе
    int calculated=BarsCalculated(handle);
    if(calculated<=0)
    {
        PrintFormat("BarsCalculated() вернул %d, код ошибки %d",calculated,GetLastError());
        return(0);
    }
    //--- если это первый запуск вычислений нашего индикатора или изменилось количество з
    //--- или если необходимо рассчитать индикатор для двух или более баров (значит что-т
    if(prev_calculated==0 || calculated!=bars_calculated || rates_total>prev_calculate
    {
        //--- если массив iATRBuffer больше, чем значений в индикаторе iCCI на паре sym
        //--- в противном случае копировать будем меньше, чем размер индикаторных буфер
        if(calculated>rates_total) values_to_copy=rates_total;
        else
            values_to_copy=calculated;
    }
    else
    {
        //--- значит наш индикатор рассчитывается не в первый раз и с момента последнет
        //--- для расчета добавилось не более одного бара
        values_to_copy=(rates_total-prev_calculated)+1;
    }
    //--- заполняем массив iCCIBuffer значениями из индикатора Commodity Channel Index
    //--- если FillArrayFromBuffer вернула false, значит данные не готовы - завершаем раб
    if(!FillArrayFromBuffer(iCCIBuffer,handle,values_to_copy)) return(0);
    //--- сформируем сообщение
    string comm=StringFormat("%s ==> Обновлено значений в индикаторе %s: %d",
                               TimeToString(TimeCurrent(),TIME_DATE|TIME_SECONDS),
                               short_name,
                               values_to_copy);
    //--- выведем на график служебное сообщение
    Comment(comm);
    //--- запомним количество значений в индикаторе Commodity Channel Index
    bars_calculated=calculated;
    //--- вернем значение prev_calculated для следующего вызова
    return(rates_total);
}
//+-----+
//| Заполняем индикаторный буфер из индикатора iCCI |
//+-----+
bool FillArrayFromBuffer(double &values[], // индикаторный буфер значений Commodity
                        int ind_handle,    // хэндл индикатора iCCI
                        int amount        // количество копируемых значений
                        )
{
    //--- сбросим код ошибки
    ResetLastError();

```

```

//--- заполняем часть массива iCCIBuffer значениями из индикаторного буфера под индекс
if(CopyBuffer(ind_handle,0,0,amount,values)<0)
{
    //--- если копирование не удалось, сообщим код ошибки
    PrintFormat("Не удалось скопировать данные из индикатора iCCI, код ошибки %d",G
    //--- завершим с нулевым результатом - это означает, что индикатор будет считат
    return(false);
}
//--- все получилось
return(true);
}
//+-----+
//| Indicator deinitialization function |
//+-----+
void OnDeinit(const int reason)
{
    //--- почистим график при удалении индикатора
    Comment("");
}

```


iChaikin

The function returns the handle of the Chaikin Oscillator indicator. It has only one buffer.

```
int iChaikin(
    string          symbol,          // 交易品种名称
    ENUM_TIMEFRAMES period,          // 周期
    int             fast_ma_period,   // 快速周期
    int             slow_ma_period,   // 慢速周期
    ENUM_MA_METHOD  ma_method,        // 平滑类型
    ENUM_APPLIED_VOLUME applied_volume // 交易量类型
);
```

参量

symbol

[in] 证券交易品种名称，数据用来计算指标。 [NULL](#) 值代表当前交易品种。

period

[in] 周期值可以是 [ENUM_TIMEFRAMES](#) 值中的一个，0代表当前时间表。

fast_ma_period

[in] 计算快速平均周期。

slow_ma_period

[in] 计算慢速平均周期。

ma_method

[in] 平滑类型。可以是 [ENUM_MA_METHOD](#) 平均常量中的一个。

applied_volume

[in] 使用的交易量。可以是 [ENUM_APPLIED_VOLUME](#) 常量中的一个。

返回值

返回特殊技术指标处理器，失败返回 [INVALID_HANDLE](#)。计算机内存从不使用的指标中释放，使用指标处理程序传递到的函数 [IndicatorRelease\(\)](#)。

```
//+-----+
//|                                     Demo_iChaikin.mq5 |
//|                                     Copyright 2011, MetaQuotes Software Corp. |
//|                                     http://www.mql5.com |
//+-----+
#property copyright "Copyright 2011, MetaQuotes Software Corp."
#property link      "http://www.mql5.com"
#property version   "1.00"
#property description "Индикатор демонстрирует как нужно получать данные"
#property description "индикаторных буферов для технического индикатора iChaikin."
#property description "Символ и таймфрейм, на котором рассчитывается индикатор,"
#property description "задаются параметрами symbol и period."
```

```

#property description "Способ создания хэндла задается параметром 'type' (тип функции

#property indicator_separate_window
#property indicator_buffers 1
#property indicator_plots 1
//--- построение iChaikin
#property indicator_label1 "iChaikin"
#property indicator_type1 DRAW_LINE
#property indicator_color1 clrLightSeaGreen
#property indicator_style1 STYLE_SOLID
#property indicator_width1 1
//+-----+
//| перечисление способов создания хэндла |
//+-----+
enum Creation
{
    Call_iChaikin,          // использовать iChaikin
    Call_IndicatorCreate    // использовать IndicatorCreate
};
//--- входные параметры
input Creation             type=Call_iChaikin;          // тип функции
input int                 fast_ma_period=3;             // быстрый период
input int                 slow_ma_period=10;            // медленный период
input ENUM_MA_METHOD      ma_method=MODE_EMA;          // тип сглаживания
input ENUM_APPLIED_VOLUME applied_volume=VOLUME_TICK;  // тип объема
input string              symbol=" ";                  // символ
input ENUM_TIMEFRAMES     period=PERIOD_CURRENT;       // таймфрейм
//--- индикаторный буфер
double iChaikinBuffer[];
//--- переменная для хранения хэндла индикатора iChaikin
int handle;
//--- переменная для хранения
string name=symbol;
//--- имя индикатора на графике
string short_name;
//--- будем хранить количество значений в индикаторе Chaikin Oscillator
int bars_calculated=0;
//+-----+
//| Custom indicator initialization function |
//+-----+
int OnInit()
{
    //--- привязка массива к индикаторному буферу
    SetIndexBuffer(0,iChaikinBuffer,INDICATOR_DATA);
    //--- определимся с символом, на котором строится индикатор
    name=symbol;
    //--- удалим пробелы слева и справа
    StringTrimRight(name);

```

```

StringTrimLeft(name);
//--- если после этого длина строки name нулевая
if(StringLen(name)==0)
{
    //--- возьмем символ с графика, на котором запущен индикатор
    name=_Symbol;
}
//--- создадим хэндл индикатора
if(type==Call_iChaikin)
    handle=iChaikin(name,period,fast_ma_period,slow_ma_period,ma_method,applied_vol
else
{
    //--- заполним структуру значениями параметров индикатора
    MqlParam pars[4];
    //--- быстрый период
    pars[0].type=TYPE_INT;
    pars[0].integer_value=fast_ma_period;
    //--- медленный период
    pars[1].type=TYPE_INT;
    pars[1].integer_value=slow_ma_period;
    //--- тип сглаживания
    pars[2].type=TYPE_INT;
    pars[2].integer_value=ma_method;
    //--- тип объема
    pars[3].type=TYPE_INT;
    pars[3].integer_value=applied_volume;
    handle=IndicatorCreate(name,period,IND_CHAIKIN,4,pars);
}
//--- если не удалось создать хэндл
if(handle==INVALID_HANDLE)
{
    //--- сообщим о неудаче и выведем номер ошибки
    PrintFormat("Не удалось создать хэндл индикатора iChaikin для пары %s/%s, код о
        name,
        EnumToString(period),
        GetLastError());
    //--- работа индикатора завершается досрочно при возврате отрицательного значен
    return(-1);
}
//--- покажем на какой паре символ/таймфрейм рассчитан индикатор Chaikin Oscillator
short_name=StringFormat("iChaikin(%s/%s, %d, %d, %s, %s)",name,EnumToString(period
        fast_ma_period,slow_ma_period,
        EnumToString(ma_method),EnumToString(applied_volume));
IndicatorSetString(INDICATOR_SHORTNAME,short_name);
//--- нормальное выполнение инициализации индикатора
return(0);
}
//+-----+

```

```

//| Custom indicator iteration function |
//+-----+
int OnCalculate(const int rates_total,
               const int prev_calculated,
               const datetime &time[],
               const double &open[],
               const double &high[],
               const double &low[],
               const double &close[],
               const long &tick_volume[],
               const long &volume[],
               const int &spread[])
{
    //--- количество копируемых значений из индикатора iChaikin
    int values_to_copy;
    //--- узнаем количество рассчитанных значений в индикаторе
    int calculated=BarsCalculated(handle);
    if(calculated<=0)
    {
        PrintFormat("BarsCalculated() вернул %d, код ошибки %d",calculated,GetLastError());
        return(0);
    }
    //--- если это первый запуск вычислений нашего индикатора или изменилось количество з
    //--- или если необходимо рассчитать индикатор для двух или более баров (значит что-т
    if(prev_calculated==0 || calculated!=bars_calculated || rates_total>prev_calculated)
    {
        //--- если массив iChaikinBuffer больше, чем значений в индикаторе iChaikin на
        //--- в противном случае копировать будем меньше, чем размер индикаторных буфер
        if(calculated>rates_total) values_to_copy=rates_total;
        else
            values_to_copy=calculated;
    }
    else
    {
        //--- значит наш индикатор рассчитывается не в первый раз и с момента последнег
        //--- для расчета добавилось не более одного бара
        values_to_copy=(rates_total-prev_calculated)+1;
    }
    //--- заполняем массив iChaikinBuffer значениями из индикатора Chaikin Oscillator
    //--- если FillArrayFromBuffer вернула false, значит данные не готовы - завершаем раб
    if(!FillArrayFromBuffer(iChaikinBuffer,handle,values_to_copy)) return(0);
    //--- сформируем сообщение
    string comm=StringFormat("%s ==> Обновлено значений в индикаторе %s: %d",
                             TimeToString(TimeCurrent(),TIME_DATE|TIME_SECONDS),
                             short_name,
                             values_to_copy);
    //--- выведем на график служебное сообщение
    Comment(comm);
    //--- запомним количество значений в индикаторе Chaikin Oscillator

```

```

bars_calculated=calculated;
//--- вернем значение prev_calculated для следующего вызова
return(rates_total);
}
//+-----+
//| Заполняем индикаторный буфер из индикатора iChaikin |
//+-----+
bool FillArrayFromBuffer(double &values[], // индикаторный буфер значений Chaikin Os
                        int ind_handle,    // хэндл индикатора iChaikin
                        int amount        // количество копируемых значений
                        )
{
//--- сбросим код ошибки
ResetLastError();
//--- заполняем часть массива iChaikinBuffer значениями из индикаторного буфера под и
if(CopyBuffer(ind_handle,0,0,amount,values)<0)
{
//--- если копирование не удалось, сообщим код ошибки
PrintFormat("Не удалось скопировать данные из индикатора iChaikin, код ошибки %
//--- завершим с нулевым результатом - это означает, что индикатор будет считат
return(false);
}
//--- все получилось
return(true);
}
//+-----+
//| Indicator deinitialization function |
//+-----+
void OnDeinit(const int reason)
{
//--- почистим график при удалении индикатора
Comment("");
}

```

iCustom

函数返回指定自定义指标的处理器。

```
int iCustom(
    string      symbol,      // 交易品种名称
    ENUM_TIMEFRAMES period,  // 周期
    string      name        // 文件夹/自定义指标_名称
    ...         // 指标输入参量列表
);
```

参量

symbol

[in] 证券交易品种名称，数据用来计算指标。 [NULL](#) 值代表当前交易品种。

period

[in] 周期值可以是 [ENUM_TIMEFRAMES](#) 值中的一个，0代表当前时间表。

name

[in] 自定义指标名称，指标的相关根基目录路径（ MQL5/Indicators/）。如果指标在子目录中保持，例如，在 MQL5/Indicators/Examples 中，名称一定是制定的，像 "Examples\indicator_ name"（分隔符有必要使用双精度斜线代替单精度斜线）。

...

[in] 自定义指标的 [输入-参量](#)，逗号分开，类型和参量命令必须匹配，如果没有指定参量，使用 [默认值](#)。

返回值

返回特殊技术指标处理器，失败返回 [INVALID_HANDLE](#)。计算机内存从不使用的指标中释放，使用指标处理程序传递到的函数 [IndicatorRelease\(\)](#)。

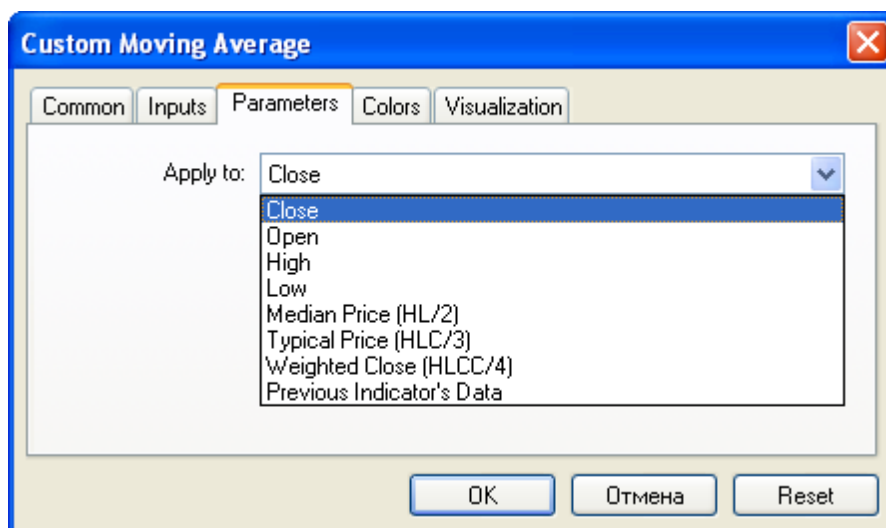
注释

必须编辑自定义指标（扩展EX5）并且在可获端或子目录中保存在MQL5/Indicators 目录中。

需要测试的指标是指自动从iCustom() 函数调用，如果相应的参数是通过一个通用的[常量字符串](#)集合。对于其它情况下（使用[IndicatorCreate\(\)](#) 函数或一个非常量字符串设定指标名称）该属性[#property tester_indicator](#)是必需的：

```
#property tester_indicator "indicator_name.ex5"
```

如果在指标中使用 [第一调用方式](#)，然后在自定义指标中可以在"Parameters"标签中自动形成为指示数据的计算，如果"Apply to"参量不是明确指定的，默认计算将以"Close"价格值为主。



当从MQL5程序中调用自定义指标时，Applied_ Price参量或者另外的指标处理器应该最后传递，在所有自定义指标的输入变量之后。

另见

[程序属性](#), [时间序列和指标访问](#), [IndicatorCreate\(\)](#), [IndicatorRelease\(\)](#)

示例：

```
#property indicator_separate_window
#property indicator_buffers 1
#property indicator_plots 1
//---- 标签图1
#property indicator_label1 "Label1"
#property indicator_type1 DRAW_LINE
#property indicator_color1 clrRed
#property indicator_style1 STYLE_SOLID
#property indicator_width1 1
//--- 输入参量
input int MA_Period=21;
input int MA_Shift=0;
input ENUM_MA_METHOD MA_Method=MODE_SMA;
//--- 指标缓冲区
double Label1Buffer[];
//--- 自定义移动平均数处理程序。mq5自定义指标
int MA_handle;
//+-----+
//| 自定义指标初始化函数 |
//+-----+
int OnInit()
{
//--- 指标缓冲区绘图
SetIndexBuffer(0,Label1Buffer,INDICATOR_DATA);
ResetLastError();
MA_handle=iCustom(NULL,0,"Custom Moving Average",
```

```

        MA_Period,
        MA_Shift,
        MA_Method,
        PRICE_CLOSE // 使用收盘价
    );

    Print("MA_handle = ",MA_handle," error = ",GetLastError());
//---
    return(0);
}
//+-----+
//| 自定义指标重复函数                                     |
//+-----+
int OnCalculate(const int rates_total,
               const int prev_calculated,
               const datetime &time[],
               const double &open[],
               const double &high[],
               const double &low[],
               const double &close[],
               const long &tick_volume[],
               const long &volume[],
               const int &spread[])
{
//--- 复制指标自定义移动平均值到指标缓冲区
    int copy=CopyBuffer(MA_handle,0,0,rates_total,Label1Buffer);
    Print("copy = ",copy," rates_total = ",rates_total);
//--- 如果尝试失败-这里报道
    if(copy<=0)
        Print("An attempt to get the values if Custom Moving Average has failed");
//--- 为下次调用返回prev_calculated值
    return(rates_total);
}
//+-----+

```


iDEMA

函数返回双指数移动平均线指标处理器。只有一个缓冲区。

```
int iDEMA(
    string          symbol,          // 交易品种名称
    ENUM_TIMEFRAMES period,          // 周期
    int             ma_period,        // 平均周期
    int             ma_shift,         // 平移
    ENUM_APPLIED_PRICE applied_price // 价格或者处理器类型
);
```

参量

symbol

[in] 证券交易品种名称，数据用来计算指标。 [NULL](#) 值代表当前交易品种。

period

[in] 周期值可以是 [ENUM_TIMEFRAMES](#) 值中的一个，0代表当前时间表。

ma_period

[in] 计算平均周期（计算柱）。

ma_shift

[in] 关于价格图表转变的指标。

applied_price

[in] 使用价格。可以是任意 [ENUM_APPLIED_PRICE](#) 价格常量或者另外指标处理器。

返回值

返回特殊技术指标处理器，失败返回 [INVALID_HANDLE](#)。计算机内存从不使用的指标中释放，使用指标处理程序传递到的函数 [IndicatorRelease\(\)](#)。

:

```
//+-----+
//|                                     Demo_iDEMA.mq5 |
//|                                     Copyright 2011, MetaQuotes Software Corp. |
//|                                     http://www.mql5.com |
//+-----+
#property copyright "Copyright 2011, MetaQuotes Software Corp."
#property link      "http://www.mql5.com"
#property version   "1.00"
#property description "Индикатор демонстрирует как нужно получать данные"
#property description "индикаторных буферов для технического индикатора iDEMA."
#property description "Символ и таймфрейм, на котором рассчитывается индикатор,"
#property description "задаются параметрами symbol и period."
#property description "Способ создания хэндла задается параметром 'type' (тип функции"

#property indicator_chart_window
#property indicator_buffers 1
```

```

#property indicator_plots 1
//--- построение iDEMA
#property indicator_label1 "iDEMA"
#property indicator_type1 DRAW_LINE
#property indicator_color1 clrRed
#property indicator_style1 STYLE_SOLID
#property indicator_width1 1
//+-----+
//| перечисление способов создания хэнгла |
//+-----+
enum Creation
{
    Call_iDEMA,          // использовать iDEMA
    Call_IndicatorCreate // использовать IndicatorCreate
};
//--- входные параметры
input Creation      type=Call_iDEMA;          // тип функции
input int           ma_period=14;              // период скользящей средней
input int           ma_shift=0;                // смещение
input ENUM_APPLIED_PRICE applied_price=PRICE_CLOSE; // тип цены
input string        symbol=" ";               // символ
input ENUM_TIMEFRAMES period=PERIOD_CURRENT;  // таймфрейм
//--- индикаторный буфер
double iDEMABuffer[];
//--- переменная для хранения хэнгла индикатора iDEMA
int handle;
//--- переменная для хранения
string name=symbol;
//--- имя индикатора на графике
string short_name;
//--- будем хранить количество значений в индикаторе Double Exponential Moving Average
int bars_calculated=0;
//+-----+
//| Custom indicator initialization function |
//+-----+
int OnInit()
{
    //--- привязка массива к индикаторному буферу
    SetIndexBuffer(0,iDEMABuffer,INDICATOR_DATA);
    //--- зададим смещение
    PlotIndexSetInteger(0,PLOT_SHIFT,ma_shift);
    //--- определимся с символом, на котором строится индикатор
    name=symbol;
    //--- удалим пробелы слева и справа
    StringTrimRight(name);
    StringTrimLeft(name);
    //--- если после этого длина строки name нулевая
    if(StringLen(name)==0)

```

```

    {
        //--- возьмем символ с графика, на котором запущен индикатор
        name=_Symbol;
    }
//--- создадим хэндл индикатора
if(type==Call_iDEMA)
    handle=iDEMA(name,period,ma_period,ma_shift,applied_price);
else
    {
        //--- заполним структуру значениями параметров индикатора
        MqlParam pars[3];
        //--- период средней
        pars[0].type=TYPE_INT;
        pars[0].integer_value=ma_period;
        //--- смещение
        pars[1].type=TYPE_INT;
        pars[1].integer_value=ma_shift;
        //--- тип цены
        pars[2].type=TYPE_INT;
        pars[2].integer_value=applied_price;
        handle=IndicatorCreate(name,period,IND_DEMA,3,pars);
    }
//--- если не удалось создать хэндл
if(handle==INVALID_HANDLE)
    {
        //--- сообщим о неудаче и выведем номер ошибки
        PrintFormat("Не удалось создать хэндл индикатора iDEMA для пары %s/%s, код ошибки",
            name,
            EnumToString(period),
            GetLastError());
        //--- работа индикатора завершается досрочно при возврате отрицательного значения
        return(-1);
    }
//--- покажем на какой паре символ/таймфрейм рассчитан индикатор Double Exponential Moving Average
short_name=StringFormat("iDEMA(%s/%s, %d, %d, %s)",name,EnumToString(period),
    ma_period,ma_shift,EnumToString(applied_price));
IndicatorSetString(INDICATOR_SHORTNAME,short_name);
//--- нормальное выполнение инициализации индикатора
return(0);
}
//+-----+
//| Custom indicator iteration function |
//+-----+
int OnCalculate(const int rates_total,
    const int prev_calculated,
    const datetime &time[],
    const double &open[],
    const double &high[],

```

```

        const double &low[],
        const double &close[],
        const long &tick_volume[],
        const long &volume[],
        const int &spread[])

{
//--- количество копируемых значений из индикатора iDEMA
    int values_to_copy;
//--- узнаем количество рассчитанных значений в индикаторе
    int calculated=BarsCalculated(handle);
    if(calculated<=0)
    {
        PrintFormat("BarsCalculated() вернул %d, код ошибки %d",calculated,GetLastError);
        return(0);
    }
//--- если это первый запуск вычислений нашего индикатора или изменилось количество з
//--- или если необходимо рассчитать индикатор для двух или более баров (значит что-т
    if(prev_calculated==0 || calculated!=bars_calculated || rates_total>prev_calculate
    {
        //--- если массив iDEMABuffer больше, чем значений в индикаторе iDEMA на паре s
        //--- в противном случае копировать будем меньше, чем размер индикаторных буфер
        if(calculated>rates_total) values_to_copy=rates_total;
        else
            values_to_copy=calculated;
    }
    else
    {
        //--- значит наш индикатор рассчитывается не в первый раз и с момента последнег
        //--- для расчета добавилось не более одного бара
        values_to_copy=(rates_total-prev_calculated)+1;
    }
//--- заполняем массив iDEMABuffer значениями из индикатора Double Exponential Movin
//--- если FillArrayFromBuffer вернула false, значит данные не готовы - завершаем раб
    if(!FillArrayFromBuffer(iDEMABuffer,ma_shift,handle,values_to_copy)) return(0);
//--- сформируем сообщение
    string comm=StringFormat("%s ==> Обновлено значений в индикаторе %s: %d",
                               TimeToString(TimeCurrent(),TIME_DATE|TIME_SECONDS),
                               short_name,
                               values_to_copy);
//--- выведем на график служебное сообщение
    Comment(comm);
//--- запомним количество значений в индикаторе Double Exponential Moving Average
    bars_calculated=calculated;
//--- вернем значение prev_calculated для следующего вызова
    return(rates_total);
}

//+-----+
//| Заполняем индикаторный буфер из индикатора iDEMA |
//+-----+

```

```

bool FillArrayFromBuffer(double &values[], // индикаторный буфер значений Double Exp
                        int shift,         // смещение
                        int ind_handle,     // хэндл индикатора iDEMA
                        int amount         // количество копируемых значений
                        )
{
    //--- сбросим код ошибки
    ResetLastError();
    //--- заполняем часть массива iDEMABuffer значениями из индикаторного буфера под инде
    if(CopyBuffer(ind_handle,0,-shift,amount,values)<0)
    {
        //--- если копирование не удалось, сообщим код ошибки
        PrintFormat("Не удалось скопировать данные из индикатора iDEMA, код ошибки %d",
        //--- завершим с нулевым результатом - это означает, что индикатор будет считат
        return(false);
    }
    //--- все получилось
    return(true);
}

//+-----+
//| Indicator deinitialization function |
//+-----+
void OnDeinit(const int reason)
{
    //--- почистим график при удалении индикатора
    Comment("");
}

```

iDeMarker

函数返回DeMarker指标处理器。只有一个缓冲区。

```
int iDeMarker(
    string      symbol,      // 交易品种名称
    ENUM_TIMEFRAMES period,  // 周期
    int         ma_period    // 平均周期
);
```

参量

symbol

[in] 证券交易品种名称，数据用来计算指标。 [NULL](#) 值代表当前交易品种。

period

[in] 周期值可以是 [ENUM_TIMEFRAMES](#) 值中的一个，0代表当前时间表。

ma_period

[in] 计算平均周期（计算柱）。

返回值

返回特殊技术指标处理器，失败返回 [INVALID_HANDLE](#)。计算机内存从不使用的指标中释放，使用指标处理程序传递到的函数 [IndicatorRelease\(\)](#)。

:

```
//+-----+
//|                                     Demo_iDeMarker.mq5 |
//|                                     Copyright 2011, MetaQuotes Software Corp. |
//|                                     http://www.mql5.com |
//+-----+

#property copyright "Copyright 2011, MetaQuotes Software Corp."
#property link      "http://www.mql5.com"
#property version   "1.00"
#property description "Индикатор демонстрирует как нужно получать данные"
#property description "индикаторных буферов для технического индикатора iDeMarker."
#property description "Символ и таймфрейм, на котором рассчитывается индикатор,"
#property description "задаются параметрами symbol и period."
#property description "Способ создания хэндла задается параметром 'type' (тип функции"

#property indicator_separate_window
#property indicator_buffers 1
#property indicator_plots 1
//--- построение iDeMarker
#property indicator_label1 "iDeMarker"
#property indicator_type1  DRAW_LINE
#property indicator_color1 clrLightSeaGreen
#property indicator_style1 STYLE_SOLID
#property indicator_width1 1
```

```

//--- горизонтальные уровни в окне индикатора
#property indicator_level1 0.3
#property indicator_level2 0.7
//+-----+
//| перечисление способов создания хэндла |
//+-----+
enum Creation
{
    Call_iDeMarker,      // использовать iDeMarker
    Call_IndicatorCreate // использовать IndicatorCreate
};
//--- входные параметры
input Creation      type=Call_iDeMarker; // тип функции
input int           ma_period=14;        // период скользящей
input string        symbol=" ";          // символ
input ENUM_TIMEFRAMES period=PERIOD_CURRENT; // таймфрейм
//--- индикаторный буфер
double iDeMarkerBuffer[];
//--- переменная для хранения хэндла индикатора iDeMarker
int handle;
//--- переменная для хранения
string name=symbol;
//--- имя индикатора на графике
string short_name;
//--- будем хранить количество значений в индикаторе DeMarker
int bars_calculated=0;
//+-----+
//| Custom indicator initialization function |
//+-----+
int OnInit()
{
    //--- привязка массива к индикаторному буферу
    SetIndexBuffer(0,iDeMarkerBuffer,INDICATOR_DATA);
    //--- определимся с символом, на котором строится индикатор
    name=symbol;
    //--- удалим пробелы слева и справа
    StringTrimRight(name);
    StringTrimLeft(name);
    //--- если после этого длина строки name нулевая
    if(StringLen(name)==0)
    {
        //--- возьмем символ с графика, на котором запущен индикатор
        name=_Symbol;
    }
    //--- создадим хэндл индикатора
    if(type==Call_iDeMarker)
        handle=iDeMarker(name,period,ma_period);
    else

```

```

    {
        //--- заполним структуру значениями параметров индикатора
        MqlParam pars[1];
        //--- период средней
        pars[0].type=TYPE_INT;
        pars[0].integer_value=ma_period;
        handle=IndicatorCreate(name,period,IND_DEMARKER,1,pars);
    }
//--- если не удалось создать хэндл
if(handle==INVALID_HANDLE)
{
    //--- сообщим о неудаче и выведем номер ошибки
    PrintFormat("Не удалось создать хэндл индикатора iDeMarker для пары %s/%s, код
                name,
                EnumToString(period),
                GetLastError());
    //--- работа индикатора завершается досрочно при возврате отрицательного значения
    return(-1);
}
//--- покажем на какой паре символ/таймфрейм рассчитан индикатор DeMarker
short_name=StringFormat("iDeMarker(%s/%s, period=%d)",name,EnumToString(period),ma
IndicatorSetString(INDICATOR_SHORTNAME,short_name);
//--- нормальное выполнение инициализации индикатора
return(0);
}
//+-----+
//| Custom indicator iteration function |
//+-----+
int OnCalculate(const int rates_total,
                const int prev_calculated,
                const datetime &time[],
                const double &open[],
                const double &high[],
                const double &low[],
                const double &close[],
                const long &tick_volume[],
                const long &volume[],
                const int &spread[])
{
    //--- количество копируемых значений из индикатора iDeMarker
    int values_to_copy;
    //--- узнаем количество рассчитанных значений в индикаторе
    int calculated=BarsCalculated(handle);
    if(calculated<=0)
    {
        PrintFormat("BarsCalculated() вернул %d, код ошибки %d",calculated,GetLastError
        return(0);
    }
}

```



```

//--- если это первый запуск вычислений нашего индикатора или изменилось количество з
//--- или если необходимо рассчитать индикатор для двух или более баров (значит что-т
    if(prev_calculated==0 || calculated!=bars_calculated || rates_total>prev_calculate
    {
        //--- если массив iDEMABuffer больше, чем значений в индикаторе iDeMarker на па
        //--- в противном случае копировать будем меньше, чем размер индикаторных буфер
        if(calculated>rates_total) values_to_copy=rates_total;
        else
            values_to_copy=calculated;
    }
    else
    {
        //--- значит наш индикатор рассчитывается не в первый раз и с момента последнет
        //--- для расчета добавилось не более одного бара
        values_to_copy=(rates_total-prev_calculated)+1;
    }

//--- заполняем массив iDeMarkerBuffer значениями из индикатора DeMarker
//--- если FillArrayFromBuffer вернула false, значит данные не готовы - завершаем раб
    if(!FillArrayFromBuffer(iDeMarkerBuffer,handle,values_to_copy)) return(0);
//--- сформируем сообщение
    string comm=StringFormat("%s ==> Обновлено значений в индикаторе %s: %d",
                            TimeToString(TimeCurrent(),TIME_DATE|TIME_SECONDS),
                            short_name,
                            values_to_copy);

//--- выведем на график служебное сообщение
    Comment(comm);

//--- заппомним количество значений в индикаторе DeMarker
    bars_calculated=calculated;

//--- вернем значение prev_calculated для следующего вызова
    return(rates_total);
}

//+-----+
//| Заполняем индикаторный буфер из индикатора iDeMarker |
//+-----+
bool FillArrayFromBuffer(double &values[], // индикаторный буфер значений DeMarker
                        int ind_handle, // хэндл индикатора iDeMarker
                        int amount // количество копируемых значений
                        )
{
    //--- сбросим код ошибки
    ResetLastError();

//--- заполняем часть массива iDeMarkerBuffer значениями из индикаторного буфера под
    if(CopyBuffer(ind_handle,0,0,amount,values)<0)
    {
        //--- если копирование не удалось, сообщим код ошибки
        PrintFormat("Не удалось скопировать данные из индикатора iDeMarker, код ошибки
        //--- завершим с нулевым результатом - это означает, что индикатор будет считат
        return(false);
    }
}

```

```
//--- все получилось
    return(true);
}
//+-----+
//| Indicator deinitialization function |
//+-----+
void OnDeinit(const int reason)
{
    //--- почистим график при удалении индикатора
    Comment("");
}
```

iEnvelopes

函数返回轨道线指标处理器。

```
int iEnvelopes(
    string          symbol,          // 交易品种名称
    ENUM_TIMEFRAMES period,          // 周期
    int             ma_period,        // 均线计算周期
    int             ma_shift,         // 指标平移
    ENUM_MA_METHOD  ma_method,        // 平滑类型
    ENUM_APPLIED_PRICE applied_price, // 价格或者处理器类型
    double          deviation         // 中线边界差(百分率)
);
```

参量

symbol

[in] 证券交易品种名称，数据用来计算指标。 [NULL](#) 值代表当前交易品种。

period

[in] 周期值可以是 [ENUM_TIMEFRAMES](#) 值中的一个，0代表当前时间表。

ma_period

[in] 主线平均周期

ma_shift

[in] 相关价格图表的指标转换。

ma_method

[in] 平滑类型。可以是 [ENUM_MA_METHOD](#) 值中的一个。

applied_price

[in] 使用价格。可以是任意 [ENUM_APPLIED_PRICE](#) 价格常量或者另外指标处理器。

deviation

[in] 偏离主线（百分比）。

返回值

返回特殊技术指标处理器，失败返回 [INVALID_HANDLE](#)。计算机内存从不使用的指标中释放，使用指标处理程序传递到的函数 [IndicatorRelease\(\)](#)。

注释

缓冲区代码：0 - UPPER_ LINE , 1 - LOWER_ LINE。

:

```
//+-----+
//|                                     Demo_iEnvelopes.mq5 |
//|                                     Copyright 2011, MetaQuotes Software Corp. |
//|                                     http://www.mql5.com |
//+-----+
```

```

#property copyright "Copyright 2011, MetaQuotes Software Corp."
#property link      "http://www.mql5.com"
#property version   "1.00"
#property description "Индикатор демонстрирует как нужно получать данные"
#property description "Индикаторных буферов для технического индикатора iEnvelopes."
#property description "Символ и таймфрейм, на котором рассчитывается индикатор,"
#property description "задаются параметрами symbol и period."
#property description "Способ создания хэндла задается параметром 'type' (тип функции)"

#property indicator_chart_window
#property indicator_buffers 2
#property indicator_plots 2
//--- построение Upper
#property indicator_label1 "Upper"
#property indicator_type1  DRAW_LINE
#property indicator_color1  clrBlue
#property indicator_style1  STYLE_SOLID
#property indicator_width1  1
//--- построение Lower
#property indicator_label2 "Lower"
#property indicator_type2  DRAW_LINE
#property indicator_color2  clrRed
#property indicator_style2  STYLE_SOLID
#property indicator_width2  1
//+-----+
//|  перечисление способов создания хэндла  |
//+-----+
enum Creation
{
    Call_iEnvelopes,      // использовать iEnvelopes
    Call_IndicatorCreate  // использовать IndicatorCreate
};
//--- входные параметры
input Creation          type=Call_iEnvelopes;      // тип функции
input int               ma_period=14;              // период скользящей
input int               ma_shift=0;                // смещение
input ENUM_MA_METHOD    ma_method=MODE_SMA;        // тип сглаживания
input ENUM_APPLIED_PRICE applied_price=PRICE_CLOSE; // тип цены
input double            deviation=0.1;              // отклонение границ от скользящей
input string            symbol=" ";                // символ
input ENUM_TIMEFRAMES   period=PERIOD_CURRENT;    // таймфрейм
//--- индикаторный буфер
double      UpperBuffer[];
double      LowerBuffer[];
//--- переменная для хранения хэндла индикатора iEnvelopes
int         handle;
//--- переменная для хранения
string name=symbol;

```

```

//--- имя индикатора на графике
string short_name;
//--- будем хранить количество значений в индикаторе Envelopes
int bars_calculated=0;
//+-----+
//| Custom indicator initialization function |
//+-----+
int OnInit()
{
//--- привязка массивов к индикаторным буферам
SetIndexBuffer(0,UpperBuffer,INDICATOR_DATA);
SetIndexBuffer(1,LowerBuffer,INDICATOR_DATA);
//--- зададим смещение для каждой линии
PlotIndexSetInteger(0,PLOT_SHIFT,ma_shift);
PlotIndexSetInteger(1,PLOT_SHIFT,ma_shift);
//--- определимся с символом, на котором строится индикатор
name=symbol;
//--- удалим пробелы слева и справа
StringTrimRight(name);
StringTrimLeft(name);
//--- если после этого длина строки name нулевая
if(StringLen(name)==0)
{
//--- возьмем символ с графика, на котором запущен индикатор
name=_Symbol;
}
//--- создадим хэндл индикатора
if(type==Call_iEnvelopes)
handle=iEnvelopes(name,period,ma_period,ma_shift,ma_method,applied_price,deviat
else
{
//--- заполним структуру значениями параметров индикатора
MqlParam pars[5];
//--- период средней
pars[0].type=TYPE_INT;
pars[0].integer_value=ma_period;
//--- смещение
pars[1].type=TYPE_INT;
pars[1].integer_value=ma_shift;
//--- тип сглаживания
pars[2].type=TYPE_INT;
pars[2].integer_value=ma_method;
//--- тип цены
pars[3].type=TYPE_INT;
pars[3].integer_value=applied_price;
//--- тип цены
pars[4].type=TYPE_DOUBLE;
pars[4].double_value=deviation;
}
}

```

```

        handle=IndicatorCreate(name,period,IND_ENVELOPES,5,pars);
    }
//--- если не удалось создать хэндл
    if(handle==INVALID_HANDLE)
    {
        //--- сообщим о неудаче и выведем номер ошибки
        PrintFormat("Не удалось создать хэндл индикатора iEnvelopes для пары %s/%s, код
                    name,
                    EnumToString(period),
                    GetLastError());
        //--- работа индикатора завершается досрочно при возврате отрицательного значения
        return(-1);
    }
//--- покажем на какой паре символ/таймфрейм рассчитан индикатор Envelopes
    short_name=StringFormat("iEnvelopes(%s/%s, %d, %d, %s,%s, %G)",name,EnumToString(p
    ma_period,ma_shift,EnumToString(ma_method),EnumToString(applied_price),deviation);
    IndicatorSetString(INDICATOR_SHORTNAME,short_name);
//--- нормальное выполнение инициализации индикатора
    return(0);
}

//+-----+
//| Custom indicator iteration function |
//+-----+
int OnCalculate(const int rates_total,
                const int prev_calculated,
                const datetime &time[],
                const double &open[],
                const double &high[],
                const double &low[],
                const double &close[],
                const long &tick_volume[],
                const long &volume[],
                const int &spread[])
{
//--- количество копируемых значений из индикатора iEnvelopes
    int values_to_copy;
//--- узнаем количество рассчитанных значений в индикаторе
    int calculated=BarsCalculated(handle);
    if(calculated<=0)
    {
        PrintFormat("BarsCalculated() вернул %d, код ошибки %d",calculated,GetLastError
        return(0);
    }
//--- если это первый запуск вычислений нашего индикатора или изменилось количество з
//--- или если необходимо рассчитать индикатор для двух или более баров (значит что-т
    if(prev_calculated==0 || calculated!=bars_calculated || rates_total>prev_calculate
    {
        //--- если массив UpperBuffer больше, чем значений в индикаторе iEnvelopes на п

```

```

    //--- в противном случае копировать будем меньше, чем размер индикаторных буфер
    if(calculated>rates_total) values_to_copy=rates_total;
    else
        values_to_copy=calculated;
    }
else
{
    //--- значит наш индикатор рассчитывается не в первый раз и с момента последнего
    //--- для расчета добавилось не более одного бара
    values_to_copy=(rates_total-prev_calculated)+1;
}

//--- заполняем массивы UpperBuffer и LowerBuffer значениями из индикатора Envelopes
//--- если FillArrayFromBuffer вернула false, значит данные не готовы - завершаем раб
if(!FillArraysFromBuffers(UpperBuffer,LowerBuffer,ma_shift,handle,values_to_copy))
//--- сформируем сообщение
string comm=StringFormat("%s ==> Обновлено значений в индикаторе %s: %d",
                          TimeToString(TimeCurrent(),TIME_DATE|TIME_SECONDS),
                          short_name,
                          values_to_copy);

//--- выведем на график служебное сообщение
Comment(comm);

//--- запомним количество значений в индикаторе Envelopes
bars_calculated=calculated;

//--- вернем значение prev_calculated для следующего вызова
return(rates_total);
}

//+-----+
//| Заполняем индикаторные буферы из индикатора iEnvelopes |
//+-----+

bool FillArraysFromBuffers(double &upper_values[], // индикаторный буфер верхней г
                          double &lower_values[], // индикаторный буфер нижней гр
                          int shift,              // смещение
                          int ind_handle,         // хэндл индикатора iEnvelopes
                          int amount              // количество копируемых значен
                          )
{
    //--- сбросим код ошибки
    ResetLastError();

    //--- заполняем часть массива UpperBuffer значениями из индикаторного буфера под инде
    if(CopyBuffer(ind_handle,0,-shift,amount,upper_values)<0)
    {
        //--- если копирование не удалось, сообщим код ошибки
        PrintFormat("Не удалось скопировать данные из индикатора iEnvelopes, код ошибки
        //--- завершим с нулевым результатом - это означает, что индикатор будет считат
        return(false);
    }

    //--- заполняем часть массива LowerBuffer значениями из индикаторного буфера под инде
    if(CopyBuffer(ind_handle,1,-shift,amount,lower_values)<0)
    {

```

```
//--- если копирование не удалось, сообщим код ошибки
PrintFormat("Не удалось скопировать данные из индикатора iEnvelopes, код ошибки");
//--- завершим с нулевым результатом - это означает, что индикатор будет считат
return(false);
}
//--- все получилось
return(true);
}
//+-----+
//| Indicator deinitialization function |
//+-----+
void OnDeinit(const int reason)
{
//--- почистим график при удалении индикатора
Comment("");
}
```


iForce

函数返回强力指数指标处理器。只有一个缓冲区。

```
int iForce(
    string          symbol,          // 交易品种名称
    ENUM_TIMEFRAMES period,          // 周期
    int             ma_period,        // 平均周期
    ENUM_MA_METHOD  ma_method,        // 平滑类型
    ENUM_APPLIED_VOLUME applied_volume // 计算的交易量类型
);
```

参量

symbol

[in] 证券交易品种名称，数据用来计算指标。 [NULL](#) 值代表当前交易品种。

period

[in] 周期值可以是 [ENUM_TIMEFRAMES](#) 值中的一个，0代表当前时间表。

ma_period

[in] 指标计算的平均周期。

ma_method

[in] 平滑类型。可以是 [ENUM_MA_METHOD](#) 值中的一个。

applied_volume

[in] 使用的交易量。可以是 [ENUM_APPLIED_VOLUME](#) 值中的一个。

返回值

返回特殊技术指标处理器，失败返回 [INVALID_HANDLE](#)。计算机内存从不使用的指标中释放，使用指标处理程序传递到的函数 [IndicatorRelease\(\)](#)。

:

```
//+-----+
//|                                     Demo_iForce.mq5 |
//|                                     Copyright 2011, MetaQuotes Software Corp. |
//|                                     http://www.mql5.com |
//+-----+
#property copyright "Copyright 2011, MetaQuotes Software Corp."
#property link      "http://www.mql5.com"
#property version   "1.00"
#property description "Индикатор демонстрирует как нужно получать данные"
#property description "индикаторных буферов для технического индикатора iForce."
#property description "Символ и таймфрейм, на котором рассчитывается индикатор,"
#property description "задаются параметрами symbol и period."
#property description "Способ создания хэндла задается параметром 'type' (тип функции"

#property indicator_separate_window
#property indicator_buffers 1
```

```

#property indicator_plots 1
//--- построение iForce
#property indicator_label1 "iForce"
#property indicator_type1 DRAW_LINE
#property indicator_color1 clrLightSeaGreen
#property indicator_style1 STYLE_SOLID
#property indicator_width1 1
//+-----+
//| перечисление способов создания хэнгла |
//+-----+
enum Creation
{
    Call_iForce,          // использовать iForce
    Call_IndicatorCreate  // использовать IndicatorCreate
};
//--- входные параметры
input Creation          type=Call_iForce;          // тип функции
input int               ma_period=13;              // период усреднения
input ENUM_MA_METHOD    ma_method=MODE_SMA;        // тип сглаживания
input ENUM_APPLIED_VOLUME applied_volume=VOLUME_TICK; // тип объема
input string            symbol=" ";                // символ
input ENUM_TIMEFRAMES   period=PERIOD_CURRENT;     // таймфрейм
//--- индикаторный буфер
double iForceBuffer[];
//--- переменная для хранения хэнгла индикатора iForce
int handle;
//--- переменная для хранения
string name=symbol;
//--- имя индикатора на графике
string short_name;
//--- будем хранить количество значений в индикаторе Force
int bars_calculated=0;
//+-----+
//| Custom indicator initialization function |
//+-----+
int OnInit()
{
    //--- привязка массива к индикаторному буферу
    SetIndexBuffer(0,iForceBuffer,INDICATOR_DATA);
    //--- определимся с символом, на котором строится индикатор
    name=symbol;
    //--- удалим пробелы слева и справа
    StringTrimRight(name);
    StringTrimLeft(name);
    //--- если после этого длина строки name нулевая
    if(StringLen(name)==0)
    {
        //--- возьмем символ с графика, на котором запущен индикатор

```

```

        name=_Symbol;
    }
//--- создадим хэндл индикатора
    if(type==Call_iForce)
        handle=iForce(name,period,ma_period,ma_method,applied_volume);
    else
    {
        //--- заполним структуру значениями параметров индикатора
        MqlParam pars[3];
        //--- период средней
        pars[0].type=TYPE_INT;
        pars[0].integer_value=ma_period;
        //--- тип сглаживания
        pars[1].type=TYPE_INT;
        pars[1].integer_value=ma_method;
        //--- тип объема
        pars[2].type=TYPE_INT;
        pars[2].integer_value=applied_volume;
        //--- тип цены
        handle=IndicatorCreate(name,period,IND_FORCE,3,pars);
    }
//--- если не удалось создать хэндл
    if(handle==INVALID_HANDLE)
    {
        //--- сообщим о неудаче и выведем номер ошибки
        PrintFormat("Не удалось создать хэндл индикатора iForce для пары %s/%s, код оши
                    name,
                    EnumToString(period),
                    GetLastError());
        //--- работа индикатора завершается досрочно при возврате отрицательного значен
        return(-1);
    }
//--- покажем на какой паре символ/таймфрейм рассчитан индикатор Force
    short_name=StringFormat("iForce(%s/%s, %d, %s, %s)",name,EnumToString(period),
                            ma_period,EnumToString(ma_method),EnumToString(applied_vol
    IndicatorSetString(INDICATOR_SHORTNAME,short_name);
//--- нормальное выполнение инициализации индикатора
    return(0);
}
//+-----+
//| Custom indicator iteration function |
//+-----+
int OnCalculate(const int rates_total,
                const int prev_calculated,
                const datetime &time[],
                const double &open[],
                const double &high[],
                const double &low[],

```

```

        const double &close[],
        const long &tick_volume[],
        const long &volume[],
        const int &spread[])
    {
//--- количество копируемых значений из индикатора iForce
        int values_to_copy;
//--- узнаем количество рассчитанных значений в индикаторе
        int calculated=BarsCalculated(handle);
        if(calculated<=0)
        {
            PrintFormat("BarsCalculated() вернул %d, код ошибки %d",calculated,GetLastError);
            return(0);
        }
//--- если это первый запуск вычислений нашего индикатора или изменилось количество з
//--- или если необходимо рассчитать индикатор для двух или более баров (значит что-т
        if(prev_calculated==0 || calculated!=bars_calculated || rates_total>prev_calculate
        {
            //--- если массив iForceBuffer больше, чем значений в индикаторе iForce на паре
            //--- в противном случае копировать будем меньше, чем размер индикаторных буфер
            if(calculated>rates_total) values_to_copy=rates_total;
            else
                values_to_copy=calculated;
        }
        else
        {
            //--- значит наш индикатор рассчитывается не в первый раз и с момента последнег
            //--- для расчета добавилось не более одного бара
            values_to_copy=(rates_total-prev_calculated)+1;
        }
//--- заполняем массив iForceBuffer значениями из индикатора Force
//--- если FillArrayFromBuffer вернула false, значит данные не готовы - завершаем раб
        if(!FillArrayFromBuffer(iForceBuffer,handle,values_to_copy)) return(0);
//--- сформируем сообщение
        string comm=StringFormat("%s ==> Обновлено значений в индикаторе %s: %d",
                                   TimeToString(TimeCurrent(),TIME_DATE|TIME_SECONDS),
                                   short_name,
                                   values_to_copy);
//--- выведем на график служебное сообщение
        Comment(comm);
//--- запомним количество значений в индикаторе Force
        bars_calculated=calculated;
//--- вернем значение prev_calculated для следующего вызова
        return(rates_total);
    }
//+-----+
//| Заполняем индикаторный буфер из индикатора iForce |
//+-----+
bool FillArrayFromBuffer(double &values[], // индикаторный буфер значений Force Inde

```

```

        int ind_handle,    // хэндл индикатора iForce
        int amount        // количество копируемых значений
    )

    {
//--- сбросим код ошибки
    ResetLastError();
//--- заполняем часть массива iForceBuffer значениями из индикаторного буфера под инд
    if(CopyBuffer(ind_handle,0,0,amount,values)<0)
    {
        //--- если копирование не удалось, сообщим код ошибки
        PrintFormat("Не удалось скопировать данные из индикатора iForce, код ошибки %d"
//--- завершим с нулевым результатом - это означает, что индикатор будет считат
        return(false);
    }
//--- все получилось
    return(true);
    }
//+-----+
//| Indicator deinitialization function |
//+-----+
void OnDeinit(const int reason)
{
//--- почистим график при удалении индикатора
    Comment("");
}

```

iFractals

函数返回分形学指标处理器。

```
int iFractals(
    string      symbol,      // 交易品种名称
    ENUM_TIMEFRAMES period   // 周期
);
```

参量

symbol

[in] 证券交易品种名称，数据用来计算指标。 [NULL](#) 值代表当前交易品种。

period

[in] 周期值可以是 [ENUM_TIMEFRAMES](#) 值中的一个，0代表当前时间表。

返回值

返回特殊技术指标处理器，失败返回 [INVALID_HANDLE](#)。计算机内存从不使用的指标中释放，使用指标处理程序传递到的函数 [IndicatorRelease\(\)](#)。

注释

缓冲区代码如下：0 - UPPER_ LINE，1 - LOWER_ LINE。

:

```
//+-----+
//|                                     Demo_iFractals.mq5 |
//|                                     Copyright 2011, MetaQuotes Software Corp. |
//|                                     http://www.mql5.com |
//+-----+

#property copyright "Copyright 2011, MetaQuotes Software Corp."
#property link      "http://www.mql5.com"
#property version   "1.00"
#property description "Индикатор демонстрирует как нужно получать данные"
#property description "индикаторных буферов для технического индикатора iFractals."
#property description "Символ и таймфрейм, на котором рассчитывается индикатор,"
#property description "задаются параметрами symbol и period."
#property description "Способ создания хэндла задается параметром 'type' (тип функции"

#property indicator_chart_window
#property indicator_buffers 1
#property indicator_plots 1
#property indicator_chart_window
#property indicator_buffers 2
#property indicator_plots 2
//--- построение FractalUp
#property indicator_label1 "FractalUp"
#property indicator_type1 DRAW_ARROW
#property indicator_color1 clrBlue
```

```

//--- построение FractalDown
#property indicator_label2 "FractalDown"
#property indicator_type2 DRAW_ARROW
#property indicator_color2 clrRed
//+-----+
//| перечисление способов создания хэнгла |
//+-----+
enum Creation
{
    Call_iFractals,          // использовать iFractals
    Call_IndicatorCreate     // использовать IndicatorCreate
};
//--- входные параметры
input Creation      type=Call_iFractals;          // тип функции
input string        symbol=" ";                  // символ
input ENUM_TIMEFRAMES period=PERIOD_CURRENT;     // таймфрейм
//--- индикаторные буферы
double      FractalUpBuffer[];
double      FractalDownBuffer[];
//--- переменная для хранения хэнгла индикатора iFractals
int         handle;
//--- переменная для хранения
string name=symbol;
//--- имя индикатора на графике
string short_name;
//--- будем хранить количество значений в индикаторе Fractals
int         bars_calculated=0;
//+-----+
//| Custom indicator initialization function |
//+-----+
int OnInit()
{
    //--- привязка массивов к индикаторным буферам
    SetIndexBuffer(0,FractalUpBuffer,INDICATOR_DATA);
    SetIndexBuffer(1,FractalDownBuffer,INDICATOR_DATA);
    //--- зададим коды символом из набора Wingdings для свойств PLOT_ARROW
    PlotIndexSetInteger(0,PLOT_ARROW,217); // стрелка вверх
    PlotIndexSetInteger(1,PLOT_ARROW,218); // стрелка вниз
    //--- определимся с символом, на котором строится индикатор
    name=symbol;
    //--- удалим пробелы слева и справа
    StringTrimRight(name);
    StringTrimLeft(name);
    //--- если после этого длина строки name нулевая
    if(StringLen(name)==0)
    {
        //--- возьмем символ с графика, на котором запущен индикатор
        name=_Symbol;
    }
}

```

```

    }
//--- создадим хэндл индикатора
    if(type==Call_iFractals)
        handle=iFractals(name,period);
    else
        handle=IndicatorCreate(name,period,IND_FRACTALS);
//--- если не удалось создать хэндл
    if(handle==INVALID_HANDLE)
    {
        //--- сообщим о неудаче и выведем номер ошибки
        PrintFormat("Не удалось создать хэндл индикатора iFractals для пары %s/%s, код
                    name,
                    EnumToString(period),
                    GetLastError());
        //--- работа индикатора завершается досрочно при возврате отрицательного значения
        return(-1);
    }
//--- покажем на какой паре символ/таймфрейм рассчитан индикатор Fractals
    short_name=StringFormat("iFractals(%s/%s)",name,EnumToString(period));
    IndicatorSetString(INDICATOR_SHORTNAME,short_name);
//--- нормальное выполнение инициализации индикатора
    return(0);
}
//+-----+
//| Custom indicator iteration function |
//+-----+
int OnCalculate(const int rates_total,
               const int prev_calculated,
               const datetime &time[],
               const double &open[],
               const double &high[],
               const double &low[],
               const double &close[],
               const long &tick_volume[],
               const long &volume[],
               const int &spread[])
{
//--- количество копируемых значений из индикатора iFractals
    int values_to_copy;
//--- узнаем количество рассчитанных значений в индикаторе
    int calculated=BarsCalculated(handle);
    if(calculated<=0)
    {
        PrintFormat("BarsCalculated() вернул %d, код ошибки %d",calculated,GetLastError());
        return(0);
    }
//--- если это первый запуск вычислений нашего индикатора или изменилось количество э
//--- или если необходимо рассчитать индикатор для двух или более баров (значит что-т

```



```

if(prev_calculated==0 || calculated!=bars_calculated || rates_total>prev_calculated)
{
    //--- если массив FractalUpBuffer больше, чем значений в индикаторе iFractals и
    //--- в противном случае копировать будем меньше, чем размер индикаторных буферов
    if(calculated>rates_total) values_to_copy=rates_total;
    else
        values_to_copy=calculated;
}
else
{
    //--- значит наш индикатор рассчитывается не в первый раз и с момента последнего
    //--- для расчета добавилось не более одного бара
    values_to_copy=(rates_total-prev_calculated)+1;
}
//--- заполняем массивы FractalUpBuffer и FractalDownBuffer значениями из индикатора
//--- если FillArrayFromBuffer вернула false, значит данные не готовы - завершаем работу
if(!FillArraysFromBuffers(FractalUpBuffer,FractalDownBuffer,handle,values_to_copy))
//--- сформируем сообщение
string comm=StringFormat("%s ==> Обновлено значений в индикаторе %s: %d",
                        TimeToString(TimeCurrent(),TIME_DATE|TIME_SECONDS),
                        short_name,
                        values_to_copy);
//--- выведем на график служебное сообщение
Comment(comm);
//--- запомним количество значений в индикаторе Fractals
bars_calculated=calculated;
//--- вернем значение prev_calculated для следующего вызова
return(rates_total);
}
//+-----+
//| Заполняем индикаторные буферы из индикатора iFractals |
//+-----+
bool FillArraysFromBuffers(double &up_arrows[],      // индикаторный буфер стрелок
                        double &down_arrows[],      // индикаторный буфер стрелок
                        int ind_handle,              // хэндл индикатора iFractals
                        int amount                   // количество копируемых значений
                        )
{
    //--- сбросим код ошибки
    ResetLastError();
    //--- заполняем часть массива FractalUpBuffer значениями из индикаторного буфера под индексом 0
    if(CopyBuffer(ind_handle,0,0,amount,up_arrows)<0)
    {
        //--- если копирование не удалось, сообщим код ошибки
        PrintFormat("Не удалось скопировать данные из индикатора iFractals в массив FractalUpBuffer");
        GetLastError();
        //--- завершим с нулевым результатом - это означает, что индикатор будет считаться неактивным
        return(false);
    }
}

```

```

//--- заполняем часть массива FractalDownBuffer значениями из индикаторного буфера по
if(CopyBuffer(ind_handle,1,0,amount,down_arrows)<0)
{
    //--- если копирование не удалось, сообщим код ошибки
    PrintFormat("Не удалось скопировать данные из индикатора iFractals в массив Fra
                GetLastError());

    //--- завершим с нулевым результатом - это означает, что индикатор будет считат
    return(false);
}
//--- все получилось
return(true);
}
//+-----+
//| Indicator deinitialization function |
//+-----+
void OnDeinit(const int reason)
{
    //--- почистим график при удалении индикатора
    Comment("");
}

```

iFrAMA

函数返回分形学适合移动平均指标处理器。只有一个缓冲区。

```
int iFrAMA(
    string          symbol,          // 交易品种名称
    ENUM_TIMEFRAMES period,          // 周期
    int             ma_period,        // 平均周期
    int             ma_shift,         // 图表平移
    ENUM_APPLIED_PRICE applied_price // 价格或者处理器类型
);
```

参量

symbol

[in] 证券交易品种名称，数据用来计算指标。 [NULL](#) 值代表当前交易品种。

period

[in] 周期值可以是 [ENUM_TIMEFRAMES](#) 值中的一个，0代表当前时间表。

ma_period

[in] 指标计算周期（计算柱）。

ma_shift

[in] 在价格图表中的指标转换。

applied_price

[in] 使用价格。可以是任意 [ENUM_APPLIED_PRICE](#) 价格常量或者另外指标处理器。

返回值

返回特殊技术指标处理器，失败返回 [INVALID_HANDLE](#)。计算机内存从不使用的指标中释放，使用指标处理程序传递到的函数 [IndicatorRelease\(\)](#)。

:

```
//+-----+
//|                                     Demo_iFrAMA.mq5 |
//|                                     Copyright 2011, MetaQuotes Software Corp. |
//|                                     http://www.mql5.com |
//+-----+
#property copyright "Copyright 2011, MetaQuotes Software Corp."
#property link      "http://www.mql5.com"
#property version   "1.00"
#property description "Индикатор демонстрирует как нужно получать данные"
#property description "индикаторных буферов для технического индикатора iFrAMA."
#property description "Символ и таймфрейм, на котором рассчитывается индикатор,"
#property description "задаются параметрами symbol и period."
#property description "Способ создания хэндла задается параметром 'type' (тип функции"

#property indicator_chart_window
#property indicator_buffers 1
```

```

#property indicator_plots 1
//--- построение iFrAMA
#property indicator_label1 "iFrAMA"
#property indicator_type1 DRAW_LINE
#property indicator_color1 clrBlue
#property indicator_style1 STYLE_SOLID
#property indicator_width1 1
//+-----+
//| перечисление способов создания хэндла |
//+-----+
enum Creation
{
    Call_iFrAMA,          // использовать iFrAMA
    Call_IndicatorCreate  // использовать IndicatorCreate
};
//--- входные параметры
input Creation          type=Call_iFrAMA;          // тип функции
input int               ma_period=14;              // период усреднения
input int               ma_shift=0;                // смещение
input ENUM_APPLIED_PRICE applied_price=PRICE_CLOSE; // тип цены
input string            symbol=" ";                // символ
input ENUM_TIMEFRAMES   period=PERIOD_CURRENT;    // таймфрейм
//--- индикаторный буфер
double iFrAMABuffer[];
//--- переменная для хранения хэндла индикатора iFrAMA
int handle;
//--- переменная для хранения
string name=symbol;
//--- имя индикатора на графике
string short_name;
//--- будем хранить количество значений в индикаторе Fractal Adaptive Moving Average
int bars_calculated=0;
//+-----+
//| Custom indicator initialization function |
//+-----+
int OnInit()
{
    //--- привязка массива к индикаторному буферу
    SetIndexBuffer(0,iFrAMABuffer,INDICATOR_DATA);
    //--- зададим смещение
    PlotIndexSetInteger(0,PLOT_SHIFT,ma_shift);
    //--- определимся с символом, на котором строится индикатор
    name=symbol;
    //--- удалим пробелы слева и справа
    StringTrimRight(name);
    StringTrimLeft(name);
    //--- если после этого длина строки name нулевая
    if(StringLen(name)==0)

```

```

    {
        //--- возьмем символ с графика, на котором запущен индикатор
        name=_Symbol;
    }
//--- создадим хэндл индикатора
if(type==Call_iFrAMA)
    handle=iFrAMA(name,period,ma_period,ma_shift,applied_price);
else
    {
        //--- заполним структуру значениями параметров индикатора
        MqlParam pars[3];
        //--- период средней
        pars[0].type=TYPE_INT;
        pars[0].integer_value=ma_period;
        //--- смещение
        pars[1].type=TYPE_INT;
        pars[1].integer_value=ma_shift;
        //--- тип цены
        pars[2].type=TYPE_INT;
        pars[2].integer_value=applied_price;
        //--- тип цены
        handle=IndicatorCreate(name,period,IND_FRAMA,3,pars);
    }
//--- если не удалось создать хэндл
if(handle==INVALID_HANDLE)
    {
        //--- сообщим о неудаче и выведем номер ошибки
        PrintFormat("Не удалось создать хэндл индикатора iFrAMA для пары %s/%s, код оши
            name,
            EnumToString(period),
            GetLastError());
        //--- работа индикатора завершается досрочно при возврате отрицательного значен
        return(-1);
    }
//--- покажем на какой паре символ/таймфрейм рассчитан индикатор iFrAMA
short_name=StringFormat("iFrAMA(%s/%s, %d, %d, %s)",name,EnumToString(period),
    ma_period,ma_shift,EnumToString(applied_price));
IndicatorSetString(INDICATOR_SHORTNAME,short_name);
//--- нормальное выполнение инициализации индикатора
return(0);
}
//+-----+
//| Custom indicator iteration function |
//+-----+
int OnCalculate(const int rates_total,
    const int prev_calculated,
    const datetime &time[],
    const double &open[],

```

```

        const double &high[],
        const double &low[],
        const double &close[],
        const long &tick_volume[],
        const long &volume[],
        const int &spread[])

    {
//--- количество копируемых значений из индикатора iFrAMA
        int values_to_copy;
//--- узнаем количество рассчитанных значений в индикаторе
        int calculated=BarsCalculated(handle);
        if(calculated<=0)
        {
            PrintFormat("BarsCalculated() вернул %d, код ошибки %d",calculated,GetLastError());
            return(0);
        }
//--- если это первый запуск вычислений нашего индикатора или изменилось количество з
//--- или если необходимо рассчитать индикатор для двух или более баров (значит что-т
        if(prev_calculated==0 || calculated!=bars_calculated || rates_total>prev_calculated)
        {
            //--- если массив iFrAMABuffer больше, чем значений в индикаторе iFrAMA на паре
            //--- в противном случае копировать будем меньше, чем размер индикаторных буфер
            if(calculated>rates_total) values_to_copy=rates_total;
            else
                values_to_copy=calculated;
        }
        else
        {
            //--- значит наш индикатор рассчитывается не в первый раз и с момента последнег
            //--- для расчета добавилось не более одного бара
            values_to_copy=(rates_total-prev_calculated)+1;
        }
//--- заполняем массив iFrAMABuffer значениями из индикатора Fractal Adaptive Moving
//--- если FillArrayFromBuffer вернула false, значит данные не готовы - завершаем раб
        if(!FillArrayFromBuffer(iFrAMABuffer,ma_shift,handle,values_to_copy)) return(0);
//--- сформируем сообщение
        string comm=StringFormat("%s ==> Обновлено значений в индикаторе %s: %d",
                                   TimeToString(TimeCurrent(),TIME_DATE|TIME_SECONDS),
                                   short_name,
                                   values_to_copy);
//--- выведем на график служебное сообщение
        Comment(comm);
//--- запомним количество значений в индикаторе Fractal Adaptive Moving Average
        bars_calculated=calculated;
//--- вернем значение prev_calculated для следующего вызова
        return(rates_total);
    }
//+-----+
//| Заполняем индикаторный буфер из индикатора iFrAMA |

```

```

//+-----+
bool FillArrayFromBuffer(double &values[], // индикаторный буфер значений Fractal Ad
                        int shift,         // смещение
                        int ind_handle,     // хэндл индикатора iFrAMA
                        int amount         // количество копируемых значений
                        )
{
//--- сбросим код ошибки
    ResetLastError();
//--- заполняем часть массива iFrAMABuffer значениями из индикаторного буфера под инд
    if(CopyBuffer(ind_handle,0,-shift,amount,values)<0)
    {
        //--- если копирование не удалось, сообщим код ошибки
        PrintFormat("Не удалось скопировать данные из индикатора iFrAMA, код ошибки %d"
        //--- завершим с нулевым результатом - это означает, что индикатор будет считат
        return(false);
    }
//--- все получилось
    return(true);
}
//+-----+
//| Indicator deinitialization function |
//+-----+
void OnDeinit(const int reason)
{
//--- почистим график при удалении индикатора
    Comment("");
}

```

iGator

函数返回鳄鱼振荡器指标处理器。振荡器表示鳄鱼指标蓝线和红线（上升柱状图）的区别以及红线和绿线的区别（下降柱状图）。

```
int iGator(
    string          symbol,          // 交易品种名称
    ENUM_TIMEFRAMES period,          // 周期
    int             jaw_period,       // 咽喉计算周期
    int             jaw_shift,        // 咽喉平移
    int             teeth_period,     // 牙齿计算周期
    int             teeth_shift,      // 牙齿平移
    int             lips_period,      // 唇部计算周期
    int             lips_shift,       // 唇部平移
    ENUM_MA_METHOD  ma_method,        // 平滑类型
    ENUM_APPLIED_PRICE applied_price // 价格或者处理器类型
);
```

参量

symbol

[in] 证券交易品种名称，数据用来计算指标。 [NULL](#) 值代表当前交易品种。

period

[in] 周期值可以是 [ENUM_TIMEFRAMES](#) 值中的一个，0代表当前时间表。

jaw_period

[in] 蓝线平均周期（鳄鱼颌骨）。

jaw_shift

[in] 与价格图表有关的蓝线变化。不能直接连接到指标柱状图的视觉变化。

teeth_period

[in] 红线的平均周期（鳄鱼牙）。

teeth_shift

[in] 关于价格图表的红线变化。不能直接连接到指标柱状图的视觉变化。

lips_period

[in] 绿线的平均周期（鳄鱼唇）。

lips_shift

[in] 关于价格图表的绿线变化。不能直接连接到指标柱状图的视觉变化。

ma_method

[in] 平滑类型。可以是 [ENUM_MA_METHOD](#) 值中的一个。

applied_price

[in] 使用价格。可以是任意 [ENUM_APPLIED_PRICE](#) 价格常量或者另外指标处理器。

返回值

返回特殊技术指标处理器，失败返回 `INVALID_HANDLE`。计算机内存从不使用的指标中释放，使用指标处理程序传递到的函数 `IndicatorRelease()`。

注释

缓冲区代码：0-UPPER_ HISTOGRAM，1-上升柱状图的颜色缓冲区，2 - LOWER_ HISTOGRAM，3 - 下降柱状图的颜色缓冲区。

```
//+-----+
//|                                     Demo_iGator.mq5 |
//|                                     Copyright 2011, MetaQuotes Software Corp. |
//|                                     http://www.mql5.com |
//+-----+
#property copyright "Copyright 2011, MetaQuotes Software Corp."
#property link      "http://www.mql5.com"
#property version   "1.00"
#property description "Индикатор демонстрирует как нужно получать данные"
#property description "индикаторных буферов для технического индикатора iGator."
#property description "Символ и таймфрейм, на котором рассчитывается индикатор,"
#property description "задаются параметрами symbol и period."
#property description "Способ создания хэндла задается параметром 'type' (тип функции)"
#property description "Все остальные параметры как в стандартном Gator Oscillator."

#property indicator_separate_window
#property indicator_buffers 4
#property indicator_plots 2
//--- построение GatorUp
#property indicator_label1 "GatorUp"
#property indicator_type1  DRAW_COLOR_HISTOGRAM
#property indicator_color1 clrGreen, clrRed
#property indicator_style1 STYLE_SOLID
#property indicator_width1 1
//--- построение GatorDown
#property indicator_label2 "GatorDown"
#property indicator_type2  DRAW_COLOR_HISTOGRAM
#property indicator_color2 clrGreen, clrRed
#property indicator_style2 STYLE_SOLID
#property indicator_width2 1
//+-----+
//| перечисление способов создания хэндла |
//+-----+
enum Creation
{
    Call_iGator,          // использовать iGator
    Call_IndicatorCreate  // использовать IndicatorCreate
};
//--- входные параметры
input Creation           type=Call_iGator;    // тип функции
```

```

input string          symbol=" ";           // символ
input ENUM_TIMEFRAMES period=PERIOD_CURRENT; // таймфрейм
input int             jaw_period=13;        // период для линии Челюстей
input int             jaw_shift=8;          // смещение линии Челюстей
input int             teeth_period=8;       // период для линии Зубов
input int             teeth_shift=5;        // смещение линии Челюстей
input int             lips_period=5;        // период для динии Губ
input int             lips_shift=3;         // смещение линии Губ
input ENUM_MA_METHOD  MA_method=MODE_SMMA; // метод усреднения линий Аллигатор
input ENUM_APPLIED_PRICE applied_price=PRICE_MEDIAN; // тип цены, от которой строится

//--- индикаторные буферы
double      GatorUpBuffer[];
double      GatorUpColors[];
double      GatorDownBuffer[];
double      GatorDownColors[];

//--- переменная для хранения хэндла индикатора iGator
int         handle;

//--- переменная для хранения
string name=symbol;

//--- имя индикатора на графике
string short_name;

//--- значения смещений для верхней и нижней гисторграммы
int shift;

//--- будем хранить количество значений в индикаторе Gator Oscillator
int bars_calculated=0;

//+-----+
//| Custom indicator initialization function |
//+-----+

int OnInit()
{
    //--- привязка массивов к индикаторным буферам
    SetIndexBuffer(0,GatorUpBuffer,INDICATOR_DATA);
    SetIndexBuffer(1,GatorUpColors,INDICATOR_COLOR_INDEX);
    SetIndexBuffer(2,GatorDownBuffer,INDICATOR_DATA);
    SetIndexBuffer(3,GatorDownColors,INDICATOR_COLOR_INDEX);

    /*
    Все указанные в параметрах смещения относятся к индикатору Alligator, по которому
    Поэтому указанные смещения не производят смещение самого индикатора Gator, а мешаю
    по значениям которого строятся значения индикатора Gator Oscillator!
    */

    //--- вычислим смещение для верхней и нижней гисторамм, которая является разницей ме
    shift=MathMin(jaw_shift,teeth_shift);
    PlotIndexSetInteger(0,PLOT_SHIFT,shift);

    //--- несмотря на то, что в индикаторе две гисторграммы, используется одинаковое смеще
    PlotIndexSetInteger(1,PLOT_SHIFT,shift);

    //--- определимся с символом, на котором строится индикатор
    name=symbol;

```

```

//--- удалим пробелы слева и справа
StringTrimRight(name);
StringTrimLeft(name);
//--- если после этого длина строки name нулевая
if(StringLen(name)==0)
{
    //--- возьмем символ с графика, на котором запущен индикатор
    name=_Symbol;
}
//--- создадим хэндл индикатора
if(type==Call_iGator)
    handle=iGator(name,period,jaw_period,jaw_shift,teeth_period,teeth_shift,
        lips_period,lips_shift,MA_method,applied_price);
else
{
    //--- заполним структуру значениями параметров индикатора
    MqlParam pars[8];
    //--- периоды и смещения линий Аллигатора
    pars[0].type=TYPE_INT;
    pars[0].integer_value=jaw_period;
    pars[1].type=TYPE_INT;
    pars[1].integer_value=jaw_shift;
    pars[2].type=TYPE_INT;
    pars[2].integer_value=teeth_period;
    pars[3].type=TYPE_INT;
    pars[3].integer_value=teeth_shift;
    pars[4].type=TYPE_INT;
    pars[4].integer_value=lips_period;
    pars[5].type=TYPE_INT;
    pars[5].integer_value=lips_shift;
    //--- тип сглаживания
    pars[6].type=TYPE_INT;
    pars[6].integer_value=MA_method;
    //--- тип цены
    pars[7].type=TYPE_INT;
    pars[7].integer_value=applied_price;
    //--- создадим хэндл
    handle=IndicatorCreate(name,period,IND_GATOR,8,pars);
}
//--- если не удалось создать хэндл
if(handle==INVALID_HANDLE)
{
    //--- сообщим о неудаче и выведем номер ошибки
    PrintFormat("Не удалось создать хэндл индикатора iGator для пары %s/%s, код оши
        name,
        EnumToString(period),
        GetLastError());
    //--- работа индикатора завершается досрочно при возврате отрицательного значен

```

```

        return(-1);
    }
    //--- покажем на какой паре символ/таймфрейм рассчитан индикатор Gator Oscillator
    short_name=StringFormat("iGator(%s/%s, %d, %d,%d, %d, %d, %d)",name,EnumToString(
        jaw_period,jaw_shift,teeth_period,teeth_shift,lips_period,
        IndicatorSetString(INDICATOR_SHORTNAME,short_name);
    //--- нормальное выполнение инициализации индикатора
    return(0);
}
//+-----+
//| Custom indicator iteration function |
//+-----+
int OnCalculate(const int rates_total,
                const int prev_calculated,
                const datetime &time[],
                const double &open[],
                const double &high[],
                const double &low[],
                const double &close[],
                const long &tick_volume[],
                const long &volume[],
                const int &spread[])
{
    //--- количество копируемых значений из индикатора iGator
    int values_to_copy;
    //--- узнаем количество рассчитанных значений в индикаторе
    int calculated=BarsCalculated(handle);
    if(calculated<=0)
    {
        PrintFormat("BarsCalculated() вернул %d, код ошибки %d",calculated,GetLastError());
        return(0);
    }
    //--- если это первый запуск вычислений нашего индикатора или изменилось количество з
    //--- или если необходимо рассчитать индикатор для двух или более баров (значит что-т
    if(prev_calculated==0 || calculated!=bars_calculated || rates_total>prev_calculate
    {
        //--- если массив GatorUpBuffer больше, чем значений в индикаторе iGator на пар
        //--- в противном случае копировать будем меньше, чем размер индикаторных буфер
        if(calculated>rates_total) values_to_copy=rates_total;
        else
            values_to_copy=calculated;
    }
    else
    {
        //--- значит наш индикатор рассчитывается не в первый раз и с момента последнег
        //--- для расчета добавилось не более одного бара
        values_to_copy=(rates_total-prev_calculated)+1;
    }
    //--- заполняем массивы значениями из индикатора Gator Oscillator

```

```

//--- если FillArraysFromBuffer вернула false, значит данные не готовы - завершаем ра
    if(!FillArraysFromBuffers(GatorUpBuffer,GatorUpColors,GatorDownBuffer,GatorDownCol
        shift,handle,values_to_copy)) return(0);
//--- сформируем сообщение
    string comm=StringFormat("%s ==> Обновлено значений в индикаторе %s: %d",
                            TimeToString(TimeCurrent(),TIME_DATE|TIME_SECONDS),
                            short_name,
                            values_to_copy);
//--- выведем на график служебное сообщение
    Comment(comm);
//--- заппомним количество значений в индикаторе Gator Oscillator
    bars_calculated=calculated;
//--- вернем значение prev_calculated для следующего вызова
    return(rates_total);
}
//+-----+
//| Заполняем индикаторные буферы из индикатора iGator |
//+-----+
bool FillArraysFromBuffers(double &ups_buffer[], // индикаторный буфер для ве
                        double &up_color_buffer[], // индикаторный буфер для ин
                        double &downs_buffer[], // индикаторный буфер для ин
                        double &downs_color_buffer[], // индикаторный буфер для ин
                        int u_shift, // смещение для верхней и ниж
                        int ind_handle, // хэндл индикатора iGator
                        int amount // количество копируемых зна

)

{
//--- сбросим код ошибки
    ResetLastError();
//--- заполняем часть массива GatorUpBuffer значениями из индикаторного буфера под ин
    if(CopyBuffer(ind_handle,0,-u_shift,amount,ups_buffer)<0)
    {
        //--- если копирование не удалось, сообщим код ошибки
        PrintFormat("Не удалось скопировать данные из индикатора iGator, код ошибки %d"
        //--- завершим с нулевым результатом - это означает, что индикатор будет считат
        return(false);
    }

//--- заполняем часть массива GatorUpColors значениями из индикаторного буфера под ин
    if(CopyBuffer(ind_handle,1,-u_shift,amount,up_color_buffer)<0)
    {
        //--- если копирование не удалось, сообщим код ошибки
        PrintFormat("Не удалось скопировать данные из индикатора iGator, код ошибки %d"
        //--- завершим с нулевым результатом - это означает, что индикатор будет считат
        return(false);
    }

//--- заполняем часть массива GatorDownBuffer значениями из индикаторного буфера под

```

```

if(CopyBuffer(ind_handle,2,-u_shift,amount,downs_buffer)<0)
{
    //--- если копирование не удалось, сообщим код ошибки
    PrintFormat("Не удалось скопировать данные из индикатора iGator, код ошибки %d"
    //--- завершим с нулевым результатом - это означает, что индикатор будет считат
    return(false);
}

//--- заполняем часть массива GatorDownColors значениями из индикаторного буфера под
if(CopyBuffer(ind_handle,3,-u_shift,amount,downs_color_buffer)<0)
{
    //--- если копирование не удалось, сообщим код ошибки
    PrintFormat("Не удалось скопировать данные из индикатора iGator, код ошибки %d"
    //--- завершим с нулевым результатом - это означает, что индикатор будет считат
    return(false);
}
//--- все получилось
return(true);
}

//+-----+
//| Indicator deinitialization function |
//+-----+
void OnDeinit(const int reason)
{
    //--- почистим график при удалении индикатора
    Comment("");
}

```

ilchimoku

函数返回一目均衡图指标处理器。

```
int iIchimoku(
    string          symbol,          // 交易品种类型
    ENUM_TIMEFRAMES period,         // 周期
    int             tenkan_sen,      // Tenkan-sen转换线周期
    int             kijun_sen,       // Kijun-sen基准线周期
    int             senkou_span_b    // Senkou Span B周期
);
```

参量

symbol

[in] 证券交易品种名称，数据用来计算指标。 [NULL](#) 值代表当前交易品种。

period

[in] 周期值可以是 [ENUM_TIMEFRAMES](#) 值中的一个，0代表当前时间表。

tenkan_sen

[in] 转换线平均周期。

kijun_sen

[in] 基准线平均周期。

senkou_span_b

[in] Senkou Span B 平均周期。

返回值

返回特殊技术指标处理器，失败返回 [INVALID_HANDLE](#)。计算机内存从不使用的指标中释放，使用指标处理程序传递到的函数 [IndicatorRelease\(\)](#)。

注释

缓冲区代码：0 - TENKANSEN_ LINE，1 - KIJUNSEN_ LINE，2 - SENKOUSPANA_ LINE，3 - SENKOUSPANB_ LINE，4 - CHINKOSPAN_ LINE。

:

```
//+-----+
//|                                     Demo_iIchimoku.mq5 |
//|                                     Copyright 2011, MetaQuotes Software Corp. |
//|                                     http://www.mql5.com |
//+-----+
#property copyright "Copyright 2011, MetaQuotes Software Corp."
#property link      "http://www.mql5.com"
#property version   "1.00"
#property description "Индикатор демонстрирует как нужно получать данные"
#property description "индикаторных буферов для технического индикатора iIchimoku."
#property description "Символ и таймфрейм, на котором рассчитывается индикатор,"
#property description "задаются параметрами symbol и period."
```

```

#property description "Способ создания хэндла задается параметром 'type' (тип функции"
#property description "Все остальные параметры как в стандартном Ichimoku Kinko Hyo."

#property indicator_chart_window
#property indicator_buffers 5
#property indicator_plots 4
//--- построение Tenkan_sen
#property indicator_label1 "Tenkan_sen"
#property indicator_type1 DRAW_LINE
#property indicator_color1 clrRed
#property indicator_style1 STYLE_SOLID
#property indicator_width1 1
//--- построение Kijun_sen
#property indicator_label2 "Kijun_sen"
#property indicator_type2 DRAW_LINE
#property indicator_color2 clrBlue
#property indicator_style2 STYLE_SOLID
#property indicator_width2 1
//--- построение Senkou_Span
#property indicator_label3 "Senkou Span A;Senkou Span B" // в Data Window будет пока
#property indicator_type3 DRAW_FILLING
#property indicator_color3 clrSandyBrown, clrThistle
#property indicator_style3 STYLE_SOLID
#property indicator_width3 1
//--- построение Chinkou_Span
#property indicator_label4 "Chinkou_Span"
#property indicator_type4 DRAW_LINE
#property indicator_color4 clrLime
#property indicator_style4 STYLE_SOLID
#property indicator_width4 1
//+-----+
//| перечисление способов создания хэндла |
//+-----+
enum Creation
{
    Call_iIchimoku,          // использовать iIchimoku
    Call_IndicatorCreate     // использовать IndicatorCreate
};
//--- входные параметры
input Creation      type=Call_iIchimoku;          // тип функции
input int           tenkan_sen=9;                  // период Tenkan-sen
input int           kijun_sen=26;                  // период Kijun-sen
input int           senkou_span_b=52;              // период Senkou Span B
input string        symbol=" ";                   // символ
input ENUM_TIMEFRAMES period=PERIOD_CURRENT;      // таймфрейм
//--- индикаторный буфер
double              Tenkan_sen_Buffer[];
double              Kijun_sen_Buffer[];

```



```

double      Senkou_Span_A_Buffer[];
double      Senkou_Span_B_Buffer[];
double      Chinkou_Span_Buffer[];
//--- переменная для хранения хэнгла индикатора iIchimoku
int         handle;
//--- переменная для хранения
string name=symbol;
//--- имя индикатора на графике
string short_name;
//--- будем хранить количество значений в индикаторе Ichimoku Kinko Hyo
int         bars_calculated=0;
//+-----+
//| Custom indicator initialization function |
//+-----+
int OnInit()
{
//--- привязка массивов к индикаторным буферам
SetIndexBuffer(0,Tenkan_sen_Buffer,INDICATOR_DATA);
SetIndexBuffer(1,Kijun_sen_Buffer,INDICATOR_DATA);
SetIndexBuffer(2,Senkou_Span_A_Buffer,INDICATOR_DATA);
SetIndexBuffer(3,Senkou_Span_B_Buffer,INDICATOR_DATA);
SetIndexBuffer(4,Chinkou_Span_Buffer,INDICATOR_DATA);
//--- зададим смещения для канала Senkou Span на kijun_sen баров в будущее
PlotIndexSetInteger(2,PLOT_SHIFT,kijun_sen);
//--- для линии Chinkou Span смещение задавать не требуется, так как данные Chinkou S
//--- хранятся в индикаторе iIchimoku уже со смещением
//--- определимся с символом, на котором строится индикатор
name=symbol;
//--- удалим пробелы слева и справа
StringTrimRight(name);
StringTrimLeft(name);
//--- если после этого длина строки name нулевая
if(StringLen(name)==0)
{
//--- возьмем символ с графика, на котором запущен индикатор
name=_Symbol;
}
//--- создадим хэнгл индикатора
if(type==Call_iIchimoku)
handle=iIchimoku(name,period,tenkan_sen,kijun_sen,senkou_span_b);
else
{
//--- заполним структуру значениями параметров индикатора
MqlParam pars[3];
//--- периоды и смещения линий Аллигатора
pars[0].type=TYPE_INT;
pars[0].integer_value=tenkan_sen;
pars[1].type=TYPE_INT;

```

```

    pars[1].integer_value=kijun_sen;
    pars[2].type=TYPE_INT;
    pars[2].integer_value=senkou_span_b;
    //--- создадим хэндл
    handle=IndicatorCreate(name,period,IND_ICHIMOKU,3,pars);
}
//--- если не удалось создать хэндл
if(handle==INVALID_HANDLE)
{
    //--- сообщим о неудаче и выведем номер ошибки
    PrintFormat("Не удалось создать хэндл индикатора iIchimoku для пары %s/%s, код
                name,
                EnumToString(period),
                GetLastError());
    //--- работа индикатора завершается досрочно при возврате отрицательного значения
    return(-1);
}
//--- покажем на какой паре символ/таймфрейм рассчитан индикатор Ichimoku Kinko Hyo
short_name=StringFormat("iIchimoku(%s/%s, %d, %d ,%d)",name,EnumToString(period),
                        tenkan_sen,kijun_sen,senkou_span_b);
IndicatorSetString(INDICATOR_SHORTNAME,short_name);
//--- нормальное выполнение инициализации индикатора
return(0);
}
//+-----+
//| Custom indicator iteration function |
//+-----+
int OnCalculate(const int rates_total,
               const int prev_calculated,
               const datetime &time[],
               const double &open[],
               const double &high[],
               const double &low[],
               const double &close[],
               const long &tick_volume[],
               const long &volume[],
               const int &spread[])
{
    //--- количество копируемых значений из индикатора iIchimoku
    int values_to_copy;
    //--- узнаем количество рассчитанных значений в индикаторе
    int calculated=BarsCalculated(handle);
    if(calculated<=0)
    {
        PrintFormat("BarsCalculated() вернул %d, код ошибки %d",calculated,GetLastError);
        return(0);
    }
    //--- если это первый запуск вычислений нашего индикатора или изменилось количество з

```

```

//--- или если необходимо рассчитать индикатор для двух или более баров (значит что-т
    if(prev_calculated==0 || calculated!=bars_calculated || rates_total>prev_calculate
    {
        //--- если массив Tenkan_sen_Buffer больше, чем значений в индикаторе iIchimoku
        //--- в противном случае копировать будем меньше, чем размер индикаторных буфер
        if(calculated>rates_total) values_to_copy=rates_total;
        else
            values_to_copy=calculated;
    }
    else
    {
        //--- значит наш индикатор рассчитывается не в первый раз и с момента последнег
        //--- для расчета добавилось не более одного бара
        values_to_copy=(rates_total-prev_calculated)+1;
    }
//--- заполняем массивы значениями из индикатора Ichimoku Kinko Hyo
//--- если FillArraysFromBuffer вернула false, значит данные не готовы - завершаем ра
    if(!FillArraysFromBuffers(Tenkan_sen_Buffer,Kijun_sen_Buffer,Senkou_Span_A_Buffer,
        kijun_sen,handle,values_to_copy)) return(0);
//--- сформируем сообщение
    string comm=StringFormat("%s ==> Обновлено значений в индикаторе %s: %d",
        TimeToString(TimeCurrent(),TIME_DATE|TIME_SECONDS),
        short_name,
        values_to_copy);
//--- выведем на график служебное сообщение
    Comment(comm);
//--- запоним количество значений в индикаторе Ichimoku Kinko Hyo
    bars_calculated=calculated;
//--- вернем значение prev_calculated для следующего вызова
    return(rates_total);
}

//+-----+
//| Заполняем индикаторные буферы из индикатора iIchimoku |
//+-----+
bool FillArraysFromBuffers(double &tenkan_sen_buffer[], // индикаторный буфер лин
    double &kijun_sen_buffer[], // индикаторный буфер лин
    double &senkou_span_A_buffer[], // индикаторный буфер лин
    double &senkou_span_B_buffer[], // индикаторный буфер лин
    double &chinkou_span_buffer[], // индикаторный буфер Chi
    int senkou_span_shift, // смещение линий Senkou
    int ind_handle, // хэндл индикатора iIchi
    int amount // количество копируемых
)
{
    //--- сбросим код ошибки
    ResetLastError();
//--- заполняем часть массива Tenkan_sen_Buffer значениями из индикаторного буфера по
    if(CopyBuffer(ind_handle,0,0,amount,tenkan_sen_buffer)<0)
    {

```

```

    //--- если копирование не удалось, сообщим код ошибки
    PrintFormat("1.Не удалось скопировать данные из индикатора iGator, код ошибки %
    //--- завершим с нулевым результатом - это означает, что индикатор будет считат
    return(false);
}

//--- заполняем часть массива Kijun_sen_Buffer значениями из индикаторного буфера под
if(CopyBuffer(ind_handle,1,0,amount,kijun_sen_buffer)<0)
{
    //--- если копирование не удалось, сообщим код ошибки
    PrintFormat("2.Не удалось скопировать данные из индикатора iGator, код ошибки %
    //--- завершим с нулевым результатом - это означает, что индикатор будет считат
    return(false);
}

//--- заполняем часть массива Chinkou_Span_Buffer значениями из индикаторного буфера
//--- при senkou_span_shift>0 линия смещена в будущее на senkou_span_shift баров
if(CopyBuffer(ind_handle,2,-senkou_span_shift,amount,senkou_span_A_buffer)<0)
{
    //--- если копирование не удалось, сообщим код ошибки
    PrintFormat("3.Не удалось скопировать данные из индикатора iGator, код ошибки %
    //--- завершим с нулевым результатом - это означает, что индикатор будет считат
    return(false);
}

//--- заполняем часть массива Senkou_Span_A_Buffer значениями из индикаторного буфера
//--- при senkou_span_shift>0 линия смещена в будущее на senkou_span_shift баров
if(CopyBuffer(ind_handle,3,-senkou_span_shift,amount,senkou_span_B_buffer)<0)
{
    //--- если копирование не удалось, сообщим код ошибки
    PrintFormat("4.Не удалось скопировать данные из индикатора iGator, код ошибки %
    //--- завершим с нулевым результатом - это означает, что индикатор будет считат
    return(false);
}

//--- заполняем часть массива Senkou_Span_B_Buffer значениями из индикаторного буфера
//--- при копировании Chinkou Span смещение учитывать не требуется, так как данные Ch
//--- хранятся в индикаторе iIchimoku уже со смещением
if(CopyBuffer(ind_handle,4,0,amount,chinkou_span_buffer)<0)
{
    //--- если копирование не удалось, сообщим код ошибки
    PrintFormat("5.Не удалось скопировать данные из индикатора iGator, код ошибки %
    //--- завершим с нулевым результатом - это означает, что индикатор будет считат
    return(false);
}

//--- все получилось
return(true);
}

```

```
//+-----+
//| Indicator deinitialization function |
//+-----+
void OnDeinit(const int reason)
{
//--- почистим график при удалении индикатора
    Comment("");
}
```

iBWMFI

函数返回市场便利指标处理器。只有一个缓冲区。

```
int iBWMFI(
    string          symbol,          // 交易品种类型
    ENUM_TIMEFRAMES period,          // 周期
    ENUM_APPLIED_VOLUME applied_volume // 计算的交易量类型
);
```

参量

symbol

[in] 证券交易品种名称，数据用来计算指标。 [NULL](#) 值代表当前交易品种。

period

[in] 周期值可以是 [ENUM_TIMEFRAMES](#) 值中的一个，0代表当前时间表。

applied_volume

[in] 使用的交易量。可以是 [ENUM_APPLIED_VOLUME](#) 常量中的一个。

返回值

返回特殊技术指标处理器，失败返回 [INVALID_HANDLE](#)。计算机内存从不使用的指标中释放，使用指标处理程序传递到的函数 [IndicatorRelease\(\)](#)。

:

```
//+-----+
//|                                     Demo_iBWMFI.mq5 |
//|                                     Copyright 2011, MetaQuotes Software Corp. |
//|                                     http://www.mql5.com |
//+-----+

#property copyright "Copyright 2011, MetaQuotes Software Corp."
#property link      "http://www.mql5.com"
#property version   "1.00"
#property description "Индикатор демонстрирует как нужно получать данные"
#property description "индикаторных буферов для технического индикатора iBWMFI."
#property description "Символ и таймфрейм, на котором рассчитывается индикатор,"
#property description "задаются параметрами symbol и period."
#property description "Способ создания хэндла задается параметром 'type' (тип функции"

#property indicator_separate_window
#property indicator_buffers 2
#property indicator_plots 1
//--- построение iBWMFI
#property indicator_label1 "iBWMFI"
#property indicator_type1  DRAW_COLOR_HISTOGRAM
#property indicator_color1 clrLime,clrSaddleBrown,clrBlue,clrPink
#property indicator_style1 STYLE_SOLID
#property indicator_width1 1
```

```

//+-----+
//|  перечисление способов создания хэндла |
//+-----+
enum Creation
{
    Call_iBWMFI,          // использовать iBWMFI
    Call_IndicatorCreate   // использовать IndicatorCreate
};
//--- входные параметры
input Creation           type=Call_iBWMFI;          // тип функции
input ENUM_APPLIED_VOLUME applied_volume=VOLUME_TICK; // тип объема
input string             symbol=" ";               // символ
input ENUM_TIMEFRAMES    period=PERIOD_CURRENT;    // таймфрейм
//--- индикаторный буфер
double      iBWMFIBuffer[];
double      iBWMFIColors[];
//--- переменная для хранения хэндла индикатора iBWMFI
int         handle;
//--- переменная для хранения
string name=symbol;
//--- имя индикатора на графике
string short_name;
//--- будем хранить количество значений в индикаторе Market Facilitation Index Билла
int         bars_calculated=0;
//+-----+
//| Custom indicator initialization function |
//+-----+
int OnInit()
{
    //--- привязка массивов к индикаторным буферам
    SetIndexBuffer(0,iBWMFIBuffer,INDICATOR_DATA);
    SetIndexBuffer(1,iBWMFIColors,INDICATOR_COLOR_INDEX);
    //--- определимся с символом, на котором строится индикатор
    name=symbol;
    //--- удалим пробелы слева и справа
    StringTrimRight(name);
    StringTrimLeft(name);
    //--- если после этого длина строки name нулевая
    if(StringLen(name)==0)
    {
        //--- возьмем символ с графика, на котором запущен индикатор
        name=_Symbol;
    }
    //--- создадим хэндл индикатора
    if(type==Call_iBWMFI)
        handle=iBWMFI(name,period,applied_volume);
    else
    {

```

```

    //--- заполним структуру значениями параметров индикатора
    MqlParam pars[1];
    //--- тип объема
    pars[0].type=TYPE_INT;
    pars[0].integer_value=applied_volume;
    handle=IndicatorCreate(name,period,IND_BWMFI,1,pars);
}
//--- если не удалось создать хэндл
if(handle==INVALID_HANDLE)
{
    //--- сообщим о неудаче и выведем номер ошибки
    PrintFormat("Не удалось создать хэндл индикатора iIchimoku для пары %s/%s, код
                name,
                EnumToString(period),
                GetLastError());
    //--- работа индикатора завершается досрочно при возврате отрицательного значения
    return(-1);
}
//--- покажем на какой паре символ/таймфрейм рассчитан индикатор Market Facilitation
short_name=StringFormat("iBWMFI(%s/%s, %s)",name,EnumToString(period),
                        EnumToString(applied_volume));
IndicatorSetString(INDICATOR_SHORTNAME,short_name);
//--- нормальное выполнение инициализации индикатора
return(0);
}
//+-----+
//| Custom indicator iteration function |
//+-----+
int OnCalculate(const int rates_total,
               const int prev_calculated,
               const datetime &time[],
               const double &open[],
               const double &high[],
               const double &low[],
               const double &close[],
               const long &tick_volume[],
               const long &volume[],
               const int &spread[])
{
    //--- количество копируемых значений из индикатора iBWMFI
    int values_to_copy;
    //--- узнаем количество рассчитанных значений в индикаторе
    int calculated=BarsCalculated(handle);
    if(calculated<=0)
    {
        PrintFormat("BarsCalculated() вернул %d, код ошибки %d",calculated,GetLastError);
        return(0);
    }
}

```



```

//--- если это первый запуск вычислений нашего индикатора или изменилось количество з
//--- или если необходимо рассчитать индикатор для двух или более баров (значит что-т
    if(prev_calculated==0 || calculated!=bars_calculated || rates_total>prev_calculate
    {
        //--- если массив Tenkan_sen_Buffer больше, чем значений в индикаторе iBWMFI на
        //--- в противном случае копировать будем меньше, чем размер индикаторных буфер
        if(calculated>rates_total) values_to_copy=rates_total;
        else
            values_to_copy=calculated;
    }
    else
    {
        //--- значит наш индикатор рассчитывается не в первый раз и с момента последнет
        //--- для расчета добавилось не более одного бара
        values_to_copy=(rates_total-prev_calculated)+1;
    }

//--- заполняем массивы значениями из индикатора Market Facilitation Index Билла Вилья
//--- если FillArraysFromBuffer вернула false, значит данные не готовы - завершаем ра
    if(!FillArraysFromBuffers(iBWMFIBuffer,iBWMFIColors,handle,values_to_copy)) return
//--- сформируем сообщение
    string comm=StringFormat("%s ==> Обновлено значений в индикаторе %s: %d",
                            TimeToString(TimeCurrent(),TIME_DATE|TIME_SECONDS),
                            short_name,
                            values_to_copy);

//--- выведем на график служебное сообщение
    Comment(comm);

//--- запоним количество значений в индикаторе Market Facilitation Index Билла Вилья
    bars_calculated=calculated;

//--- вернем значение prev_calculated для следующего вызова
    return(rates_total);
}

//+-----+
//| Заполняем индикаторные буферы из индикатора iBWMFI |
//+-----+

bool FillArraysFromBuffers(double &values[], // индикаторный буфер значений гистогра
                           double &colors[], // индикаторный буфер цветов гистогра
                           int ind_handle, // хэндл индикатора iBWMFI
                           int amount // количество копируемых значений
                           )
{
//--- сбросим код ошибки
    ResetLastError();

//--- заполняем часть массива iBWMFIBuffer значениями из индикаторного буфера под инд
    if(CopyBuffer(ind_handle,0,0,amount,values)<0)
    {
        //--- если копирование не удалось, сообщим код ошибки
        PrintFormat("Не удалось скопировать данные из индикатора iBWMFI, код ошибки %d"
        //--- завершим с нулевым результатом - это означает, что индикатор будет считат
        return(false);
    }
}

```

```

    }
    //--- заполняем часть массива iBWMFIColors значениями из индикаторного буфера под инд
    if(CopyBuffer(ind_handle,1,0,amount,colors)<0)
    {
        //--- если копирование не удалось, сообщим код ошибки
        PrintFormat("Не удалось скопировать данные из индикатора iBWMFI, код ошибки %d"
        //--- завершим с нулевым результатом - это означает, что индикатор будет считат
        return(false);
    }
    //--- все получилось
    return(true);
}
//+-----+
//| Indicator deinitialization function |
//+-----+
void OnDeinit(const int reason)
{
    //--- почистим график при удалении индикатора
    Comment("");
}

```

iMomentum

函数返回动量指标处理器。只有一个缓冲区。

```
int iMomentum(
    string          symbol,          // 交易品种名称
    ENUM_TIMEFRAMES period,          // 周期
    int             mom_period,       // 平均周期
    ENUM_APPLIED_PRICE applied_price // 价格或者处理器类型
);
```

参量

symbol

[in] 证券交易品种名称，数据用来计算指标。 [NULL](#) 值代表当前交易品种。

period

[in] 周期值可以是 [ENUM_TIMEFRAMES](#) 值中的一个，0代表当前时间表。

mom_period

[in] 计算价格变化的平均周期（计算柱）。

applied_price

[in] 使用价格，可以是任意 [ENUM_APPLIED_PRICE](#) 价格常量或者另外指标处理器。

返回值

返回特殊技术指标处理器，失败返回 [INVALID_HANDLE](#)。计算机内存从不使用的指标中释放，使用指标处理程序传递到的函数 [IndicatorRelease\(\)](#)。

:

```
//+-----+
//|                                     Demo_iMomentum.mq5 |
//|                                     Copyright 2011, MetaQuotes Software Corp. |
//|                                     http://www.mql5.com |
//+-----+

#property copyright "Copyright 2011, MetaQuotes Software Corp."
#property link      "http://www.mql5.com"
#property version   "1.00"
#property description "Индикатор демонстрирует как нужно получать данные"
#property description "индикаторных буферов для технического индикатора iMomentum."
#property description "Символ и таймфрейм, на котором рассчитывается индикатор,"
#property description "задаются параметрами symbol и period."
#property description "Способ создания хэндла задается параметром 'type' (тип функции"
#property description "Все остальные параметры как в стандартном Momentum."

#property indicator_separate_window
#property indicator_buffers 1
#property indicator_plots 1
//--- plot iMomentum
#property indicator_label1 "iMomentum"
```

```

#property indicator_type1    DRAW_LINE
#property indicator_color1   clrDodgerBlue
#property indicator_style1   STYLE_SOLID
#property indicator_width1   1

//+-----+
//|  перечисление способов создания хэнгла                                |
//+-----+
enum Creation
{
    Call_iMomentum,          // использовать iMomentum
    Call_IndicatorCreate      // использовать IndicatorCreate
};

//--- входные параметры
input Creation              type=Call_iMomentum;          // тип функции
input int                   mom_period=14;                 // период моментума
input ENUM_APPLIED_PRICE    applied_price=PRICE_CLOSE;     // тип цены
input string                 symbol=" ";                   // символ
input ENUM_TIMEFRAMES        period=PERIOD_CURRENT;        // таймфрейм

//--- индикаторный буфер
double                       iMomentumBuffer[];

//--- переменная для хранения хэнгла индикатора iMomentum
int                           handle;

//--- переменная для хранения
string name=symbol;

//--- имя индикатора на графике
string short_name;

//--- будем хранить количество значений в индикаторе Momentum
int                           bars_calculated=0;

//+-----+
//| Custom indicator initialization function                                |
//+-----+
int OnInit()
{
    //--- привязка массива к индикаторному буферу
    SetIndexBuffer(0,iMomentumBuffer,INDICATOR_DATA);

    //--- определимся с символом, на котором строится индикатор
    name=symbol;

    //--- удалим пробелы слева и справа
    StringTrimRight(name);
    StringTrimLeft(name);

    //--- если после этого длина строки name нулевая
    if(StringLen(name)==0)
    {
        //--- возьмем символ с графика, на котором запущен индикатор
        name=_Symbol;
    }

    //--- создадим хэнгл индикатора
    if(type==Call_iMomentum)

```

```

        handle=iMomentum(name,period,mom_period,applied_price);
    else
    {
        //--- заполним структуру значениями параметров индикатора
        MqlParam pars[2];
        //--- период
        pars[0].type=TYPE_INT;
        pars[0].integer_value=mom_period;
        //--- тип цены
        pars[1].type=TYPE_INT;
        pars[1].integer_value=applied_price;
        handle=IndicatorCreate(name,period,IND_MOMENTUM,2,pars);
    }
//--- если не удалось создать хэндл
if(handle==INVALID_HANDLE)
{
    //--- сообщим о неудаче и выведем номер ошибки
    PrintFormat("Не удалось создать хэндл индикатора iMomentum для пары %s/%s, код
                name,
                EnumToString(period),
                GetLastError());
    //--- работа индикатора завершается досрочно при возврате отрицательного значения
    return(-1);
}
//--- покажем на какой паре символ/таймфрейм рассчитан индикатор Momentum
short_name=StringFormat("iMomentum(%s/%s, %d, %s)",name,EnumToString(period),
                        mom_period, EnumToString(applied_price));
IndicatorSetString(INDICATOR_SHORTNAME,short_name);
//--- нормальное выполнение инициализации индикатора
return(0);
}
//+-----+
//| Custom indicator iteration function |
//+-----+
int OnCalculate(const int rates_total,
                const int prev_calculated,
                const datetime &time[],
                const double &open[],
                const double &high[],
                const double &low[],
                const double &close[],
                const long &tick_volume[],
                const long &volume[],
                const int &spread[])
{
    //--- количество копируемых значений из индикатора iMomentum
    int values_to_copy;
    //--- узнаем количество рассчитанных значений в индикаторе

```

```

int calculated=BarsCalculated(handle);
if(calculated<=0)
{
    PrintFormat("BarsCalculated() вернул %d, код ошибки %d",calculated,GetLastError());
    return(0);
}

//--- если это первый запуск вычислений нашего индикатора или изменилось количество э
//--- или если необходимо рассчитать индикатор для двух или более баров (значит что-т
if(prev_calculated==0 || calculated!=bars_calculated || rates_total>prev_calculated)
{
    //--- если массив iMomentumBuffer больше, чем значений в индикаторе iMomentum н
    //--- в противном случае копировать будем меньше, чем размер индикаторных буфер
    if(calculated>rates_total) values_to_copy=rates_total;
    else
        values_to_copy=calculated;
}
else
{
    //--- значит наш индикатор рассчитывается не в первый раз и с момента последнег
    //--- для расчета добавилось не более одного бара
    values_to_copy=(rates_total-prev_calculated)+1;
}

//--- заполняем массив iMomentumBuffer значениями из индикатора Momentum
//--- если FillArrayFromBuffer вернула false, значит данные не готовы - завершаем раб
if(!FillArrayFromBuffer(iMomentumBuffer,handle,values_to_copy)) return(0);
//--- сформируем сообщение
string comm=StringFormat("%s ==> Обновлено значений в индикаторе %s: %d",
                           TimeToString(TimeCurrent(),TIME_DATE|TIME_SECONDS),
                           short_name,
                           values_to_copy);

//--- выведем на график служебное сообщение
Comment(comm);

//--- заппомним количество значений в индикаторе Momentum
bars_calculated=calculated;
//--- вернем значение prev_calculated для следующего вызова
return(rates_total);
}

//+-----+
//| Заполняем индикаторный буфер из индикатора iMomentum |
//+-----+

bool FillArrayFromBuffer(double &values[], // индикаторный буфер значений Momentum
                        int ind_handle,    // хэндл индикатора iMomentum
                        int amount        // количество копируемых значений
                        )
{
    //--- сбросим код ошибки
    ResetLastError();
    //--- заполняем часть массива iMomentumBuffer значениями из индикаторного буфера под
    if(CopyBuffer(ind_handle,0,0,amount,values)<0)

```

```
{
    //--- если копирование не удалось, сообщим код ошибки
    PrintFormat("Не удалось скопировать данные из индикатора iMomentum, код ошибки");
    //--- завершим с нулевым результатом - это означает, что индикатор будет считат
    return(false);
}

//--- все получилось
return(true);
}

//+-----+
//| Indicator deinitialization function |
//+-----+
void OnDeinit(const int reason)
{
    //--- почистим график при удалении индикатора
    Comment("");
}
```

iMFI

函数返回资金流向指标处理器。

```
int iMFI(
    string          symbol,          // 交易品种名称
    ENUM_TIMEFRAMES period,          // 周期
    int             ma_period,        // 平均周期
    ENUM_APPLIED_VOLUME applied_volume // 计算的交易量类型
);
```

参量

symbol

[in] 证券交易品种名称，数据用来计算指标。 [NULL](#) 值代表当前交易品种。

period

[in] 周期值可以是 [ENUM_TIMEFRAMES](#) 值中的一个，0代表当前时间表。

ma_period

[in] 计算的平均周期（计算柱）。

applied_volume

[in] 使用的交易量。可以是 [ENUM_APPLIED_VOLUME](#) 中任意值。

返回值

返回特殊技术指标处理器，失败返回 [INVALID_HANDLE](#)。计算机内存从不使用的指标中释放，使用指标处理程序传递到的函数 [IndicatorRelease\(\)](#)。

:

```
//+-----+
//|                                     Demo_iMFI.mq5 |
//|                                     Copyright 2011, MetaQuotes Software Corp. |
//|                                     http://www.mql5.com |
//+-----+

#property copyright "Copyright 2011, MetaQuotes Software Corp."
#property link      "http://www.mql5.com"
#property version   "1.00"
#property description "Индикатор демонстрирует как нужно получать данные"
#property description "индикаторных буферов для технического индикатора iMFI."
#property description "Символ и таймфрейм, на котором рассчитывается индикатор,"
#property description "задаются параметрами symbol и period."
#property description "Способ создания хэндла задается параметром 'type' (тип функции"
#property description "Все остальные параметры как в стандартном Money Flow Index."

#property indicator_separate_window
#property indicator_buffers 1
#property indicator_plots   1
//--- построение iMFI
#property indicator_label1  "iMFI"
```



```

#property indicator_type1    DRAW_LINE
#property indicator_color1   clrDodgerBlue
#property indicator_style1   STYLE_SOLID
#property indicator_width1   1
//--- горизонтальные уровни в окне индикатора
#property indicator_level1   20
#property indicator_level2   80
//+-----+
//|  перечисление способов создания хэндла  |
//+-----+
enum Creation
{
    Call_iMFI,           // использовать iMFI
    Call_IndicatorCreate // использовать IndicatorCreate
};
//--- входные параметры
input Creation          type=Call_iMFI;           // тип функции
input int               ma_period=14;             // период
input ENUM_APPLIED_VOLUME applied_volume=VOLUME_TICK; // тип объема
input string            symbol=" ";               // символ
input ENUM_TIMEFRAMES   period=PERIOD_CURRENT;    // таймфрейм
//--- индикаторный буфер
double iMFIBuffer[];
//--- переменная для хранения хэндла индикатора iMFI
int handle;
//--- переменная для хранения
string name=symbol;
//--- имя индикатора на графике
string short_name;
//--- будем хранить количество значений в индикаторе Money Flow Index
int bars_calculated=0;
//+-----+
//| Custom indicator initialization function  |
//+-----+
int OnInit()
{
    //--- привязка массива к индикаторному буферу
    SetIndexBuffer(0,iMFIBuffer,INDICATOR_DATA);
    //--- определимся с символом, на котором строится индикатор
    name=symbol;
    //--- удалим пробелы слева и справа
    StringTrimRight(name);
    StringTrimLeft(name);
    //--- если после этого длина строки name нулевая
    if(StringLen(name)==0)
    {
        //--- возьмем символ с графика, на котором запущен индикатор
        name=_Symbol;
    }
}

```

```

    }
//--- создадим хэндл индикатора
if(type==Call_iMFI)
    handle=iMFI(name,period,ma_period,applied_volume);
else
{
    //--- заполним структуру значениями параметров индикатора
    MqlParam pars[2];
    //--- период
    pars[0].type=TYPE_INT;
    pars[0].integer_value=ma_period;
    //--- тип объема
    pars[1].type=TYPE_INT;
    pars[1].integer_value=applied_volume;
    handle=IndicatorCreate(name,period,IND_MFI,2,pars);
}
//--- если не удалось создать хэндл
if(handle==INVALID_HANDLE)
{
    //--- сообщим о неудаче и выведем номер ошибки
    PrintFormat("Не удалось создать хэндл индикатора iMFI для пары %s/%s, код ошибки",
                name,
                EnumToString(period),
                GetLastError());
    //--- работа индикатора завершается досрочно при возврате отрицательного значения
    return(-1);
}
//--- покажем на какой паре символ/таймфрейм рассчитан индикатор Money Flow Index
short_name=StringFormat("iMFI(%s/%s, %d, %s)",name,EnumToString(period),
                        ma_period, EnumToString(applied_volume));
IndicatorSetString(INDICATOR_SHORTNAME,short_name);
//--- нормальное выполнение инициализации индикатора
return(0);
}
//+-----+
//| Custom indicator iteration function |
//+-----+
int OnCalculate(const int rates_total,
                const int prev_calculated,
                const datetime &time[],
                const double &open[],
                const double &high[],
                const double &low[],
                const double &close[],
                const long &tick_volume[],
                const long &volume[],
                const int &spread[])
{

```

```

//--- количество копируемых значений из индикатора iMFI
int values_to_copy;
//--- узнаем количество рассчитанных значений в индикаторе
int calculated=BarsCalculated(handle);
if(calculated<=0)
{
    PrintFormat("BarsCalculated() вернул %d, код ошибки %d",calculated,GetLastError());
    return(0);
}
//--- если это первый запуск вычислений нашего индикатора или изменилось количество з
//--- или если необходимо рассчитать индикатор для двух или более баров (значит что-т
if(prev_calculated==0 || calculated!=bars_calculated || rates_total>prev_calculated)
{
    //--- если массив iMFIBuffer больше, чем значений в индикаторе iMFI на паре sym
    //--- в противном случае копировать будем меньше, чем размер индикаторных буфер
    if(calculated>rates_total) values_to_copy=rates_total;
    else
        values_to_copy=calculated;
}
else
{
    //--- значит наш индикатор рассчитывается не в первый раз и с момента последнег
    //--- для расчета добавилось не более одного бара
    values_to_copy=(rates_total-prev_calculated)+1;
}
//--- заполняем массив iMFIBuffer значениями из индикатора Money Flow Index
//--- если FillArrayFromBuffer вернула false, значит данные не готовы - завершаем раб
if(!FillArrayFromBuffer(iMFIBuffer,handle,values_to_copy)) return(0);
//--- сформируем сообщение
string comm=StringFormat("%s ==> Обновлено значений в индикаторе %s: %d",
    TimeToString(TimeCurrent(),TIME_DATE|TIME_SECONDS),
    short_name,
    values_to_copy);
//--- выведем на график служебное сообщение
Comment(comm);
//--- заппомним количество значений в индикаторе Money Flow Index
bars_calculated=calculated;
//--- вернем значение prev_calculated для следующего вызова
return(rates_total);
}
//+-----+
//| Заполняем индикаторный буфер из индикатора iMFI |
//+-----+
bool FillArrayFromBuffer(double &values[], // индикаторный буфер значений Money Flow
    int ind_handle, // хэндл индикатора iMFI
    int amount // количество копируемых значений
)
{
    //--- сбросим код ошибки

```

```

ResetLastError();
//--- заполняем часть массива iMFIBuffer значениями из индикаторного буфера под индекс
if(CopyBuffer(ind_handle,0,0,amount,values)<0)
{
    //--- если копирование не удалось, сообщим код ошибки
    PrintFormat("Не удалось скопировать данные из индикатора iMFI, код ошибки %d",G
    //--- завершим с нулевым результатом - это означает, что индикатор будет считат
    return(false);
}
//--- все получилось
return(true);
}
//+-----+
//| Indicator deinitialization function |
//+-----+
void OnDeinit(const int reason)
{
    //--- почистим график при удалении индикатора
    Comment("");
}

```

iMA

函数返回移动平均数指标处理器。只有一个缓冲区。

```
int iMA(
    string          symbol,          // 交易品种名称
    ENUM_TIMEFRAMES period,          // 周期
    int             ma_period,        // 平均周期
    int             ma_shift,         // 平移
    ENUM_MA_METHOD  ma_method,        // 平滑类型
    ENUM_APPLIED_PRICE applied_price // 价格或者处理程序类型
);
```

参量

symbol

[in] 证券交易品种名称，数据用来计算指标。 [NULL](#) 值代表当前交易品种。

period

[in] 周期值可以是 [ENUM_TIMEFRAMES](#) 值中的一个，0代表当前时间表。

ma_period

[in] 计算移动平均数的平均周期。

ma_shift

[in] 与价格图表有关的指标变化。

ma_method

[in] 平滑类型。可以是 [ENUM_MA_METHOD](#) 值中的一个。

applied_price

[in] 使用价格。可以是任意 [ENUM_APPLIED_PRICE](#) 价格常量或者另外指标处理器。

返回值

返回特殊技术指标处理器，失败返回 [INVALID_HANDLE](#)。计算机内存从不使用的指标中释放，使用指标处理程序传递到的函数 [IndicatorRelease\(\)](#)。

```
//+-----+
//|                                     Demo_iMA.mq5 |
//|                                     Copyright 2011, MetaQuotes Software Corp. |
//|                                     http://www.mql5.com |
//+-----+
#property copyright "Copyright 2011, MetaQuotes Software Corp."
#property link      "http://www.mql5.com"
#property version   "1.00"
#property description "Индикатор демонстрирует как нужно получать данные"
#property description "индикаторных буферов для технического индикатора iMA."
#property description "Символ и таймфрейм, на котором рассчитывается индикатор,"
#property description "задаются параметрами symbol и period."
```

```

#property description "Способ создания хэндла задается параметром 'type' (тип функции"
#property description "Все остальные параметры как в стандартном Moving Average."

#property indicator_chart_window
#property indicator_buffers 1
#property indicator_plots 1
//--- построение iMA
#property indicator_label1 "iMA"
#property indicator_type1 DRAW_LINE
#property indicator_color1 clrRed
#property indicator_style1 STYLE_SOLID
#property indicator_width1 1
//+-----+
//| перечисление способов создания хэндла |
//+-----+
enum Creation
{
    Call_iMA,          // использовать iMA
    Call_IndicatorCreate // использовать IndicatorCreate
};
//--- входные параметры
input Creation      type=Call_iMA;          // тип функции
input int           ma_period=10;           // период средней
input int           ma_shift=0;             // смещение
input ENUM_MA_METHOD ma_method=MODE_SMA;    // тип сглаживания
input ENUM_APPLIED_PRICE applied_price=PRICE_CLOSE; // тип цены
input string        symbol=" ";            // символ
input ENUM_TIMEFRAMES period=PERIOD_CURRENT; // таймфрейм
//--- индикаторный буфер
double iMABuffer[];
//--- переменная для хранения хэндла индикатора iMA
int handle;
//--- переменная для хранения
string name=symbol;
//--- имя индикатора на графике
string short_name;
//--- будем хранить количество значений в индикаторе Moving Average
int bars_calculated=0;
//+-----+
//| Custom indicator initialization function |
//+-----+
int OnInit()
{
    //--- привязка массива к индикаторному буферу
    SetIndexBuffer(0,iMABuffer,INDICATOR_DATA);
    //--- зададим смещение
    PlotIndexSetInteger(0,PLOT_SHIFT,ma_shift);
    //--- определимся с символом, на котором строится индикатор

```

```

name=symbol;
//--- удалим пробелы слева и справа
StringTrimRight(name);
StringTrimLeft(name);
//--- если после этого длина строки name нулевая
if(StringLen(name)==0)
{
    //--- возьмем символ с графика, на котором запущен индикатор
    name=_Symbol;
}
//--- создадим хэндл индикатора
if(type==Call_iMA)
    handle=iMA(name,period,ma_period,ma_shift,ma_method,applied_price);
else
{
    //--- заполним структуру значениями параметров индикатора
    MqlParam pars[4];
    //--- период
    pars[0].type=TYPE_INT;
    pars[0].integer_value=ma_period;
    //--- смещение
    pars[1].type=TYPE_INT;
    pars[1].integer_value=ma_shift;
    //--- тип сглаживания
    pars[2].type=TYPE_INT;
    pars[2].integer_value=ma_method;
    //--- тип цены
    pars[3].type=TYPE_INT;
    pars[3].integer_value=applied_price;
    handle=IndicatorCreate(name,period,IND_MA,4,pars);
}
//--- если не удалось создать хэндл
if(handle==INVALID_HANDLE)
{
    //--- сообщим о неудаче и выведем номер ошибки
    PrintFormat("Не удалось создать хэндл индикатора iMA для пары %s/%s, код ошибки",
        name,
        EnumToString(period),
        GetLastError());
    //--- работа индикатора завершается досрочно при возврате отрицательного значения
    return(-1);
}
//--- покажем на какой паре символ/таймфрейм рассчитан индикатор Moving Average
short_name=StringFormat("iMA(%s/%s, %d, %d, %s, %s)",name,EnumToString(period),
    ma_period, ma_shift,EnumToString(ma_method),EnumToString(applied_price));
IndicatorSetString(INDICATOR_SHORTNAME,short_name);
//--- нормальное выполнение инициализации индикатора
return(0);

```

```

    }
//+-----+
//| Custom indicator iteration function |
//+-----+
int OnCalculate(const int rates_total,
               const int prev_calculated,
               const datetime &time[],
               const double &open[],
               const double &high[],
               const double &low[],
               const double &close[],
               const long &tick_volume[],
               const long &volume[],
               const int &spread[])
{
//--- количество копируемых значений из индикатора iMA
    int values_to_copy;
//--- узнаем количество рассчитанных значений в индикаторе
    int calculated=BarsCalculated(handle);
    if(calculated<=0)
    {
        PrintFormat("BarsCalculated() вернул %d, код ошибки %d",calculated,GetLastError());
        return(0);
    }
//--- если это первый запуск вычислений нашего индикатора или изменилось количество э
//--- или если необходимо рассчитать индикатор для двух или более баров (значит что-т
    if(prev_calculated==0 || calculated!=bars_calculated || rates_total>prev_calculate
    {
        //--- если массив iMABuffer больше, чем значений в индикаторе iMA на паре symbo
        //--- в противном случае копировать будем меньше, чем размер индикаторных буфер
        if(calculated>rates_total) values_to_copy=rates_total;
        else
            values_to_copy=calculated;
    }
    else
    {
        //--- значит наш индикатор рассчитывается не в первый раз и с момента последнег
        //--- для расчета добавилось не более одного бара
        values_to_copy=(rates_total-prev_calculated)+1;
    }
//--- заполняем массив iMABuffer значениями из индикатора Moving Average
//--- если FillArrayFromBuffer вернула false, значит данные не готовы - завершаем раб
    if(!FillArrayFromBuffer(iMABuffer,ma_shift,handle,values_to_copy)) return(0);
//--- сформируем сообщение
    string comm=StringFormat("%s ==> Обновлено значений в индикаторе %s: %d",
                             TimeToString(TimeCurrent(),TIME_DATE|TIME_SECONDS),
                             short_name,
                             values_to_copy);
//--- выведем на график служебное сообщение

```



```

    Comment(comm);
//--- заппомним количество значений в индикаторе Moving Average
    bars_calculated=calculated;
//--- вернем значение prev_calculated для следующего вызова
    return(rates_total);
}
//+-----+
//| Заполняем индикаторный буфер из индикатора iMA |
//+-----+
bool FillArrayFromBuffer(double &values[], // индикаторный буфер значений Moving Av
                        int shift,         // смещение
                        int ind_handle,     // хэндл индикатора iMA
                        int amount         // количество копируемых значений
                        )
{
//--- сбросим код ошибки
    ResetLastError();
//--- заполняем часть массива iMABuffer значениями из индикаторного буфера под индекс
    if(CopyBuffer(ind_handle,0,-shift,amount,values)<0)
    {
        //--- если копирование не удалось, сообщим код ошибки
        PrintFormat("Не удалось скопировать данные из индикатора iMA, код ошибки %d",Ge
        //--- завершим с нулевым результатом - это означает, что индикатор будет считат
        return(false);
    }
//--- все получилось
    return(true);
}
//+-----+
//| Indicator deinitialization function |
//+-----+
void OnDeinit(const int reason)
{
//--- почистим график при удалении индикатора
    Comment("");
}

```

iOsMA

函数返回移动平均震荡指标处理器。OsMA振荡器显示MACD与其讯息线之间的区别。只有一个缓冲区。

```
int iOsMA(
    string          symbol,           // 交易品种名称
    ENUM_TIMEFRAMES period,          // 周期
    int             fast_ema_period,  // 快速移动平均数周期
    int             slow_ema_period,  // 慢速移动平均数周期
    int             signal_period,    // 不同点的平均周期
    ENUM_APPLIED_PRICE applied_price  // 价格或者处理器的类型
);
```

参量

symbol

[in] 证券交易品种名称，数据用来计算指标。 [NULL](#) 值代表当前交易品种。

period

[in] 周期值可以是 [ENUM_TIMEFRAMES](#) 值中的一个，0代表当前时间表。

fast_ema_period

[in] 快速移动平均数计算周期。

slow_ema_period

[in] 慢速移动平均数计算周期。

signal_period

[in] 讯息线计算平均周期。

applied_price

[in] 使用价格。可以是任意[ENUM_APPLIED_PRICE](#)价格常量或者另外指标处理器。

返回值

返回特殊技术指标处理器，失败返回 [INVALID_HANDLE](#)。计算机内存从不使用的指标中释放，使用指标处理程序传递到的函数 [IndicatorRelease\(\)](#)。

注释

在一些系统中该振荡器也被认为是MACD柱状图。

:

```
//+-----+
//|                                     Demo_iOsMA.mq5 |
//|                                     Copyright 2011, MetaQuotes Software Corp. |
//|                                     http://www.mql5.com |
//+-----+
#property copyright "Copyright 2011, MetaQuotes Software Corp."
#property link      "http://www.mql5.com"
#property version   "1.00"
#property description "Индикатор демонстрирует как нужно получать данные"
```

```

#property description "индикаторных буферов для технического индикатора iOsMA."
#property description "Символ и таймфрейм, на котором рассчитывается индикатор,"
#property description "задаются параметрами symbol и period."
#property description "Способ создания хэндла задается параметром 'type' (тип функции"
#property description "Все остальные параметры как в стандартном Moving Average of Os

#property indicator_separate_window
#property indicator_buffers 1
#property indicator_plots 1
//--- построение iOsMA
#property indicator_label1 "iOsMA"
#property indicator_type1 DRAW_HISTOGRAM
#property indicator_color1 clrSilver
#property indicator_style1 STYLE_SOLID
#property indicator_width1 1
//+-----+
//| перечисление способов создания хэндла |
//+-----+
enum Creation
{
    Call_iOsMA,          // использовать iOsMA
    Call_IndicatorCreate // использовать IndicatorCreate
};
//--- входные параметры
input Creation      type=Call_iOsMA;          // тип функции
input int           fast_ema_period=12;        // период быстрой средней
input int           slow_ema_period=26;        // период медленной средней
input int           signal_period=9;           // период усреднения разности
input ENUM_APPLIED_PRICE applied_price=PRICE_CLOSE; // тип цены
input string        symbol=" ";               // символ
input ENUM_TIMEFRAMES period=PERIOD_CURRENT;  // таймфрейм
//--- индикаторный буфер
double           iOsMABuffer[];
//--- переменная для хранения хэндла индикатора iAMA
int             handle;
//--- переменная для хранения
string name=symbol;
//--- имя индикатора на графике
string short_name;
//--- будем хранить количество значений в индикаторе Moving Average of Oscillator
int             bars_calculated=0;
//+-----+
//| Custom indicator initialization function |
//+-----+
int OnInit()
{
    //--- привязка массива к индикаторному буферу
    SetIndexBuffer(0,iOsMABuffer,INDICATOR_DATA);

```

```

//--- определимся с символом, на котором строится индикатор
name=symbol;
//--- удалим пробелы слева и справа
StringTrimRight(name);
StringTrimLeft(name);
//--- если после этого длина строки name нулевая
if(StringLen(name)==0)
{
    //--- возьмем символ с графика, на котором запущен индикатор
    name=_Symbol;
}
//--- создадим хэндл индикатора
if(type==Call_iOsMA)
    handle=iOsMA(name,period,fast_ema_period,slow_ema_period,signal_period,applied_
else
{
    //--- заполним структуру значениями параметров индикатора
    MqlParam pars[4];
    //--- быстрый период
    pars[0].type=TYPE_INT;
    pars[0].integer_value=fast_ema_period;
    //--- медленный период
    pars[1].type=TYPE_INT;
    pars[1].integer_value=slow_ema_period;
    //--- период усреднения разницы между быстрой и медленной средними
    pars[2].type=TYPE_INT;
    pars[2].integer_value=signal_period;
    //--- тип цены
    pars[3].type=TYPE_INT;
    pars[3].integer_value=applied_price;
    handle=IndicatorCreate(name,period,IND_OSMA,4,pars);
}
//--- если не удалось создать хэндл
if(handle==INVALID_HANDLE)
{
    //--- сообщим о неудаче и выведем номер ошибки
    PrintFormat("Не удалось создать хэндл индикатора iOsMA для пары %s/%s, код ошибк
        name,
        EnumToString(period),
        GetLastError());
    //--- работа индикатора завершается досрочно при возврате отрицательного значен
    return(-1);
}
//--- покажем на какой паре символ/таймфрейм рассчитан индикатор Moving Average of Os
short_name=StringFormat("iOsMA(%s/%s,%d,%d,%d,%s)",name,EnumToString(period),
        fast_ema_period,slow_ema_period,signal_period,EnumToString
    IndicatorSetString(INDICATOR_SHORTNAME,short_name);
//--- нормальное выполнение инициализации индикатора

```

```

    return(0);
}
//+-----+
//| Custom indicator iteration function |
//+-----+
int OnCalculate(const int rates_total,
               const int prev_calculated,
               const datetime &time[],
               const double &open[],
               const double &high[],
               const double &low[],
               const double &close[],
               const long &tick_volume[],
               const long &volume[],
               const int &spread[])
{
    //--- количество копируемых значений из индикатора iOsMA
    int values_to_copy;
    //--- узнаем количество рассчитанных значений в индикаторе
    int calculated=BarsCalculated(handle);
    if(calculated<=0)
    {
        PrintFormat("BarsCalculated() вернул %d, код ошибки %d",calculated,GetLastError());
        return(0);
    }
    //--- если это первый запуск вычислений нашего индикатора или изменилось количество э
    //--- или если необходимо рассчитать индикатор для двух или более баров (значит что-т
    if(prev_calculated==0 || calculated!=bars_calculated || rates_total>prev_calculate
    {
        //--- если массив iOsMABuffer больше, чем значений в индикаторе iOsMA на паре s
        //--- в противном случае копировать будем меньше, чем размер индикаторных буфер
        if(calculated>rates_total) values_to_copy=rates_total;
        else
            values_to_copy=calculated;
    }
    else
    {
        //--- значит наш индикатор рассчитывается не в первый раз и с момента последнег
        //--- для расчета добавилось не более одного бара
        values_to_copy=(rates_total-prev_calculated)+1;
    }
    //--- заполняем массивы значениями из индикатора iOsMA
    //--- если FillArrayFromBuffer вернула false, значит данные не готовы - завершаем раб
    if(!FillArrayFromBuffer(iOsMABuffer,handle,values_to_copy)) return(0);
    //--- сформируем сообщение
    string comm=StringFormat("%s ==> Обновлено значений в индикаторе %s: %d",
                             TimeToString(TimeCurrent(),TIME_DATE|TIME_SECONDS),
                             short_name,
                             values_to_copy);

```

```

//--- выведем на график служебное сообщение
    Comment(comm);
//--- заппомним количество значений в индикаторе Moving Average of Oscillator
    bars_calculated=calculated;
//--- вернем значение prev_calculated для следующего вызова
    return(rates_total);
}
//+-----+
//| Заполняем индикаторный буфер из индикатора iOsMA |
//+-----+
bool FillArrayFromBuffer(double &ama_buffer[], // индикаторный буфер значений OsMA
                        int ind_handle,        // хэндл индикатора iOsMA
                        int amount            // количество копируемых значений
                        )
{
//--- сбросим код ошибки
    ResetLastError();
//--- заполняем часть массива iOsMABuffer значениями из индикаторного буфера под инде
    if(CopyBuffer(ind_handle,0,0,amount,ama_buffer)<0)
    {
        //--- если копирование не удалось, сообщим код ошибки
        PrintFormat("Не удалось скопировать данные из индикатора iOsMA, код ошибки %d",
        //--- завершим с нулевым результатом - это означает, что индикатор будет считат
        return(false);
    }
//--- все получилось
    return(true);
}
//+-----+
//| Indicator deinitialization function |
//+-----+
void OnDeinit(const int reason)
{
//--- почистим график при удалении индикатора
    Comment("");
}

```

iMACD

函数返回移动平均数聚/散指标处理器。在OsMA被称为MACD柱状图的系统中，该指标显示为两条线。在客户端移动平均数聚/散像柱状图。

```
int iMACD(
    string          symbol,          // 交易品种名称
    ENUM_TIMEFRAMES period,          // 周期
    int             fast_ema_period, // 快速移动平均数周期
    int             slow_ema_period, // 慢速移动平均数周期
    int             signal_period,   // 不同点的平均周期
    ENUM_APPLIED_PRICE applied_price // 价格或者处理器的类型
);
```

参量

symbol

[in] 证券交易品种名称，数据用来计算指标。 [NULL](#) 值代表当前交易品种。

period

[in] 周期值可以是 [ENUM_TIMEFRAMES](#) 值中的一个，0代表当前时间表。

fast_ema_period

[in] 快速移动平均数计算周期。

slow_ema_period

[in] 慢速移动平均数计算周期。

signal_period

[in] 讯号线计算周期。

applied_price

[in] 使用价格。可以是任意 [ENUM_APPLIED_PRICE](#) 价格常量或者另外指标处理器。

返回值

返回特殊技术指标处理器，失败返回 [INVALID_HANDLE](#)。计算机内存从不使用的指标中释放，使用指标处理程序传递到的函数 [IndicatorRelease\(\)](#)。

注释

缓冲区代码如下：0 - MAIN_ LINE，1 - SIGNAL_ LINE。

:

```
//+-----+
//|                                     Demo_iMACD.mq5 |
//|                                     Copyright 2011, MetaQuotes Software Corp. |
//|                                     http://www.mql5.com |
//+-----+
#property copyright "Copyright 2011, MetaQuotes Software Corp."
#property link      "http://www.mql5.com"
#property version   "1.00"
```

```

#property description "Индикатор демонстрирует как нужно получать данные"
#property description "индикаторных буферов для технического индикатора iMACD."
#property description "Символ и таймфрейм, на котором рассчитывается индикатор,"
#property description "задаются параметрами symbol и period."
#property description "Способ создания хэндла задается параметром 'type' (тип функции"
#property description "Все остальные параметры как в стандартном MACD."

#property indicator_separate_window
#property indicator_buffers 2
#property indicator_plots 2
//--- построение MACD
#property indicator_label1 "MACD"
#property indicator_type1 DRAW_HISTOGRAM
#property indicator_color1 clrSilver
#property indicator_style1 STYLE_SOLID
#property indicator_width1 1
//--- построение Signal
#property indicator_label2 "Signal"
#property indicator_type2 DRAW_LINE
#property indicator_color2 clrRed
#property indicator_style2 STYLE_DOT
#property indicator_width2 1
//+-----+
//| перечисление способов создания хэндла |
//+-----+
enum Creation
{
    Call_iMACD,          // использовать iMACD
    Call_IndicatorCreate // использовать IndicatorCreate
};
//--- входные параметры
input Creation      type=Call_iMACD;          // тип функции
input int           fast_ema_period=12;        // период быстрой средней
input int           slow_ema_period=26;        // период медленной средней
input int           signal_period=9;          // период усреднения разности
input ENUM_APPLIED_PRICE applied_price=PRICE_CLOSE; // тип цены
input string        symbol=" ";              // символ
input ENUM_TIMEFRAMES period=PERIOD_CURRENT; // таймфрейм
//--- индикаторные буферы
double      MACDBuffer[];
double      SignalBuffer[];
//--- переменная для хранения хэндла индикатора iMACD
int         handle;
//--- переменная для хранения
string name=symbol;
//--- имя индикатора на графике
string short_name;
//--- будем хранить количество значений в индикаторе Moving Averages Convergence/Dive

```



```

int    bars_calculated=0;
//+-----+
//| Custom indicator initialization function |
//+-----+
int OnInit()
{
//--- привязка массивов к индикаторным буферам
    SetIndexBuffer(0,MACDBuffer,INDICATOR_DATA);
    SetIndexBuffer(1,SignalBuffer,INDICATOR_DATA);
//--- определимся с символом, на котором строится индикатор
    name=symbol;
//--- удалим пробелы слева и справа
    StringTrimRight(name);
    StringTrimLeft(name);
//--- если после этого длина строки name нулевая
    if(StringLen(name)==0)
    {
        //--- возьмем символ с графика, на котором запущен индикатор
        name=_Symbol;
    }
//--- создадим хэндл индикатора
    if(type==Call_iMACD)
        handle=iMACD(name,period,fast_ema_period,slow_ema_period,signal_period,applied_
    else
    {
        //--- заполним структуру значениями параметров индикатора
        MqlParam pars[4];
        //--- быстрый период
        pars[0].type=TYPE_INT;
        pars[0].integer_value=fast_ema_period;
        //--- медленный период
        pars[1].type=TYPE_INT;
        pars[1].integer_value=slow_ema_period;
        //--- период усреднения разницы между быстрой и медленной средними
        pars[2].type=TYPE_INT;
        pars[2].integer_value=signal_period;
        //--- тип цены
        pars[3].type=TYPE_INT;
        pars[3].integer_value=applied_price;
        handle=IndicatorCreate(name,period,IND_MACD,4,pars);
    }
//--- если не удалось создать хэндл
    if(handle==INVALID_HANDLE)
    {
        //--- сообщим о неудаче и выведем номер ошибки
        PrintFormat("Не удалось создать хэндл индикатора iMACD для пары %s/%s, код ошибки
            name,
            EnumToString(period),

```

```

        GetLastError();

        //--- работа индикатора завершается досрочно при возврате отрицательного значения
        return(-1);
    }

    //--- покажем на какой паре символ/таймфрейм рассчитан индикатор Moving Averages Conv
    short_name=StringFormat("iMACD(%s/%s,%d,%d,%d,%s)",name,EnumToString(period),
        fast_ema_period,slow_ema_period,signal_period,EnumToString
        IndicatorSetString(INDICATOR_SHORTNAME,short_name);
    //--- нормальное выполнение инициализации индикатора
    return(0);
}

//+-----+
//| Custom indicator iteration function |
//+-----+

int OnCalculate(const int rates_total,
                const int prev_calculated,
                const datetime &time[],
                const double &open[],
                const double &high[],
                const double &low[],
                const double &close[],
                const long &tick_volume[],
                const long &volume[],
                const int &spread[])
{
    //--- количество копируемых значений из индикатора iMACD
    int values_to_copy;
    //--- узнаем количество рассчитанных значений в индикаторе
    int calculated=BarsCalculated(handle);
    if(calculated<=0)
    {
        PrintFormat("BarsCalculated() вернул %d, код ошибки %d",calculated,GetLastError()
        return(0);
    }

    //--- если это первый запуск вычислений нашего индикатора или изменилось количество з
    //--- или если необходимо рассчитать индикатор для двух или более баров (значит что-т
    if(prev_calculated==0 || calculated!=bars_calculated || rates_total>prev_calculate
    {
        //--- если массив MACDBuffer больше, чем значений в индикаторе iMACD на паре sy
        //--- в противном случае копировать будем меньше, чем размер индикаторных буфер
        if(calculated>rates_total) values_to_copy=rates_total;
        else
            values_to_copy=calculated;
    }
    else
    {
        //--- значит наш индикатор рассчитывается не в первый раз и с момента последнег
        //--- для расчета добавилось не более одного бара
        values_to_copy=(rates_total-prev_calculated)+1;
    }
}

```

```

    }
    //--- заполняем массивы значениями из индикатора iMACD
    //--- если FillArraysFromBuffers вернула false, значит данные не готовы - завершаем р
    if(!FillArraysFromBuffers(MACDBuffer,SignalBuffer,handle,values_to_copy)) return(0)
    //--- сформируем сообщение
    string comm=StringFormat("%s ==> Обновлено значений в индикаторе %s: %d",
                              TimeToString(TimeCurrent(),TIME_DATE|TIME_SECONDS),
                              short_name,
                              values_to_copy);
    //--- выведем на график служебное сообщение
    Comment(comm);
    //--- запоним количество значений в индикаторе Moving Averages Convergence/Divergen
    bars_calculated=calculated;
    //--- вернем значение prev_calculated для следующего вызова
    return(rates_total);
}
//+-----+
//| Заполняем индикаторные буферы из индикатора iMACD |
//+-----+
bool FillArraysFromBuffers(double &macd_buffer[], // индикаторный буфер значений М
                           double &signal_buffer[], // индикаторный буфер сигнальной
                           int ind_handle, // хэндл индикатора iMACD
                           int amount // количество копируемых значений
                           )
{
    //--- сбросим код ошибки
    ResetLastError();
    //--- заполняем часть массива iMACDBuffer значениями из индикаторного буфера под инде
    if(CopyBuffer(ind_handle,0,0,amount,macd_buffer)<0)
    {
        //--- если копирование не удалось, сообщим код ошибки
        PrintFormat("Не удалось скопировать данные из индикатора iMACD, код ошибки %d",
        //--- завершим с нулевым результатом - это означает, что индикатор будет считат
        return(false);
    }

    //--- заполняем часть массива SignalBuffer значениями из индикаторного буфера под инд
    if(CopyBuffer(ind_handle,1,0,amount,signal_buffer)<0)
    {
        //--- если копирование не удалось, сообщим код ошибки
        PrintFormat("Не удалось скопировать данные из индикатора iMACD, код ошибки %d",
        //--- завершим с нулевым результатом - это означает, что индикатор будет считат
        return(false);
    }
    //--- все получилось
    return(true);
}
//+-----+

```

```
///| Indicator deinitialization function |  
//+-----+  
void OnDeinit(const int reason)  
{  
//--- почистим график при удалении индикатора  
    Comment("");  
}
```

iOBV

函数返回平衡交易量指标处理器。只有一个缓冲区。

```
int iOBV(
    string          symbol,          // 交易品种名称
    ENUM_TIMEFRAMES period,          // 周期
    ENUM_APPLIED_VOLUME applied_volume // 计算的交易量类型
);
```

参量

symbol

[in] 证券交易品种名称，数据用来计算指标。 [NULL](#) 值代表当前交易品种。

period

[in] 周期值可以是 [ENUM_TIMEFRAMES](#) 值中的一个，0代表当前时间表。

applied_volume

[in] 使用的交易量，可以是 [ENUM_APPLIED_VOLUME](#) 中任意值。

返回值

返回特殊技术指标处理器，失败返回 [INVALID_HANDLE](#)。计算机内存从不使用的指标中释放，使用指标处理程序传递到的函数 [IndicatorRelease\(\)](#)。

:

```
//+-----+
//|                                     Demo_iOBV.mq5 |
//|                                     Copyright 2011, MetaQuotes Software Corp. |
//|                                     http://www.mql5.com |
//+-----+

#property copyright "Copyright 2011, MetaQuotes Software Corp."
#property link      "http://www.mql5.com"
#property version   "1.00"
#property description "Индикатор демонстрирует как нужно получать данные"
#property description "индикаторных буферов для технического индикатора iOBV."
#property description "Символ и таймфрейм, на котором рассчитывается индикатор,"
#property description "задаются параметрами symbol и period."
#property description "Способ создания хэндла задается параметром 'type' (тип функции"

#property indicator_separate_window
#property indicator_buffers 1
#property indicator_plots 1
//--- iOBV
#property indicator_label1 "iOBV"
#property indicator_type1  DRAW_LINE
#property indicator_color1 clrLightSeaGreen
#property indicator_style1 STYLE_SOLID
#property indicator_width1 1
```

```

//+-----+
//|  перечисление способов создания хэндла                                     |
//+-----+
enum Creation
{
    Call_iOBV ,           // использовать iOBV
    Call_IndicatorCreate  // использовать IndicatorCreate
};
//--- входные параметры
input Creation          type=Call_iOBV;           // тип функции
input ENUM_APPLIED_VOLUME applied_volume=VOLUME_TICK; // тип объема
input string            symbol=" ";               // символ
input ENUM_TIMEFRAMES   period=PERIOD_CURRENT;    // таймфрейм
//--- индикаторные буферы
double      iOBVBuffer[];
//--- переменная для хранения хэндла индикатора iOBV
int         handle;
//--- переменная для хранения
string name=symbol;
//--- имя индикатора на графике
string short_name;
//--- будем хранить количество значений в индикаторе On Balance Volume
int      bars_calculated=0;
//+-----+
//| Custom indicator initialization function                                     |
//+-----+
int OnInit()
{
    //--- привязка массива к индикаторному буферу
    SetIndexBuffer(0,iOBVBuffer,INDICATOR_DATA);
    //--- определимся с символом, на котором строится индикатор
    name=symbol;
    //--- удалим пробелы слева и справа
    StringTrimRight(name);
    StringTrimLeft(name);
    //--- если после этого длина строки name нулевая
    if(StringLen(name)==0)
    {
        //--- возьмем символ с графика, на котором запущен индикатор
        name=_Symbol;
    }
    //--- создадим хэндл индикатора
    if(type==Call_iOBV)
        handle=iOBV(name,period,applied_volume);
    else
    {
        //--- заполним структуру значениями параметров индикатора
        MqlParam pars[1];
    }
}

```

```

    //--- тип объема
    pars[0].type=TYPE_INT;
    pars[0].integer_value=applied_volume;
    handle=IndicatorCreate(name,period,IND_OBV,1,pars);
}
//--- если не удалось создать хэндл
if(handle==INVALID_HANDLE)
{
    //--- сообщим о неудаче и выведем номер ошибки
    PrintFormat("Не удалось создать хэндл индикатора iOBV для пары %s/%s, код ошибки",
        name,
        EnumToString(period),
        GetLastError());
    //--- работа индикатора завершается досрочно при возврате отрицательного значения
    return(-1);
}
//--- покажем на какой паре символ/таймфрейм рассчитан индикатор On Balance Volume
short_name=StringFormat("iOBV(%s/%s, %s)",name,EnumToString(period),
    EnumToString(applied_volume));
IndicatorSetString(INDICATOR_SHORTNAME,short_name);
//--- нормальное выполнение инициализации индикатора
return(0);
}
//+-----+
//| Custom indicator iteration function |
//+-----+
int OnCalculate(const int rates_total,
    const int prev_calculated,
    const datetime &time[],
    const double &open[],
    const double &high[],
    const double &low[],
    const double &close[],
    const long &tick_volume[],
    const long &volume[],
    const int &spread[])
{
    //--- количество копируемых значений из индикатора iOBV
    int values_to_copy;
    //--- узнаем количество рассчитанных значений в индикаторе
    int calculated=BarsCalculated(handle);
    if(calculated<=0)
    {
        PrintFormat("BarsCalculated() вернул %d, код ошибки %d",calculated,GetLastError());
        return(0);
    }
    //--- если это первый запуск вычислений нашего индикатора или изменилось количество э
    //--- или если необходимо рассчитать индикатор для двух или более баров (значит что-т

```

```

if(prev_calculated==0 || calculated!=bars_calculated || rates_total>prev_calculated)
{
    //--- если массив iOBVBuffer больше, чем значений в индикаторе iOBV на паре sym
    //--- в противном случае копировать будем меньше, чем размер индикаторных буфер
    if(calculated>rates_total) values_to_copy=rates_total;
    else
        values_to_copy=calculated;
}
else
{
    //--- значит наш индикатор рассчитывается не в первый раз и с момента последнего
    //--- для расчета добавилось не более одного бара
    values_to_copy=(rates_total-prev_calculated)+1;
}
//--- заполняем массивы значениями из индикатора iOBV
//--- если FillArrayFromBuffer вернула false, значит данные не готовы - завершаем работу
if(!FillArrayFromBuffer(iOBVBuffer,handle,values_to_copy)) return(0);
//--- сформируем сообщение
string comm=StringFormat("%s ==> Обновлено значений в индикаторе %s: %d",
                          TimeToString(TimeCurrent(),TIME_DATE|TIME_SECONDS),
                          short_name,
                          values_to_copy);
//--- выведем на график служебное сообщение
Comment(comm);
//--- заппомним количество значений в индикаторе On Balance Volume
bars_calculated=calculated;
//--- вернем значение prev_calculated для следующего вызова
return(rates_total);
}
//+-----+
//| Заполняем индикаторный буфер из индикатора iOBV |
//+-----+
bool FillArrayFromBuffer(double &obv_buffer[], // индикаторный буфер значений OBV
                        int ind_handle,       // хэндл индикатора iOBV
                        int amount            // количество копируемых значений
                        )
{
    //--- сбросим код ошибки
    ResetLastError();
    //--- заполняем часть массива iOBVBuffer значениями из индикаторного буфера под индексом
    if(CopyBuffer(ind_handle,0,0,amount,obv_buffer)<0)
    {
        //--- если копирование не удалось, сообщим код ошибки
        PrintFormat("Не удалось скопировать данные из индикатора iOBV, код ошибки %d",GetLastError());
        //--- завершим с нулевым результатом - это означает, что индикатор будет считаться неактивным
        return(false);
    }
    //--- все получилось
    return(true);
}

```



```
    }  
    //+-----+  
    //| Indicator deinitialization function |  
    //+-----+  
    void OnDeinit(const int reason)  
    {  
        //--- почистим график при удалении индикатора  
        Comment("");  
    }
```

iSAR

函数返回抛物转向系统指标处理器。只有一个缓冲区。

```
int iSAR(
    string          symbol,      // 交易品种名称
    ENUM_TIMEFRAMES period,     // 周期
    double          step,       // 逐步增加
    double          maximum     // 最大止损水平
);
```

参量

symbol

[in] 证券交易品种名称，数据用来计算指标。 [NULL](#) 值代表当前交易品种。

period

[in] 周期值可以是 [ENUM_TIMEFRAMES](#) 值中的一个，0代表当前时间表。

step

[in] 停止水平增量，通常是0.02。

maximum

[in] 最大停止水平，通常是0.2。

返回值

返回特殊技术指标处理器，失败返回 [INVALID_HANDLE](#)。计算机内存从不使用的指标中释放，使用指标处理程序传递到的函数 [IndicatorRelease\(\)](#)。

:

```
//+-----+
//|                                     Demo_iSAR.mq5 |
//|                                     Copyright 2011, MetaQuotes Software Corp. |
//|                                     http://www.mql5.com |
//+-----+

#property copyright "Copyright 2011, MetaQuotes Software Corp."
#property link      "http://www.mql5.com"
#property version   "1.00"
#property description "Индикатор демонстрирует как нужно получать данные"
#property description "индикаторных буферов для технического индикатора iSAR."
#property description "Символ и таймфрейм, на котором рассчитывается индикатор,"
#property description "задаются параметрами symbol и period."
#property description "Способ создания хэндла задается параметром 'type' (тип функции"
#property description "Все остальные параметры как в стандартном Parabolic Stop and R

#property indicator_chart_window
#property indicator_buffers 1
#property indicator_plots   1
//--- построение iSAR
#property indicator_label1  "iSAR"
```

```

#property indicator_type1    DRAW_ARROW
#property indicator_color1   clrBlue
#property indicator_style1   STYLE_SOLID
#property indicator_width1   1

//+-----+
//|  перечисление способов создания хэнгла  |
//+-----+

enum Creation
{
    Call_iSAR,           // использовать iSAR
    Call_IndicatorCreate // использовать IndicatorCreate
};

//--- входные параметры
input Creation          type=Call_iSAR;           // тип функции
input double            step=0.02;                // шаг - фактор ускорения пр
input double            maximum=0.2;              // максимальное значение шаг
input string            symbol=" ";               // символ
input ENUM_TIMEFRAMES   period=PERIOD_CURRENT;    // таймфрейм

//--- индикаторные буферы
double                iSARBuffer[];

//--- переменная для хранения хэнгла индикатора iSAR
int    handle;

//--- переменная для хранения
string name=symbol;

//--- имя индикатора на графике
string short_name;

//--- будем хранить количество значений в индикаторе Parabolic SAR
int    bars_calculated=0;

//+-----+
//| Custom indicator initialization function  |
//+-----+

int OnInit()
{
    //--- привязка массива к индикаторному буферу
    SetIndexBuffer(0,iSARBuffer,INDICATOR_DATA);
    //--- установим свойству PLOT_ARROW код символа из набора Wingdings для отображения н
    PlotIndexSetInteger(0,PLOT_ARROW,159);
    //--- определимся с символом, на котором строится индикатор
    name=symbol;
    //--- удалим пробелы слева и справа
    StringTrimRight(name);
    StringTrimLeft(name);
    //--- если после этого длина строки name нулевая
    if(StringLen(name)==0)
    {
        //--- возьмем символ с графика, на котором запущен индикатор
        name=_Symbol;
    }
}

```

```

//--- создадим хэндл индикатора
if(type==Call_iSAR)
    handle=iSAR(name,period,step,maximum);
else
{
    //--- заполним структуру значениями параметров индикатора
    MqlParam pars[2];
    //--- значение шага
    pars[0].type=TYPE_DOUBLE;
    pars[0].double_value=step;
    //--- предельное значение шага, которое может использоваться при расчетах
    pars[1].type=TYPE_DOUBLE;
    pars[1].double_value=maximum;
    handle=IndicatorCreate(name,period,IND_SAR,2,pars);
}

//--- если не удалось создать хэндл
if(handle==INVALID_HANDLE)
{
    //--- сообщим о неудаче и выведем номер ошибки
    PrintFormat("Не удалось создать хэндл индикатора iSAR для пары %s/%s, код ошибки",
        name,
        EnumToString(period),
        GetLastError());

    //--- работа индикатора завершается досрочно при возврате отрицательного значения
    return(-1);
}

//--- покажем на какой паре символ/таймфрейм рассчитан индикатор Parabolic SAR
short_name=StringFormat("iSAR(%s/%s, %G, %G)",name,EnumToString(period),
    step,maximum);
IndicatorSetString(INDICATOR_SHORTNAME,short_name);
//--- нормальное выполнение инициализации индикатора
return(0);
}

//+-----+
//| Custom indicator iteration function |
//+-----+

int OnCalculate(const int rates_total,
    const int prev_calculated,
    const datetime &time[],
    const double &open[],
    const double &high[],
    const double &low[],
    const double &close[],
    const long &tick_volume[],
    const long &volume[],
    const int &spread[])
{
    //--- количество копируемых значений из индикатора iSAR

```

```

    int values_to_copy;
    //--- узнаем количество рассчитанных значений в индикаторе
    int calculated=BarsCalculated(handle);
    if(calculated<=0)
    {
        PrintFormat("BarsCalculated() вернул %d, код ошибки %d",calculated,GetLastError());
        return(0);
    }
    //--- если это первый запуск вычислений нашего индикатора или изменилось количество з
    //--- или если необходимо рассчитать индикатор для двух или более баров (значит что-т
    if(prev_calculated==0 || calculated!=bars_calculated || rates_total>prev_calculate
    {
        //--- если массив iSARBuffer больше, чем значений в индикаторе iSAR на паре sym
        //--- в противном случае копировать будем меньше, чем размер индикаторных буфер
        if(calculated>rates_total) values_to_copy=rates_total;
        else
            values_to_copy=calculated;
    }
    else
    {
        //--- значит наш индикатор рассчитывается не в первый раз и с момента последнет
        //--- для расчета добавилось не более одного бара
        values_to_copy=(rates_total-prev_calculated)+1;
    }
    //--- заполняем массивы значениями из индикатора iSAR
    //--- если FillArrayFromBuffer вернула false, значит данные не готовы - завершаем раб
    if(!FillArrayFromBuffer(iSARBuffer,handle,values_to_copy)) return(0);
    //--- сформируем сообщение
    string comm=StringFormat("%s ==> Обновлено значений в индикаторе %s: %d",
                               TimeToString(TimeCurrent(),TIME_DATE|TIME_SECONDS),
                               short_name,
                               values_to_copy);
    //--- выведем на график служебное сообщение
    Comment(comm);
    //--- запомним количество значений в индикаторе Parabolic SAR
    bars_calculated=calculated;
    //--- вернем значение prev_calculated для следующего вызова
    return(rates_total);
}

//+-----+
//| Заполняем индикаторный буфер из индикатора iSAR |
//+-----+
bool FillArrayFromBuffer(double &sar_buffer[], // индикаторный буфер значений Parabo
                        int ind_handle,        // хэндл индикатора iSAR
                        int amount             // количество копируемых значений
                        )
{
    //--- сбросим код ошибки
    ResetLastError();

```

```
//--- заполняем часть массива iSARBuffer значениями из индикаторного буфера под индекс
if(CopyBuffer(ind_handle,0,0,amount,sar_buffer)<0)
{
    //--- если копирование не удалось, сообщим код ошибки
    PrintFormat("Не удалось скопировать данные из индикатора iSAR, код ошибки %d",G
    //--- завершим с нулевым результатом - это означает, что индикатор будет считат
    return(false);
}
//--- все получилось
return(true);
}
//+-----+
//| Indicator deinitialization function |
//+-----+
void OnDeinit(const int reason)
{
    //--- почистим график при удалении индикатора
    Comment("");
}
```

iRSI

函数返回相对强度指数指标处理器。只有一个缓冲区。

```
int iRSI(
    string          symbol,          // 交易品种类型
    ENUM_TIMEFRAMES period,          // 周期
    int             ma_period,        // 平均周期
    ENUM_APPLIED_PRICE applied_price // 价格或者处理器类型
);
```

参量

symbol

[in] 证券交易品种名称，数据用来计算指标。 [NULL](#) 值代表当前交易品种。

period

[in] 周期值可以是 [ENUM_TIMEFRAMES](#) 值中的一个，0代表当前时间表。

ma_period

[in] RSI计算平均周期。

applied_price

[in] 使用价格。可以是任意 [ENUM_APPLIED_PRICE](#) 价格常量或者另外指标处理器。

返回值

返回特殊技术指标处理器，失败返回 [INVALID_HANDLE](#)。计算机内存从不使用的指标中释放，使用指标处理程序传递到的函数 [IndicatorRelease\(\)](#)。

:

```
//+-----+
//|                                     Demo_iRSI.mq5 |
//|                                     Copyright 2011, MetaQuotes Software Corp. |
//|                                     http://www.mql5.com |
//+-----+

#property copyright "Copyright 2011, MetaQuotes Software Corp."
#property link      "http://www.mql5.com"
#property version   "1.00"
#property description "Индикатор демонстрирует как нужно получать данные"
#property description "индикаторных буферов для технического индикатора iRSI."
#property description "Символ и таймфрейм, на котором рассчитывается индикатор,"
#property description "задаются параметрами symbol и period."
#property description "Способ создания хэндла задается параметром 'type' (тип функции"
#property description "Все остальные параметры как в стандартном Relative Strength In

#property indicator_separate_window
#property indicator_buffers 1
#property indicator_plots   1
//--- построение iRSI
#property indicator_label1  "iRSI"
```

```

#property indicator_type1    DRAW_LINE
#property indicator_color1   clrDodgerBlue
#property indicator_style1   STYLE_SOLID
#property indicator_width1   1
//--- пределы для отображения значений в окне индикатора
#property indicator_maximum 100
#property indicator_minimum 0
//--- горизонтальные уровни в окне индикатора
#property indicator_level1   70.0
#property indicator_level2   30.0
//+-----+
//|  перечисление способов создания хэндла  |
//+-----+
enum Creation
{
    Call_iRSI,           // использовать iRSI
    Call_IndicatorCreate // использовать IndicatorCreate
};
//--- входные параметры
input Creation          type=Call_iRSI;           // тип функции
input int               ma_period=14;             // период усреднения
input ENUM_APPLIED_PRICE applied_price=PRICE_CLOSE; // тип цены
input string            symbol=" ";               // символ
input ENUM_TIMEFRAMES   period=PERIOD_CURRENT;    // таймфрейм
//--- индикаторный буфер
double iRSIBuffer[];
//--- переменная для хранения хэндла индикатора iRSI
int handle;
//--- переменная для хранения
string name=symbol;
//--- имя индикатора на графике
string short_name;
//--- будем хранить количество значений в индикаторе Relative Strength Index
int bars_calculated=0;
//+-----+
//| Custom indicator initialization function |
//+-----+
int OnInit()
{
    //--- привязка массива к индикаторному буферу
    SetIndexBuffer(0,iRSIBuffer,INDICATOR_DATA);
    //--- определимся с символом, на котором строится индикатор
    name=symbol;
    //--- удалим пробелы слева и справа
    StringTrimRight(name);
    StringTrimLeft(name);
    //--- если после этого длина строки name нулевая
    if(StringLen(name)==0)

```



```

    {
        //--- возьмем символ с графика, на котором запущен индикатор
        name=_Symbol;
    }
//--- создадим хэндл индикатора
if(type==Call_iRSI)
    handle=iRSI(name,period,ma_period,applied_price);
else
    {
        //--- заполним структуру значениями параметров индикатора
        MqlParam pars[2];
        //--- период средней
        pars[0].type=TYPE_INT;
        pars[0].integer_value=ma_period;
        //--- предельное значение шага, которое может использоваться при расчетах
        pars[1].type=TYPE_INT;
        pars[1].integer_value=applied_price;
        handle=IndicatorCreate(name,period,IND_RSI,2,pars);
    }
//--- если не удалось создать хэндл
if(handle==INVALID_HANDLE)
    {
        //--- сообщим о неудаче и выведем номер ошибки
        PrintFormat("Не удалось создать хэндл индикатора iRSI для пары %s/%s, код ошибки",
            name,
            EnumToString(period),
            GetLastError());
        //--- работа индикатора завершается досрочно при возврате отрицательного значения
        return(-1);
    }
//--- покажем на какой паре символ/таймфрейм рассчитан индикатор Relative Strength Index
short_name=StringFormat("iRSI(%s/%s, %d, %d)",name,EnumToString(period),
    ma_period,applied_price);
IndicatorSetString(INDICATOR_SHORTNAME,short_name);
//--- нормальное выполнение инициализации индикатора
return(0);
}
//+-----+
//| Custom indicator iteration function |
//+-----+
int OnCalculate(const int rates_total,
    const int prev_calculated,
    const datetime &time[],
    const double &open[],
    const double &high[],
    const double &low[],
    const double &close[],
    const long &tick_volume[],

```

```

        const long &volume[],
        const int &spread[])
    {
//--- количество копируемых значений из индикатора iRSI
        int values_to_copy;
//--- узнаем количество рассчитанных значений в индикаторе
        int calculated=BarsCalculated(handle);
        if(calculated<=0)
        {
            PrintFormat("BarsCalculated() вернул %d, код ошибки %d",calculated,GetLastError());
            return(0);
        }
//--- если это первый запуск вычислений нашего индикатора или изменилось количество э
//--- или если необходимо рассчитать индикатор для двух или более баров (значит что-т
        if(prev_calculated==0 || calculated!=bars_calculated || rates_total>prev_calculated)
        {
            //--- если массив iRSIBuffer больше, чем значений в индикаторе iRSI на паре sym
            //--- в противном случае копировать будем меньше, чем размер индикаторных буфер
            if(calculated>rates_total) values_to_copy=rates_total;
            else
                values_to_copy=calculated;
        }
        else
        {
            //--- значит наш индикатор рассчитывается не в первый раз и с момента последнег
            //--- для расчета добавилось не более одного бара
            values_to_copy=(rates_total-prev_calculated)+1;
        }
//--- заполняем массив значениями из индикатора iRSI
//--- если FillArrayFromBuffer вернула false, значит данные не готовы - завершаем раб
        if(!FillArrayFromBuffer(iRSIBuffer,handle,values_to_copy)) return(0);
//--- сформируем сообщение
        string comm=StringFormat("%s ==> Обновлено значений в индикаторе %s: %d",
                                   TimeToString(TimeCurrent(),TIME_DATE|TIME_SECONDS),
                                   short_name,
                                   values_to_copy);
//--- выведем на график служебное сообщение
        Comment(comm);
//--- запоним количество значений в индикаторе Relative Strength Index
        bars_calculated=calculated;
//--- вернем значение prev_calculated для следующего вызова
        return(rates_total);
    }
//+-----+
//| Заполняем индикаторный буфер из индикатора iRSI |
//+-----+
bool FillArrayFromBuffer(double &rsi_buffer[], // индикаторный буфер значений Relati
                        int ind_handle,        // хэндл индикатора iSAR
                        int amount             // количество копируемых значений

```

```

    )

{
//--- сбросим код ошибки
    ResetLastError();
//--- заполняем часть массива iRSIBuffer значениями из индикаторного буфера под индекс
    if(CopyBuffer(ind_handle,0,0,amount,rsi_buffer)<0)
    {
        //--- если копирование не удалось, сообщим код ошибки
        PrintFormat("Не удалось скопировать данные из индикатора iRSI, код ошибки %d",G
        //--- завершим с нулевым результатом - это означает, что индикатор будет считат
        return(false);
    }
//--- все получилось
    return(true);
}

//+-----+
//| Indicator deinitialization function |
//+-----+
void OnDeinit(const int reason)
{
//--- почистим график при удалении индикатора
    Comment("");
}

```

iRVI

函数返回相对活力指数指标处理器。

```
int iRVI(
    string      symbol,      // 交易品种名称
    ENUM_TIMEFRAMES period,  // 周期
    int         ma_period    // 平均周期
);
```

参量

symbol

[in] 证券交易品种名称，数据用来计算指标。 [NULL](#) 值代表当前交易品种。

period

[in] 周期值可以是 [ENUM_TIMEFRAMES](#) 值中的一个，0代表当前时间表。

ma_period

[in] RVI 计算平均周期。

返回值

返回特殊技术指标处理器，失败返回 [INVALID_HANDLE](#)。计算机内存从不使用的指标中释放，使用指标处理程序传递到的函数 [IndicatorRelease\(\)](#)。

注释

缓冲区代码如下：0 - MAIN_ LINE，1 - SIGNAL_ LINE。

:

```
//+-----+
//|                                     Demo_iRVI.mq5 |
//|                                     Copyright 2011, MetaQuotes Software Corp. |
//|                                     http://www.mql5.com |
//+-----+

#property copyright "Copyright 2011, MetaQuotes Software Corp."
#property link      "http://www.mql5.com"
#property version   "1.00"
#property description "Индикатор демонстрирует как нужно получать данные"
#property description "индикаторных буферов для технического индикатора iRVI."
#property description "Символ и таймфрейм, на котором рассчитывается индикатор,"
#property description "задаются параметрами symbol и period."
#property description "Способ создания хэндла задается параметром 'type' (тип функции"
#property description "Все остальные параметры как в стандартном Relative Vigor Index"

#property indicator_separate_window
#property indicator_buffers 2
#property indicator_plots   2
//--- построение RVI
#property indicator_label1  "RVI"
```

```

#property indicator_type1   DRAW_LINE
#property indicator_color1  clrGreen
#property indicator_style1  STYLE_SOLID
#property indicator_width1  1
//--- построение Signal
#property indicator_label2  "Signal"
#property indicator_type2   DRAW_LINE
#property indicator_color2  clrRed
#property indicator_style2  STYLE_SOLID
#property indicator_width2  1
//+-----+
//|  перечисление способов создания хэндла  |
//+-----+
enum Creation
{
    Call_iRVI,           // использовать iRVI
    Call_IndicatorCreate // использовать IndicatorCreate
};
//--- входные параметры
input Creation      type=Call_iRVI;           // тип функции
input int           ma_period=10;             // период для расчетов
input string        symbol=" ";              // символ
input ENUM_TIMEFRAMES period=PERIOD_CURRENT; // таймфрейм
//--- индикаторные буферы
double      RVIBuffer[];
double      SignalBuffer[];
//--- переменная для хранения хэндла индикатора iRVI
int    handle;
//--- переменная для хранения
string name=symbol;
//--- имя индикатора на графике
string short_name;
//--- будем хранить количество значений в индикаторе Relative Vigor Index
int    bars_calculated=0;
//+-----+
//| Custom indicator initialization function  |
//+-----+
int OnInit()
{
    //--- привязка массивов к индикаторным буферам
    SetIndexBuffer(0,RVIBuffer,INDICATOR_DATA);
    SetIndexBuffer(1,SignalBuffer,INDICATOR_DATA);
    //--- определимся с символом, на котором строится индикатор
    name=symbol;
    //--- удалим пробелы слева и справа
    StringTrimRight(name);
    StringTrimLeft(name);
    //--- если после этого длина строки name нулевая

```

```

    if (StringLen(name)==0)
    {
        //--- возьмем символ с графика, на котором запущен индикатор
        name=_Symbol;
    }
//--- создадим хэндл индикатора
    if (type==Call_iRVI)
        handle=iRVI(name,period,ma_period);
    else
    {
        //--- заполним структуру значениями параметров индикатора
        MqlParam pars[1];
        //--- период для расчетов
        pars[0].type=TYPE_INT;
        pars[0].integer_value=ma_period;
        handle=IndicatorCreate(name,period,IND_RVI,1,pars);
    }
//--- если не удалось создать хэндл
    if (handle==INVALID_HANDLE)
    {
        //--- сообщим о неудаче и выведем номер ошибки
        PrintFormat("Не удалось создать хэндл индикатора iRVI для пары %s/%s, код ошибки",
                    name,
                    EnumToString(period),
                    GetLastError());
        //--- работа индикатора завершается досрочно при возврате отрицательного значения
        return(-1);
    }
//--- покажем на какой паре символ/таймфрейм рассчитан индикатор Relative Vigor Index
    short_name=StringFormat("iRSI(%s/%s, %d, %d)",name,EnumToString(period),ma_period);
    IndicatorSetString(INDICATOR_SHORTNAME,short_name);
//--- нормальное выполнение инициализации индикатора
    return(0);
}

//+-----+
//| Custom indicator iteration function |
//+-----+

int OnCalculate(const int rates_total,
                const int prev_calculated,
                const datetime &time[],
                const double &open[],
                const double &high[],
                const double &low[],
                const double &close[],
                const long &tick_volume[],
                const long &volume[],
                const int &spread[])
{

```

```

//--- количество копируемых значений из индикатора iRVI
int values_to_copy;
//--- узнаем количество рассчитанных значений в индикаторе
int calculated=BarsCalculated(handle);
if(calculated<=0)
{
    PrintFormat("BarsCalculated() вернул %d, код ошибки %d",calculated,GetLastError());
    return(0);
}
//--- если это первый запуск вычислений нашего индикатора или изменилось количество з
//--- или если необходимо рассчитать индикатор для двух или более баров (значит что-т
if(prev_calculated==0 || calculated!=bars_calculated || rates_total>prev_calculate
{
    //--- если массив RVIBuffer больше, чем значений в индикаторе iRVI на паре symb
    //--- в противном случае копировать будем меньше, чем размер индикаторных буфер
    if(calculated>rates_total) values_to_copy=rates_total;
    else
        values_to_copy=calculated;
}
else
{
    //--- значит наш индикатор рассчитывается не в первый раз и с момента последнет
    //--- для расчета добавилось не более одного бара
    values_to_copy=(rates_total-prev_calculated)+1;
}
//--- заполняем массивы значениями из индикатора iRVI
//--- если FillArrayFromBuffer вернула false, значит данные не готовы - завершаем раб
if(!FillArrayFromBuffer(RVIBuffer,SignalBuffer,handle,values_to_copy)) return(0);
//--- сформируем сообщение
string comm=StringFormat("%s ==> Обновлено значений в индикаторе %s: %d",
                        TimeToString(TimeCurrent(),TIME_DATE|TIME_SECONDS),
                        short_name,
                        values_to_copy);
//--- выведем на график служебное сообщение
Comment(comm);
//--- заппомним количество значений в индикаторе Relative Vigor Index
bars_calculated=calculated;
//--- вернем значение prev_calculated для следующего вызова
return(rates_total);
}
//+-----+
//| Заполняем индикаторные буферы из индикатора iRVI |
//+-----+
bool FillArrayFromBuffer(double &rvi_buffer[], // индикаторный буфер значений Rel
                        double &signal_buffer[], // индикаторный буфер сигнальной т
                        int ind_handle, // хэндл индикатора iRVI
                        int amount // количество копируемых значений
                        )
{

```

```

//--- сбросим код ошибки
ResetLastError();
//--- заполняем часть массива iRVIBuffer значениями из индикаторного буфера под индекс
if(CopyBuffer(ind_handle,0,0,amount,rvi_buffer)<0)
{
    //--- если копирование не удалось, сообщим код ошибки
    PrintFormat("Не удалось скопировать данные из индикатора iRVI, код ошибки %d",G
    //--- завершим с нулевым результатом - это означает, что индикатор будет считат
    return(false);
}
//--- заполняем часть массива SignalBuffer значениями из индикаторного буфера под инд
if(CopyBuffer(ind_handle,1,0,amount,signal_buffer)<0)
{
    //--- если копирование не удалось, сообщим код ошибки
    PrintFormat("Не удалось скопировать данные из индикатора iRVI, код ошибки %d",G
    //--- завершим с нулевым результатом - это означает, что индикатор будет считат
    return(false);
}
//--- все получилось
return(true);
}
//+-----+
//| Indicator deinitialization function |
//+-----+
void OnDeinit(const int reason)
{
    //--- почистим график при удалении индикатора
    Comment("");
}

```


iStdDev

函数返回标准偏差指标处理器。只有一个缓冲区。

```
int iStdDev(
    string          symbol,           // 交易品种名称
    ENUM_TIMEFRAMES period,          // 周期
    int             ma_period,        // 平均周期
    int             ma_shift,         // 平移
    ENUM_MA_METHOD  ma_method,        // 平滑类型
    ENUM_APPLIED_PRICE applied_price // 价格或者处理器类型
);
```

参量

symbol

[in] 证券交易品种名称，数据用来计算指标。 [NULL](#) 值代表当前交易品种。

period

[in] 周期值可以是 [ENUM_TIMEFRAMES](#) 值中的一个，0代表当前时间表。

ma_period

[in] 指标计算平均周期。

ma_shift

[in] 与价格图表有关的指标变化。

ma_method

[in] 平均类型。可以使 [ENUM_MA_METHOD](#) 中任意值。

applied_price

[in] 使用价格。可以是任意 [ENUM_APPLIED_PRICE](#) 价格常量或者另外指标处理器。

返回值

返回特殊技术指标处理器，失败返回 [INVALID_HANDLE](#)。计算机内存从不使用的指标中释放，使用指标处理程序传递到的函数 [IndicatorRelease\(\)](#)。

```
//+-----+
//|                                     Demo_iStdDev.mq5 |
//|                                     Copyright 2011, MetaQuotes Software Corp. |
//|                                     http://www.mql5.com |
//+-----+
#property copyright "Copyright 2011, MetaQuotes Software Corp."
#property link      "http://www.mql5.com"
#property version   "1.00"
#property description "Индикатор демонстрирует как нужно получать данные"
#property description "индикаторных буферов для технического индикатора iStdDev."
#property description "Символ и таймфрейм, на котором рассчитывается индикатор,"
#property description "задаются параметрами symbol и period."
```

```

#property description "Способ создания хэндла задается параметром 'type' (тип функции"
#property description "Все остальные параметры как в стандартном Standard Deviation."

#property indicator_separate_window
#property indicator_buffers 1
#property indicator_plots 1
//--- построение iStdDev
#property indicator_label1 "iStdDev"
#property indicator_type1 DRAW_LINE
#property indicator_color1 clrMediumSeaGreen
#property indicator_style1 STYLE_SOLID
#property indicator_width1 1
//+-----+
//| перечисление способов создания хэндла |
//+-----+
enum Creation
{
    Call_iStdDev,          // использовать iStdDev
    Call_IndicatorCreate   // использовать IndicatorCreate
};
//--- входные параметры
input Creation            type=Call_iStdDev;          // тип функции
input int                 ma_period=20;              // период усреднения
input int                 ma_shift=0;                // смещение
input ENUM_MA_METHOD      ma_method=MODE_SMA;        // тип сглаживания
input ENUM_APPLIED_PRICE  applied_price=PRICE_CLOSE; // тип цены
input string              symbol=" ";                // символ
input ENUM_TIMEFRAMES     period=PERIOD_CURRENT;    // таймфрейм
//--- индикаторный буфер
double iStdDevBuffer[];
//--- переменная для хранения хэндла индикатора iStdDev
int handle;
//--- переменная для хранения
string name=symbol;
//--- имя индикатора на графике
string short_name;
//--- будем хранить количество значений в индикаторе Standard Deviation
int bars_calculated=0;
//+-----+
//| Custom indicator initialization function |
//+-----+
int OnInit()
{
    //--- привязка массива к индикаторному буферу
    SetIndexBuffer(0,iStdDevBuffer,INDICATOR_DATA);
    //--- зададим смещение
    PlotIndexSetInteger(0,PLOT_SHIFT,ma_shift);
    //--- определимся с символом, на котором строится индикатор

```

```

    name=symbol;
//--- удалим пробелы слева и справа
    StringTrimRight(name);
    StringTrimLeft(name);
//--- если после этого длина строки name нулевая
    if(StringLen(name)==0)
    {
        //--- возьмем символ с графика, на котором запущен индикатор
        name=_Symbol;
    }
//--- создадим хэндл индикатора
    if(type==Call_iStdDev)
        handle=iStdDev(name,period,ma_period,ma_shift,ma_method,applied_price);
    else
    {
        //--- заполним структуру значениями параметров индикатора
        MqlParam pars[4];
        //--- период
        pars[0].type=TYPE_INT;
        pars[0].integer_value=ma_period;
        //--- смещение
        pars[1].type=TYPE_INT;
        pars[1].integer_value=ma_shift;
        //--- тип сглаживания
        pars[2].type=TYPE_INT;
        pars[2].integer_value=ma_method;
        //--- тип цены
        pars[3].type=TYPE_INT;
        pars[3].integer_value=applied_price;
        handle=IndicatorCreate(name,period,IND_STDDEV,4,pars);
    }
//--- если не удалось создать хэндл
    if(handle==INVALID_HANDLE)
    {
        //--- сообщим о неудаче и выведем номер ошибки
        PrintFormat("Не удалось создать хэндл индикатора iStdDev для пары %s/%s, код оши
            name,
            EnumToString(period),
            GetLastError());
        //--- работа индикатора завершается досрочно при возврате отрицательного значен
        return(-1);
    }
//--- покажем на какой паре символ/таймфрейм рассчитан индикатор Standard Deviation
    short_name=StringFormat("iStdDev(%s/%s, %d, %d, %s, %s)",name,EnumToString(period)
        ma_period,ma_shift,EnumToString(ma_method),EnumToString(ap
    IndicatorSetString(INDICATOR_SHORTNAME,short_name);
//--- нормальное выполнение инициализации индикатора
    return(0);

```

```

    }
//+-----+
//| Custom indicator iteration function |
//+-----+
int OnCalculate(const int rates_total,
               const int prev_calculated,
               const datetime &time[],
               const double &open[],
               const double &high[],
               const double &low[],
               const double &close[],
               const long &tick_volume[],
               const long &volume[],
               const int &spread[])
{
//--- количество копируемых значений из индикатора iStdDev
    int values_to_copy;
//--- узнаем количество рассчитанных значений в индикаторе
    int calculated=BarsCalculated(handle);
    if(calculated<=0)
    {
        PrintFormat("BarsCalculated() вернул %d, код ошибки %d",calculated,GetLastError());
        return(0);
    }
//--- если это первый запуск вычислений нашего индикатора или изменилось количество э
//--- или если необходимо рассчитать индикатор для двух или более баров (значит что-т
    if(prev_calculated==0 || calculated!=bars_calculated || rates_total>prev_calculate
    {
        //--- если массив iStdDevBuffer больше, чем значений в индикаторе iStdDev на па
        //--- в противном случае копировать будем меньше, чем размер индикаторных буфер
        if(calculated>rates_total) values_to_copy=rates_total;
        else
            values_to_copy=calculated;
    }
    else
    {
        //--- значит наш индикатор рассчитывается не в первый раз и с момента последнег
        //--- для расчета добавилось не более одного бара
        values_to_copy=(rates_total-prev_calculated)+1;
    }
//--- заполняем массив значениями из индикатора Standard Deviation
//--- если FillArrayFromBuffer вернула false, значит данные не готовы - завершаем раб
    if(!FillArrayFromBuffer(iStdDevBuffer,ma_shift,handle,values_to_copy)) return(0);
//--- сформируем сообщение
    string comm=StringFormat("%s ==> Обновлено значений в индикаторе %s: %d",
                             TimeToString(TimeCurrent(),TIME_DATE|TIME_SECONDS),
                             short_name,
                             values_to_copy);
//--- выведем на график служебное сообщение

```

```

    Comment(comm);
//--- заппомним количество значений в индикаторе Standard Deviation
    bars_calculated=calculated;
//--- вернем значение prev_calculated для следующего вызова
    return(rates_total);
}
//+-----+
//| Заполняем индикаторный буфер из индикатора iStdDev |
//+-----+
bool FillArrayFromBuffer(double &std_buffer[], // индикаторный буфер линии Standard
                        int std_shift,         // смещение линии Standard Deviation
                        int ind_handle,        // хэндл индикатора iStdDev
                        int amount            // количество копируемых значений
                        )
{
//--- сбросим код ошибки
    ResetLastError();
//--- заполняем часть массива iStdDevBuffer значениями из индикаторного буфера под инд
    if(CopyBuffer(ind_handle,0,-std_shift,amount,std_buffer)<0)
    {
        //--- если копирование не удалось, сообщим код ошибки
        PrintFormat("Не удалось скопировать данные из индикатора iStdDev, код ошибки %d", GetLastError());
        //--- завершим с нулевым результатом - это означает, что индикатор будет считат
        return(false);
    }
//--- все получилось
    return(true);
}
//+-----+
//| Indicator deinitialization function |
//+-----+
void OnDeinit(const int reason)
{
//--- почистим график при удалении индикатора
    Comment("");
}

```

iStochastic

函数返回随机摆动指标处理器。

```
int iStochastic(
    string      symbol,           // 交易品种名称
    ENUM_TIMEFRAMES period,      // 周期
    int         Kperiod,         // K线周期（用于计算的柱数）
    int         Dperiod,         // D线周期（开始平滑周期）
    int         slowing,         // 最终平滑
    ENUM_MA_METHOD ma_method,    // 平滑类型
    ENUM_STO_PRICE price_field   // 随机算法
);
```

参量

symbol

[in] 证券交易品种名称，数据用来计算指标。 [NULL](#) 值代表当前交易品种。

period

[in] 周期值可以是 [ENUM_TIMEFRAMES](#) 值中的一个，0代表当前时间表。

Kperiod

[in] %K 线计算平均周期（计算柱）。

Dperiod

[in] %D 线计算平均周期（计算柱）。

slowing

[in] 减速值。

ma_method

[in] 平均类型。可以是 [ENUM_MA_METHOD](#) 中任意值。

price_field

[in] 计算价格选择参量。可以是 [ENUM_STO_PRICE](#) 值中的一个。

返回值

返回特殊技术指标处理器，失败返回 [INVALID_HANDLE](#)。计算机内存从不使用的指标中释放，使用指标处理程序传递到的函数 [IndicatorRelease\(\)](#)。

注释

缓冲区代码：0 - MAIN_ LINE，1 - SIGNAL_ LINE。

:

```
//+-----+
//|                                     Demo_iStochastic.mq5 |
//|                                     Copyright 2011, MetaQuotes Software Corp. |
//|                                     http://www.mql5.com |
//+-----+
```

```

#property copyright "Copyright 2011, MetaQuotes Software Corp."
#property link      "http://www.mql5.com"
#property version   "1.00"
#property description "Индикатор демонстрирует как нужно получать данные"
#property description "индикаторных буферов для технического индикатора iStochastic."
#property description "Символ и таймфрейм, на котором рассчитывается индикатор,"
#property description "задаются параметрами symbol и period."
#property description "Способ создания хэндла задается параметром 'type' (тип функции"
#property description "Все остальные параметры как в стандартном Stochastic Oscillato

#property indicator_separate_window
#property indicator_buffers 2
#property indicator_plots 2
//--- построение Stochastic
#property indicator_label1 "Stochastic"
#property indicator_type1  DRAW_LINE
#property indicator_color1  clrLightSeaGreen
#property indicator_style1  STYLE_SOLID
#property indicator_width1  1
//--- построение Signal
#property indicator_label2 "Signal"
#property indicator_type2  DRAW_LINE
#property indicator_color2  clrRed
#property indicator_style2  STYLE_SOLID
#property indicator_width2  1
//--- зададим граничные значения индикатора
#property indicator_minimum 0
#property indicator_maximum 100
//--- горизонтальные уровни в окне индикатора
#property indicator_level1 -100.0
#property indicator_level2 100.0
//+-----+
//|  перечисление способов создания хэндла  |
//+-----+
enum Creation
{
    Call_iStochastic,      // использовать iStochastic
    Call_IndicatorCreate    // использовать IndicatorCreate
};
//--- входные параметры
input Creation      type=Call_iStochastic;    // тип функции
input int           Kperiod=5;                // K-период (количество баров д
input int           Dperiod=3;                // D-период (период первичного
input int           slowing=3;                // период для окончательного ст
input ENUM_MA_METHOD ma_method=MODE_SMA;      // тип сглаживания
input ENUM_STO_PRICE price_field=STO_LOWHIGH; // способ расчета стохастика
input string        symbol=" ";              // символ
input ENUM_TIMEFRAMES period=PERIOD_CURRENT; // таймфрейм

```

```

//--- индикаторные буферы
double      StochasticBuffer[];
double      SignalBuffer[];
//--- переменная для хранения хэндла индикатора iStochastic
int    handle;
//--- переменная для хранения
string name=symbol;
//--- имя индикатора на графике
string short_name;
//--- будем хранить количество значений в индикаторе Stochastic Oscillator
int    bars_calculated=0;
//+-----+
//| Custom indicator initialization function |
//+-----+
int OnInit()
{
//--- привязка массивов к индикаторным буферам
    SetIndexBuffer(0,StochasticBuffer,INDICATOR_DATA);
    SetIndexBuffer(1,SignalBuffer,INDICATOR_DATA);
//--- определимся с символом, на котором строится индикатор
    name=symbol;
//--- удалим пробелы слева и справа
    StringTrimRight(name);
    StringTrimLeft(name);
//--- если после этого длина строки name нулевая
    if(StringLen(name)==0)
    {
        //--- возьмем символ с графика, на котором запущен индикатор
        name=_Symbol;
    }
//--- создадим хэндл индикатора
    if(type==Call_iStochastic)
        handle=iStochastic(name,period,Kperiod,Dperiod,slowing,ma_method,price_field);
    else
    {
        //--- заполним структуру значениями параметров индикатора
        MqlParam pars[5];
        //--- период K для расчетов
        pars[0].type=TYPE_INT;
        pars[0].integer_value=Kperiod;
        //--- период D для первичного сглаживания
        pars[1].type=TYPE_INT;
        pars[1].integer_value=Dperiod;
        //--- период K для окончательного сглаживания
        pars[2].type=TYPE_INT;
        pars[2].integer_value=slowing;
        //--- тип сглаживания
        pars[3].type=TYPE_INT;

```



```

    pars[3].integer_value=ma_method;
    ///--- способ расчета стохастика
    pars[4].type=TYPE_INT;
    pars[4].integer_value=price_field;
    handle=IndicatorCreate(name,period,IND_STOCHASTIC,5,pars);
}
///--- если не удалось создать хэндл
if(handle==INVALID_HANDLE)
{
    ///--- сообщим о неудаче и выведем номер ошибки
    PrintFormat("Не удалось создать хэндл индикатора iStochastic для пары %s/%s, код ошибки %d",
        name,
        EnumToString(period),
        GetLastError());
    ///--- работа индикатора завершается досрочно при возврате отрицательного значения
    return(-1);
}
///--- покажем на какой паре символ/таймфрейм рассчитан индикатор Stochastic Oscillator
short_name=StringFormat("iStochastic(%s/%s, %d, %d, %d, %s, %s)",name,EnumToString(period),Kperiod,Dperiod,slowing,EnumToString(ma_method),EnumToString(price_field));
IndicatorSetString(INDICATOR_SHORTNAME,short_name);
///--- нормальное выполнение инициализации индикатора
return(0);
}
//+-----+
//| Custom indicator iteration function |
//+-----+
int OnCalculate(const int rates_total,
    const int prev_calculated,
    const datetime &time[],
    const double &open[],
    const double &high[],
    const double &low[],
    const double &close[],
    const long &tick_volume[],
    const long &volume[],
    const int &spread[])
{
    ///--- количество копируемых значений из индикатора iStochastic
    int values_to_copy;
    ///--- узнаем количество рассчитанных значений в индикаторе
    int calculated=BarsCalculated(handle);
    if(calculated<=0)
    {
        PrintFormat("BarsCalculated() вернул %d, код ошибки %d",calculated,GetLastError());
        return(0);
    }
    ///--- если это первый запуск вычислений нашего индикатора или изменилось количество з

```

```

//--- или если необходимо рассчитать индикатор для двух или более баров (значит что-т
if(prev_calculated==0 || calculated!=bars_calculated || rates_total>prev_calculate
{
    //--- если массив StochasticBuffer больше, чем значений в индикаторе iStochastic
    //--- в противном случае копировать будем меньше, чем размер индикаторных буфер
    if(calculated>rates_total) values_to_copy=rates_total;
    else
        values_to_copy=calculated;
}
else
{
    //--- значит наш индикатор рассчитывается не в первый раз и с момента последнег
    //--- для расчета добавилось не более одного бара
    values_to_copy=(rates_total-prev_calculated)+1;
}

//--- заполняем массивы значениями из индикатора iStochastic
//--- если FillArraysFromBuffers вернула false, значит данные не готовы - завершаем р
if(!FillArraysFromBuffers(StochasticBuffer,SignalBuffer,handle,values_to_copy)) re
//--- сформируем сообщение
string comm=StringFormat("%s ==> Обновлено значений в индикаторе %s: %d",
                        TimeToString(TimeCurrent(),TIME_DATE|TIME_SECONDS),
                        short_name,
                        values_to_copy);

//--- выведем на график служебное сообщение
Comment(comm);

//--- заппомним количество значений в индикаторе Stochastic Oscillator
bars_calculated=calculated;

//--- вернем значение prev_calculated для следующего вызова
return(rates_total);
}

//+-----+
//| Заполняем индикаторные буферы из индикатора iStochastic |
//+-----+

bool FillArraysFromBuffers(double &main_buffer[], // индикаторный буфер значений S
                        double &signal_buffer[], // индикаторный буфер сигнальной
                        int ind_handle, // хэндл индикатора iStochastic
                        int amount // количество копируемых значени
                        )
{
    //--- сбросим код ошибки
    ResetLastError();

    //--- заполняем часть массива StochasticBuffer значениями из индикаторного буфера под
    if(CopyBuffer(ind_handle,MAIN_LINE,0,amount,main_buffer)<0)
    {
        //--- если копирование не удалось, сообщим код ошибки
        PrintFormat("Не удалось скопировать данные из индикатора iStochastic, код ошибк
        //--- завершим с нулевым результатом - это означает, что индикатор будет считат
        return(false);
    }
}

```

```
//--- заполняем часть массива SignalBuffer значениями из индикаторного буфера под инд
if(CopyBuffer(ind_handle,SIGNAL_LINE,0,amount,signal_buffer)<0)
{
    //--- если копирование не удалось, сообщим код ошибки
    PrintFormat("Не удалось скопировать данные из индикатора iStochastic, код ошибки
    //--- завершим с нулевым результатом - это означает, что индикатор будет считат
    return(false);
}
//--- все получилось
return(true);
}
//+-----+
//| Indicator deinitialization function |
//+-----+
void OnDeinit(const int reason)
{
    //--- почистим график при удалении индикатора
    Comment("");
}
```

iTEMA

函数返回三倍指数移动平均指标处理器。只有一个缓冲区。

```
int iTEMA(
    string          symbol,          // 交易品种名称
    ENUM_TIMEFRAMES period,          // 周期
    int             ma_period,        // 平均周期
    int             ma_shift,         // 指标平移
    ENUM_APPLIED_PRICE applied_price // 价格或者处理器类型
);
```

参量

symbol

[in] 证券交易品种名称，数据用来计算指标。 [NULL](#) 值代表当前交易品种。

period

[in] 周期值可以是 [ENUM_TIMEFRAMES](#) 值中的一个，0代表当前时间表。

ma_period

[in] 计算平均周期（计算柱）。

ma_shift

[in] 与价格图表相关的指标变化。

applied_price

[in] 使用价格。可以是任意 [ENUM_APPLIED_PRICE](#) 价格常量或者另外指标处理器。

返回值

返回特殊技术指标处理器，失败返回 [INVALID_HANDLE](#)。计算机内存从不使用的指标中释放，使用指标处理程序传递到的函数 [IndicatorRelease\(\)](#)。

:

```
//+-----+
//|                                     Demo_iTEMA.mq5 |
//|                                     Copyright 2011, MetaQuotes Software Corp. |
//|                                     http://www.mql5.com |
//+-----+
#property copyright "Copyright 2011, MetaQuotes Software Corp."
#property link      "http://www.mql5.com"
#property version   "1.00"
#property description "Индикатор демонстрирует как нужно получать данные"
#property description "индикаторных буферов для технического индикатора iTEMA."
#property description "Символ и таймфрейм, на котором рассчитывается индикатор,"
#property description "задаются параметрами symbol и period."
#property description "Способ создания хэндла задается параметром 'type' (тип функции"
#property description "Все остальные параметры как в стандартном Triple Exponential M"

#property indicator_chart_window
```

```

#property indicator_buffers 1
#property indicator_plots 1
//--- построение iTEMA
#property indicator_label1 "iTEMA"
#property indicator_type1 DRAW_LINE
#property indicator_color1 clrRed
#property indicator_style1 STYLE_SOLID
#property indicator_width1 1
//+-----+
//| перечисление способов создания хэнгла |
//+-----+
enum Creation
{
    Call_iTEMA,          // использовать iTEMA
    Call_IndicatorCreate // использовать IndicatorCreate
};
//--- входные параметры
input Creation      type=Call_iTEMA;          // тип функции
input int           ma_period=14;             // период усреднения
input int           ma_shift=0;               // смещение
input ENUM_APPLIED_PRICE applied_price=PRICE_CLOSE; // тип цены
input string        symbol=" ";              // символ
input ENUM_TIMEFRAMES period=PERIOD_CURRENT; // таймфрейм
//--- индикаторный буфер
double iTEMABuffer[];
//--- переменная для хранения хэнгла индикатора iTEMA
int handle;
//--- переменная для хранения
string name=symbol;
//--- имя индикатора на графике
string short_name;
//--- будем хранить количество значений в индикаторе Triple Exponential Moving Average
int bars_calculated=0;
//+-----+
//| Custom indicator initialization function |
//+-----+
int OnInit()
{
    //--- привязка массива к индикаторному буферу
    SetIndexBuffer(0,iTEMABuffer,INDICATOR_DATA);
    //--- зададим смещение
    PlotIndexSetInteger(0,PLOT_SHIFT,ma_shift);
    //--- определимся с символом, на котором строится индикатор
    name=symbol;
    //--- удалим пробелы слева и справа
    StringTrimRight(name);
    StringTrimLeft(name);
    //--- если после этого длина строки name нулевая

```

```

    if (StringLen(name)==0)
    {
        //--- возьмем символ с графика, на котором запущен индикатор
        name=_Symbol;
    }
//--- создадим хэндл индикатора
    if (type==Call_iTEMA)
        handle=iTEMA(name,period,ma_period,ma_shift,applied_price);
    else
    {
        //--- заполним структуру значениями параметров индикатора
        MqlParam pars[3];
        //--- период
        pars[0].type=TYPE_INT;
        pars[0].integer_value=ma_period;
        //--- смещение
        pars[1].type=TYPE_INT;
        pars[1].integer_value=ma_shift;
        //--- тип цены
        pars[2].type=TYPE_INT;
        pars[2].integer_value=applied_price;
        handle=IndicatorCreate(name,period,IND_TEMA,3,pars);
    }
//--- если не удалось создать хэндл
    if (handle==INVALID_HANDLE)
    {
        //--- сообщим о неудаче и выведем номер ошибки
        PrintFormat("Не удалось создать хэндл индикатора iTEMA для пары %s/%s, код ошибки",
                    name,
                    EnumToString(period),
                    GetLastError());
        //--- работа индикатора завершается досрочно при возврате отрицательного значения
        return(-1);
    }
//--- покажем на какой паре символ/таймфрейм рассчитан индикатор Triple Exponential M
    short_name=StringFormat("iTEMA(%s/%s, %d, %d, %s)",name,EnumToString(period),
                             ma_period,ma_shift,EnumToString(applied_price));
    IndicatorSetString(INDICATOR_SHORTNAME,short_name);
//--- нормальное выполнение инициализации индикатора
    return(0);
}
//+-----+
//| Custom indicator iteration function |
//+-----+
int OnCalculate(const int rates_total,
               const int prev_calculated,
               const datetime &time[],
               const double &open[],

```

```

        const double &high[],
        const double &low[],
        const double &close[],
        const long &tick_volume[],
        const long &volume[],
        const int &spread[])

{
//--- количество копируемых значений из индикатора iTEMA
    int values_to_copy;
//--- узнаем количество рассчитанных значений в индикаторе
    int calculated=BarsCalculated(handle);
    if(calculated<=0)
    {
        PrintFormat("BarsCalculated() вернул %d, код ошибки %d",calculated,GetLastError());
        return(0);
    }
//--- если это первый запуск вычислений нашего индикатора или изменилось количество з
//--- или если необходимо рассчитать индикатор для двух или более баров (значит что-т
    if(prev_calculated==0 || calculated!=bars_calculated || rates_total>prev_calculated)
    {
        //--- если массив iTEMABuffer больше, чем значений в индикаторе iTEMA на паре s
        //--- в противном случае копировать будем меньше, чем размер индикаторных буфер
        if(calculated>rates_total) values_to_copy=rates_total;
        else
            values_to_copy=calculated;
    }
    else
    {
        //--- значит наш индикатор рассчитывается не в первый раз и с момента последнег
        //--- для расчета добавилось не более одного бара
        values_to_copy=(rates_total-prev_calculated)+1;
    }
//--- заполняем массив значениями из индикатора Triple Exponential Moving Average
//--- если FillArrayFromBuffer вернула false, значит данные не готовы - завершаем раб
    if(!FillArrayFromBuffer(iTEMABuffer,ma_shift,handle,values_to_copy)) return(0);
//--- сформируем сообщение
    string comm=StringFormat("%s ==> Обновлено значений в индикаторе %s: %d",
                               TimeToString(TimeCurrent(),TIME_DATE|TIME_SECONDS),
                               short_name,
                               values_to_copy);
//--- выведем на график служебное сообщение
    Comment(comm);
//--- запомним количество значений в индикаторе Triple Exponential Moving Average
    bars_calculated=calculated;
//--- вернем значение prev_calculated для следующего вызова
    return(rates_total);
}
//+-----+
//| Заполняем индикаторный буфер из индикатора iTEMA |

```

```

//+-----+
bool FillArrayFromBuffer(double &tema_buffer[], // индикаторный буфер значений Triple
                        int t_shift,           // смещение линии
                        int ind_handle,        // хэндл индикатора iTEMA
                        int amount             // количество копируемых значений
                        )
{
//--- сбросим код ошибки
    ResetLastError();
//--- заполняем часть массива iTEMABuffer значениями из индикаторного буфера под инде
    if(CopyBuffer(ind_handle,0,-t_shift,amount,tema_buffer)<0)
    {
        //--- если копирование не удалось, сообщим код ошибки
        PrintFormat("Не удалось скопировать данные из индикатора iTEMA, код ошибки %d",
        //--- завершим с нулевым результатом - это означает, что индикатор будет считат
        return(false);
    }
//--- все получилось
    return(true);
}
//+-----+
//| Indicator deinitialization function |
//+-----+
void OnDeinit(const int reason)
{
//--- почистим график при удалении индикатора
    Comment("");
}

```


iTriX

函数返回三倍指数移动平均数振荡指标处理器。只有一个缓冲区。

```
int iTriX(
    string          symbol,          // 交易品种名称
    ENUM_TIMEFRAMES period,          // 周期
    int             ma_period,        // 平均周期
    ENUM_APPLIED_PRICE applied_price // 价格或者处理器类型
);
```

参量

symbol

[in] 证券交易品种名称，数据用来计算指标。 [NULL](#) 值代表当前交易品种。

period

[in] 周期值可以是 [ENUM_TIMEFRAMES](#) 值中的一个，0代表当前时间表。

ma_period

[in] 计算平均周期（计算柱）。

applied_price

[in] 使用价格。可以是任意 [ENUM_APPLIED_PRICE](#) 价格常量或者另外指标处理器。

返回值

返回特殊技术指标处理器，失败返回 [INVALID_HANDLE](#)。计算机内存从不使用的指标中释放，使用指标处理程序传递到的函数 [IndicatorRelease\(\)](#)。

:

```
//+-----+
//|                                     Demo_iTriX.mq5 |
//|                                     Copyright 2011, MetaQuotes Software Corp. |
//|                                     http://www.mql5.com |
//+-----+

#property copyright "Copyright 2011, MetaQuotes Software Corp."
#property link      "http://www.mql5.com"
#property version   "1.00"
#property description "Индикатор демонстрирует как нужно получать данные"
#property description "индикаторных буферов для технического индикатора iTriX."
#property description "Символ и таймфрейм, на котором рассчитывается индикатор,"
#property description "задаются параметрами symbol и period."
#property description "Способ создания хэндла задается параметром 'type' (тип функции"

#property indicator_separate_window
#property indicator_buffers 1
#property indicator_plots   1
//--- построение iTriX
#property indicator_label1  "iTriX"
#property indicator_type1   DRAW_LINE
```

```

#property indicator_color1 clrRed
#property indicator_style1 STYLE_SOLID
#property indicator_width1 1
//+-----+
//|  перечисление способов создания хэндла  |
//+-----+
enum Creation
{
    Call_iTriX,          // использовать iTriX
    Call_IndicatorCreate // использовать IndicatorCreate
};
//--- входные параметры
input Creation      type=Call_iTriX;          // тип функции
input int           ma_period=14;             // период
input ENUM_APPLIED_PRICE applied_price=PRICE_CLOSE; // тип цены
input string        symbol=" ";              // символ
input ENUM_TIMEFRAMES period=PERIOD_CURRENT; // таймфрейм
//--- индикаторный буфер
double iTriXBuffer[];
//--- переменная для хранения хэндла индикатора iTriX
int handle;
//--- переменная для хранения
string name=symbol;
//--- имя индикатора на графике
string short_name;
//--- будем хранить количество значений в индикаторе Triple Exponential Moving Average
int bars_calculated=0;
//+-----+
//| Custom indicator initialization function  |
//+-----+
int OnInit()
{
    //--- привязка массива к индикаторному буферу
    SetIndexBuffer(0,iTriXBuffer,INDICATOR_DATA);
    //--- определимся с символом, на котором строится индикатор
    name=symbol;
    //--- удалим пробелы слева и справа
    StringTrimRight(name);
    StringTrimLeft(name);
    //--- если после этого длина строки name нулевая
    if(StringLen(name)==0)
    {
        //--- возьмем символ с графика, на котором запущен индикатор
        name=_Symbol;
    }
    //--- создадим хэндл индикатора
    if(type==Call_iTriX)
        handle=iTriX(name,period,ma_period,applied_price);
}

```

```

else
{
    //--- заполним структуру значениями параметров индикатора
    MqlParam pars[2];
    //--- период
    pars[0].type=TYPE_INT;
    pars[0].integer_value=ma_period;
    //--- тип цены
    pars[1].type=TYPE_INT;
    pars[1].integer_value=applied_price;
    handle=IndicatorCreate(name,period,IND_TRIX,2,pars);
}
//--- если не удалось создать хэндл
if(handle==INVALID_HANDLE)
{
    //--- сообщим о неудаче и выведем номер ошибки
    PrintFormat("Не удалось создать хэндл индикатора iTriX для пары %s/%s, код ошибки: %d",
                name,
                EnumToString(period),
                GetLastError());
    //--- работа индикатора завершается досрочно при возврате отрицательного значения
    return(-1);
}
//--- покажем на какой паре символ/таймфрейм рассчитан индикатор Triple Exponential Moving Average
short_name=StringFormat("iTriX(%s/%s, %d, %s)",name,EnumToString(period),
                        ma_period,EnumToString(applied_price));
IndicatorSetString(INDICATOR_SHORTNAME,short_name);
//--- нормальное выполнение инициализации индикатора
return(0);
}
//+-----+
//| Custom indicator iteration function |
//+-----+
int OnCalculate(const int rates_total,
                const int prev_calculated,
                const datetime &time[],
                const double &open[],
                const double &high[],
                const double &low[],
                const double &close[],
                const long &tick_volume[],
                const long &volume[],
                const int &spread[])
{
    //--- количество копируемых значений из индикатора iTriX
    int values_to_copy;
    //--- узнаем количество рассчитанных значений в индикаторе
    int calculated=BarsCalculated(handle);

```

```

    if(calculated<=0)
    {
        PrintFormat("BarsCalculated() вернул %d, код ошибки %d",calculated,GetLastError()
        return(0);
    }
//--- если это первый запуск вычислений нашего индикатора или изменилось количество з
//--- или если необходимо рассчитать индикатор для двух или более баров (значит что-т
    if(prev_calculated==0 || calculated!=bars_calculated || rates_total>prev_calculate
    {
        //--- если массив iTriXBuffer больше, чем значений в индикаторе iTriX на паре s
        //--- в противном случае копировать будем меньше, чем размер индикаторных буфер
        if(calculated>rates_total) values_to_copy=rates_total;
        else
            values_to_copy=calculated;
    }
    else
    {
        //--- значит наш индикатор рассчитывается не в первый раз и с момента последнег
        //--- для расчета добавилось не более одного бара
        values_to_copy=(rates_total-prev_calculated)+1;
    }
//--- заполняем массив значениями из индикатора Triple Exponential Moving Averages Os
//--- если FillArrayFromBuffer вернула false, значит данные не готовы - завершаем раб
    if(!FillArrayFromBuffer(iTriXBuffer,handle,values_to_copy)) return(0);
//--- сформируем сообщение
    string comm=StringFormat("%s ==> Обновлено значений в индикаторе %s: %d",
                                TimeToString(TimeCurrent(),TIME_DATE|TIME_SECONDS),
                                short_name,
                                values_to_copy);
//--- выведем на график служебное сообщение
    Comment(comm);
//--- запоним количество значений в индикаторе Triple Exponential Moving Averages Os
    bars_calculated=calculated;
//--- вернем значение prev_calculated для следующего вызова
    return(rates_total);
}
//+-----+
//| Заполняем индикаторный буфер из индикатора iTriX |
//+-----+
bool FillArrayFromBuffer(double &trix_buffer[], // индикаторный буфер значений Triple
                        int ind_handle,          // хэндл индикатора iTriX
                        int amount               // количество копируемых значений
                        )
{
    //--- сбросим код ошибки
    ResetLastError();
//--- заполняем часть массива iTriXBuffer значениями из индикаторного буфера под инде
    if(CopyBuffer(ind_handle,0,0,amount,trix_buffer)<0)
    {

```

```
//--- если копирование не удалось, сообщим код ошибки
PrintFormat("Не удалось скопировать данные из индикатора iTriX, код ошибки %d",
//--- завершим с нулевым результатом - это означает, что индикатор будет считат
return(false);
}
//--- все получилось
return(true);
}
//+-----+
//| Indicator deinitialization function |
//+-----+
void OnDeinit(const int reason)
{
//--- почистим график при удалении индикатора
Comment("");
}
```

iWPR

函数返回威廉指数指标。只有一个缓冲区。

```
int iWPR(
    string      symbol,           // 交易品种名称
    ENUM_TIMEFRAMES period,       // 周期
    int         calc_period       // 平均周期
);
```

参量

symbol

[in] 证券交易品种名称，数据用来计算指标。 [NULL](#) 值代表当前交易品种。

period

[in] 周期值可以是 [ENUM_TIMEFRAMES](#) 值中的一个，0代表当前时间表。

calc_period

[in] 指标计算周期（计算柱）。

返回值

返回特殊技术指标处理器，失败返回 [INVALID_HANDLE](#)。计算机内存从不使用的指标中释放，使用指标处理程序传递到的函数 [IndicatorRelease\(\)](#)。

:

```
//+-----+
//|                                     Demo_iWPR.mq5 |
//|                                     Copyright 2011, MetaQuotes Software Corp. |
//|                                     http://www.mql5.com |
//+-----+

#property copyright "Copyright 2011, MetaQuotes Software Corp."
#property link      "http://www.mql5.com"
#property version   "1.00"
#property description "Индикатор демонстрирует как нужно получать данные"
#property description "индикаторных буферов для технического индикатора iWPR."
#property description "Символ и таймфрейм, на котором рассчитывается индикатор,"
#property description "задаются параметрами symbol и period."
#property description "Способ создания хэндла задается параметром 'type' (тип функции"

#property indicator_separate_window
#property indicator_buffers 1
#property indicator_plots   1
//--- построение iWPR
#property indicator_label1  "iWPR"
#property indicator_type1   DRAW_LINE
#property indicator_color1  clrCyan
#property indicator_style1  STYLE_SOLID
#property indicator_width1  1
```

```

//--- зададим граничные значения индикатора
#property indicator_minimum -100
#property indicator_maximum 0
//--- горизонтальные уровни в окне индикатора
#property indicator_level1 -20.0
#property indicator_level2 -80.0
//+-----+
//| перечисление способов создания хэндла |
//+-----+
enum Creation
{
    Call_iWPR,          // использовать iWPR
    Call_IndicatorCreate // использовать IndicatorCreate
};
//--- входные параметры
input Creation      type=Call_iWPR;          // тип функции
input int           calc_period=14;          // период
input string        symbol=" ";              // символ
input ENUM_TIMEFRAMES period=PERIOD_CURRENT; // таймфрейм
//--- индикаторный буфер
double iWPRBuffer[];
//--- переменная для хранения хэндла индикатора iWPR
int handle;
//--- переменная для хранения
string name=symbol;
//--- имя индикатора на графике
string short_name;
//--- будем хранить количество значений в индикаторе Larry Williams' Percent Range
int bars_calculated=0;
//+-----+
//| Custom indicator initialization function |
//+-----+
int OnInit()
{
    //--- привязка массива к индикаторному буферу
    SetIndexBuffer(0,iWPRBuffer,INDICATOR_DATA);
    //--- определимся с символом, на котором строится индикатор
    name=symbol;
    //--- удалим пробелы слева и справа
    StringTrimRight(name);
    StringTrimLeft(name);
    //--- если после этого длина строки name нулевая
    if(StringLen(name)==0)
    {
        //--- возьмем символ с графика, на котором запущен индикатор
        name=_Symbol;
    }
    //--- создадим хэндл индикатора

```

```

if (type==Call_iWPR)
    handle=iWPR(name,period,calc_period);
else
{
    //--- заполним структуру значениями параметров индикатора
    MqlParam pars[1];
    //--- период
    pars[0].type=TYPE_INT;
    pars[0].integer_value=calc_period;
    handle=IndicatorCreate(name,period,IND_WPR,1,pars);
}
//--- если не удалось создать хэндл
if (handle==INVALID_HANDLE)
{
    //--- сообщим о неудаче и выведем номер ошибки
    PrintFormat("Не удалось создать хэндл индикатора iWPR для пары %s/%s, код ошибки",
        name,
        EnumToString(period),
        GetLastError());
    //--- работа индикатора завершается досрочно при возврате отрицательного значения
    return(-1);
}
//--- покажем на какой паре символ/таймфрейм рассчитан индикатор Williams' Percent Range
short_name=StringFormat("iWPR(%s/%s, %d)",name,EnumToString(period),calc_period);
IndicatorSetString(INDICATOR_SHORTNAME,short_name);
//--- нормальное выполнение инициализации индикатора
return(0);
}

//+-----+
//| Custom indicator iteration function |
//+-----+

int OnCalculate(const int rates_total,
                const int prev_calculated,
                const datetime &time[],
                const double &open[],
                const double &high[],
                const double &low[],
                const double &close[],
                const long &tick_volume[],
                const long &volume[],
                const int &spread[])
{
    //--- количество копируемых значений из индикатора iWPR
    int values_to_copy;
    //--- узнаем количество рассчитанных значений в индикаторе
    int calculated=BarsCalculated(handle);
    if (calculated<=0)
    {

```



```

        PrintFormat("BarsCalculated() вернул %d, код ошибки %d",calculated,GetLastError());
        return(0);
    }

    //--- если это первый запуск вычислений нашего индикатора или изменилось количество з
    //--- или если необходимо рассчитать индикатор для двух или более баров (значит что-т
    if(prev_calculated==0 || calculated!=bars_calculated || rates_total>prev_calculate
    {
        //--- если массив iWPRBuffer больше, чем значений в индикаторе iWPR на паре sym
        //--- в противном случае копировать будем меньше, чем размер индикаторных буфер
        if(calculated>rates_total) values_to_copy=rates_total;
        else
            values_to_copy=calculated;
    }
    else
    {
        //--- значит наш индикатор рассчитывается не в первый раз и с момента последнег
        //--- для расчета добавилось не более одного бара
        values_to_copy=(rates_total-prev_calculated)+1;
    }

    //--- заполняем массив значениями из индикатора Williams' Percent Range
    //--- если FillArrayFromBuffer вернула false, значит данные не готовы - завершаем раб
    if(!FillArrayFromBuffer(iWPRBuffer,handle,values_to_copy)) return(0);
    //--- сформируем сообщение
    string comm=StringFormat("%s ==> Обновлено значений в индикаторе %s: %d",
                               TimeToString(TimeCurrent(),TIME_DATE|TIME_SECONDS),
                               short_name,
                               values_to_copy);

    //--- выведем на график служебное сообщение
    Comment(comm);

    //--- запомним количество значений в индикаторе Williams' Percent Range
    bars_calculated=calculated;
    //--- вернем значение prev_calculated для следующего вызова
    return(rates_total);
}

//+-----+
//| Заполняем индикаторный буфер из индикатора iWPR |
//+-----+

bool FillArrayFromBuffer(double &wpr_buffer[], // индикаторный буфер значений Willia
                        int ind_handle,        // хэндл индикатора iWPR
                        int amount             // количество копируемых значений
                        )
{
    //--- сбросим код ошибки
    ResetLastError();
    //--- заполняем часть массива iWPRBuffer значениями из индикаторного буфера под индек
    if(CopyBuffer(ind_handle,0,0,amount,wpr_buffer)<0)
    {
        //--- если копирование не удалось, сообщим код ошибки
        PrintFormat("Не удалось скопировать данные из индикатора iWPR, код ошибки %d",G

```

```
        //--- завершим с нулевым результатом - это означает, что индикатор будет считат
        return(false);
    }
//--- все получилось
    return(true);
}
//+-----+
//| Indicator deinitialization function |
//+-----+
void OnDeinit(const int reason)
{
//--- почистим график при удалении индикатора
    Comment("");
}
```

iVIDyA

函数返回变量指数动态平均数指标处理器。只有一个缓冲区。

```
int iVIDyA(
    string          symbol,          // 交易品种名称
    ENUM_TIMEFRAMES period,          // 周期
    int             cmo_period,      // Chande 动量指标周期
    int             ema_period,      // EMA 平滑周期
    int             ma_shift,        // 价格图表平移
    ENUM_APPLIED_PRICE applied_price // 价格或者处理器类型
);
```

参量

symbol

[in] 证券交易品种名称，数据用来计算指标。 [NULL](#) 值代表当前交易品种。

period

[in] 周期值可以是 [ENUM_TIMEFRAMES](#) 值中的一个，0代表当前时间表。

cmo_period

[in] 钱德动量摆动指标计算周期（计算柱）。

ema_period

[in] 平滑系数计算EMA 周期（计算柱）。

ma_shift

[in] 与价格图表相关的指标变化。

applied_price

[in] 使用价格。可以是任意 [ENUM_APPLIED_PRICE](#) 价格常量或者另外指标处理器。

返回值

返回特殊技术指标处理器，失败返回 [INVALID_HANDLE](#). 计算机内存从不使用的指标中释放，使用指标处理程序传递到的函数 [IndicatorRelease\(\)](#) 。

:

```
//+-----+
//|                                     Demo_iVIDyA.mq5 |
//|                                     Copyright 2011, MetaQuotes Software Corp. |
//|                                     http://www.mql5.com |
//+-----+
#property copyright "Copyright 2011, MetaQuotes Software Corp."
#property link      "http://www.mql5.com"
#property version   "1.00"
#property description "Индикатор демонстрирует как нужно получать данные"
#property description "индикаторных буферов для технического индикатора iVIDyA."
#property description "Символ и таймфрейм, на котором рассчитывается индикатор,"
#property description "задаются параметрами symbol и period."
```

```

#property description "Способ создания хэндла задается параметром 'type' (тип функции
#property description "Все остальные параметры как в стандартном Variable Index Dynam

#property indicator_chart_window
#property indicator_buffers 1
#property indicator_plots 1
//--- построение iVIDyA
#property indicator_label1 "iVIDyA"
#property indicator_type1 DRAW_LINE
#property indicator_color1 clrBlue
#property indicator_style1 STYLE_SOLID
#property indicator_width1 1
//+-----+
//| перечисление способов создания хэндла |
//+-----+
enum Creation
{
    Call_iVIDyA,          // использовать iVIDyA
    Call_IndicatorCreate  // использовать IndicatorCreate
};
//--- входные параметры
input Creation          type=Call_iVIDyA;          // тип функции
input int               cmo_period=15;             // период Chande Momentum
input int               ema_period=12;             // период фактора сглаживания
input int               ma_shift=0;                // смещение
input ENUM_APPLIED_PRICE applied_price=PRICE_CLOSE; // тип цены
input string            symbol=" ";                // символ
input ENUM_TIMEFRAMES   period=PERIOD_CURRENT;    // таймфрейм
//--- индикаторный буфер
double iVIDyABuffer[];
//--- переменная для хранения хэндла индикатора iVIDyA
int handle;
//--- переменная для хранения
string name=symbol;
//--- имя индикатора на графике
string short_name;
//--- будем хранить количество значений в индикаторе Variable Index Dynamic Average
int bars_calculated=0;
//+-----+
//| Custom indicator initialization function |
//+-----+
int OnInit()
{
    //--- привязка массива к индикаторному буферу
    SetIndexBuffer(0,iVIDyABuffer,INDICATOR_DATA);
    //--- зададим смещение
    PlotIndexSetInteger(0,PLOT_SHIFT,ma_shift);
    //--- определимся с символом, на котором строится индикатор

```

```

    name=symbol;
    //--- удалим пробелы слева и справа
    StringTrimRight(name);
    StringTrimLeft(name);
    //--- если после этого длина строки name нулевая
    if(StringLen(name)==0)
    {
        //--- возьмем символ с графика, на котором запущен индикатор
        name=_Symbol;
    }
    //--- создадим хэндл индикатора
    if(type==Call_iVIDyA)
        handle=iVIDyA(name,period,cmo_period,ema_period,ma_shift,applied_price);
    else
    {
        //--- заполним структуру значениями параметров индикатора
        MqlParam pars[4];
        //--- период Chande Momentum
        pars[0].type=TYPE_INT;
        pars[0].integer_value=cmo_period;
        //--- период фактора сглаживания
        pars[1].type=TYPE_INT;
        pars[1].integer_value=ema_period;
        //--- смещение
        pars[2].type=TYPE_INT;
        pars[2].integer_value=ma_shift;
        //--- тип цены
        pars[3].type=TYPE_INT;
        pars[3].integer_value=applied_price;
        handle=IndicatorCreate(name,period,IND_VIDYA,4,pars);
    }
    //--- если не удалось создать хэндл
    if(handle==INVALID_HANDLE)
    {
        //--- сообщим о неудаче и выведем номер ошибки
        PrintFormat("Не удалось создать хэндл индикатора iVIDyA для пары %s/%s, код оши
            name,
            EnumToString(period),
            GetLastError());
        //--- работа индикатора завершается досрочно при возврате отрицательного значен
        return(-1);
    }
    //--- покажем на какой паре символ/таймфрейм рассчитан индикатор Triple Exponential M
    short_name=StringFormat("iVIDyA(%s/%s, %d, %d, %d, %s)",name,EnumToString(period),
        cmo_period,ema_period,ma_shift,EnumToString(applied_price)
        IndicatorSetString(INDICATOR_SHORTNAME,short_name);
    //--- нормальное выполнение инициализации индикатора
    return(0);

```

```

    }
//+-----+
//| Custom indicator iteration function |
//+-----+
int OnCalculate(const int rates_total,
               const int prev_calculated,
               const datetime &time[],
               const double &open[],
               const double &high[],
               const double &low[],
               const double &close[],
               const long &tick_volume[],
               const long &volume[],
               const int &spread[])
{
//--- количество копируемых значений из индикатора iVIDyA
    int values_to_copy;
//--- узнаем количество рассчитанных значений в индикаторе
    int calculated=BarsCalculated(handle);
    if(calculated<=0)
    {
        PrintFormat("BarsCalculated() вернул %d, код ошибки %d",calculated,GetLastError());
        return(0);
    }
//--- если это первый запуск вычислений нашего индикатора или изменилось количество э
//--- или если необходимо рассчитать индикатор для двух или более баров (значит что-т
    if(prev_calculated==0 || calculated!=bars_calculated || rates_total>prev_calculate
    {
        //--- если массив iWPRBuffer больше, чем значений в индикаторе iVIDyA на паре s
        //--- в противном случае копировать будем меньше, чем размер индикаторных буфер
        if(calculated>rates_total) values_to_copy=rates_total;
        else
            values_to_copy=calculated;
    }
    else
    {
        //--- значит наш индикатор рассчитывается не в первый раз и с момента последнег
        //--- для расчета добавилось не более одного бара
        values_to_copy=(rates_total-prev_calculated)+1;
    }
//--- заполняем массив значениями из индикатора Variable Index Dynamic Average
//--- если FillArrayFromBuffer вернула false, значит данные не готовы - завершаем раб
    if(!FillArrayFromBuffer(iVIDyABuffer,ma_shift,handle,values_to_copy)) return(0);
//--- сформируем сообщение
    string comm=StringFormat("%s ==> Обновлено значений в индикаторе %s: %d",
                             TimeToString(TimeCurrent(),TIME_DATE|TIME_SECONDS),
                             short_name,
                             values_to_copy);
//--- выведем на график служебное сообщение

```

```

    Comment(comm);
//--- заппомним количество значений в индикаторе Variable Index Dynamic Average
    bars_calculated=calculated;
//--- вернем значение prev_calculated для следующего вызова
    return(rates_total);
}
//+-----+
//| Заполняем индикаторный буфер из индикатора iVIDyA |
//+-----+
bool FillArrayFromBuffer(double &vidya_buffer[], // индикаторный буфер значений Variab
                        int v_shift,           // смещение линии
                        int ind_handle,        // хэндл индикатора iVIDyA
                        int amount            // количество копируемых значений
                        )
{
//--- сбросим код ошибки
    ResetLastError();
//--- заполняем часть массива iVIDyABuffer значениями из индикаторного буфера под инд
    if(CopyBuffer(ind_handle,0,-v_shift,amount,vidya_buffer)<0)
    {
        //--- если копирование не удалось, сообщим код ошибки
        PrintFormat("Не удалось скопировать данные из индикатора iVIDyA, код ошибки %d"
        //--- завершим с нулевым результатом - это означает, что индикатор будет считат
        return(false);
    }
//--- все получилось
    return(true);
}
//+-----+
//| Indicator deinitialization function |
//+-----+
void OnDeinit(const int reason)
{
//--- почистим график при удалении индикатора
    Comment("");
}

```

iVolumes

函数返回交易量指标处理器。只有一个缓冲区。

```
int iVolumes(
    string          symbol,          // 交易品种名称
    ENUM_TIMEFRAMES period,          // 周期
    ENUM_APPLIED_VOLUME applied_volume // 计算的交易量类型
)
```

参量

symbol

[in] 证券交易品种名称，数据用来计算指标。 [NULL](#) 值代表当前交易品种。

[in] 周期值可以是 [ENUM_TIMEFRAMES](#) 值中的一个，0代表当前时间表。

applied_volume

[in] 使用的交易量。可以是 [ENUM_APPLIED_VOLUME](#) 的任意值。

返回值

返回特殊技术指标处理器，失败返回 [INVALID_HANDLE](#)。计算机内存从不使用的指标中释放，使用指标处理程序传递到的函数 [IndicatorRelease\(\)](#)。

:

```
//+-----+
//|                                     Demo_iVolumes.mq5 |
//|                                     Copyright 2011, MetaQuotes Software Corp. |
//|                                     http://www.mql5.com |
//+-----+

#property copyright "Copyright 2011, MetaQuotes Software Corp."
#property link      "http://www.mql5.com"
#property version   "1.00"
#property description "Индикатор демонстрирует как нужно получать данные"
#property description "индикаторных буферов для технического индикатора iVolumes."
#property description "Символ и таймфрейм, на котором рассчитывается индикатор,"
#property description "задаются параметрами symbol и period."
#property description "Способ создания хэндла задается параметром 'type' (тип функции"

#property indicator_separate_window
#property indicator_buffers 2
#property indicator_plots 1
//--- построение iVolumes
#property indicator_label1 "iVolumes"
#property indicator_type1  DRAW_COLOR_HISTOGRAM
#property indicator_color1 clrGreen, clrRed
#property indicator_style1 STYLE_SOLID
#property indicator_width1 1
//+-----+
```



```

//| перечисление способов создания хэндла |
//+-----+
enum Creation
{
    Call_iVolumes,          // использовать iVolumes
    Call_IndicatorCreate    // использовать IndicatorCreate
};
//--- входные параметры
input Creation             type=Call_iVolumes;          // тип функции
input ENUM_APPLIED_VOLUME applied_volume=VOLUME_TICK;   // тип объема
input string               symbol=" ";                  // символ
input ENUM_TIMEFRAMES      period=PERIOD_CURRENT;       // таймфрейм
//--- индикаторные буферы
double iVolumesBuffer[];
double iVolumesColors[];
//--- переменная для хранения хэндла индикатора iVolumes
int handle;
//--- переменная для хранения
string name=symbol;
//--- имя индикатора на графике
string short_name;
//--- будем хранить количество значений в индикаторе Volumes
int bars_calculated=0;
//+-----+
//| Custom indicator initialization function |
//+-----+
int OnInit()
{
    //--- привязка массива к индикаторным буферам
    SetIndexBuffer(0,iVolumesBuffer,INDICATOR_DATA);
    SetIndexBuffer(1,iVolumesColors,INDICATOR_COLOR_INDEX);
    //--- определимся с символом, на котором строится индикатор
    name=symbol;
    //--- удалим пробелы слева и справа
    StringTrimRight(name);
    StringTrimLeft(name);
    //--- если после этого длина строки name нулевая
    if(StringLen(name)==0)
    {
        //--- возьмем символ с графика, на котором запущен индикатор
        name=_Symbol;
    }
    //--- создадим хэндл индикатора
    if(type==Call_iVolumes)
        handle=iVolumes(name,period,applied_volume);
    else
    {
        //--- заполним структуру значениями параметров индикатора

```

```

MqlParam pars[1];
//--- тип цены
pars[0].type=TYPE_INT;
pars[0].integer_value=applied_volume;
handle=IndicatorCreate(name,period,IND_VOLUMES,1,pars);
}
//--- если не удалось создать хэндл
if(handle==INVALID_HANDLE)
{
    //--- сообщим о неудаче и выведем номер ошибки
    PrintFormat("Не удалось создать хэндл индикатора iVolumes для пары %s/%s, код о
                name,
                EnumToString(period),
                GetLastError());
    //--- работа индикатора завершается досрочно при возврате отрицательного значен
    return(-1);
}
//--- покажем на какой паре символ/таймфрейм рассчитан индикатор Volumes
short_name=StringFormat("iVolumes(%s/%s, %s)",name,EnumToString(period),EnumToStri
IndicatorSetString(INDICATOR_SHORTNAME,short_name);
//--- нормальное выполнение инициализации индикатора
return(0);
}
//+-----+
//| Custom indicator iteration function |
//+-----+
int OnCalculate(const int rates_total,
                const int prev_calculated,
                const datetime &time[],
                const double &open[],
                const double &high[],
                const double &low[],
                const double &close[],
                const long &tick_volume[],
                const long &volume[],
                const int &spread[])
{
    //--- количество копируемых значений из индикатора iVolumes
    int values_to_copy;
    //--- узнаем количество рассчитанных значений в индикаторе
    int calculated=BarsCalculated(handle);
    if(calculated<=0)
    {
        PrintFormat("BarsCalculated() вернул %d, код ошибки %d",calculated,GetLastError
        return(0);
    }
    //--- если это первый запуск вычислений нашего индикатора или изменилось количество з
    //--- или если необходимо рассчитать индикатор для двух или более баров (значит что-т

```

```

    if(prev_calculated==0 || calculated!=bars_calculated || rates_total>prev_calculated)
    {
        //--- если массив iVolumesBuffer больше, чем значений в индикаторе iVolumes на
        //--- в противном случае копировать будем меньше, чем размер индикаторных буфер
        if(calculated>rates_total) values_to_copy=rates_total;
        else
            values_to_copy=calculated;
    }
    else
    {
        //--- значит наш индикатор рассчитывается не в первый раз и с момента последнег
        //--- для расчета добавилось не более одного бара
        values_to_copy=(rates_total-prev_calculated)+1;
    }
    //--- заполняем массивы значениями из индикатора iVolumes
    //--- если FillArraysFromBuffers вернула false, значит данные не готовы - завершаем р
    if(!FillArraysFromBuffers(iVolumesBuffer,iVolumesColors,handle,values_to_copy)) re
    //--- сформируем сообщение
    string comm=StringFormat("%s ==> Обновлено значений в индикаторе %s: %d",
                               TimeToString(TimeCurrent(),TIME_DATE|TIME_SECONDS),
                               short_name,
                               values_to_copy);
    //--- выведем на график служебное сообщение
    Comment(comm);
    //--- запомним количество значений в индикаторе Volumes
    bars_calculated=calculated;
    //--- вернем значение prev_calculated для следующего вызова
    return(rates_total);
}

//+-----+
//| Заполняем индикаторные буферы из индикатора iVolumes |
//+-----+
bool FillArraysFromBuffers(double &volume_buffer[], // индикаторный буфер значений
                           double &color_buffer[], // индикаторный буфер цветов
                           int ind_handle, // хэндл индикатора iVolumes
                           int amount // количество копируемых значен

)

{
    //--- сбросим код ошибки
    ResetLastError();
    //--- заполняем часть массива iVolumesBuffer значениями из индикаторного буфера под и
    if(CopyBuffer(ind_handle,0,0,amount,volume_buffer)<0)
    {
        //--- если копирование не удалось, сообщим код ошибки
        PrintFormat("Не удалось скопировать данные из индикатора iVolumes, код ошибки %
        //--- завершим с нулевым результатом - это означает, что индикатор будет считат
        return(false);
    }
    //--- заполняем часть массива iVolumesColors значениями из индикаторного буфера под и

```

```
if(CopyBuffer(ind_handle,1,0,amount,color_buffer)<0)
{
    //--- если копирование не удалось, сообщим код ошибки
    PrintFormat("Не удалось скопировать данные из индикатора iVolumes, код ошибки %d", GetLastError());
    //--- завершим с нулевым результатом - это означает, что индикатор будет считаться неинициализированным
    return(false);
}
//--- все получилось
return(true);
}

//+-----+
//| Indicator deinitialization function |
//+-----+

void OnDeinit(const int reason)
{
    //--- почистим график при удалении индикатора
    Comment("");
}
```

事件函数

该组包括自定义事件函数和定时器事件函数。除了该组函数外，有处理[预定义事件](#)特别函数。

函数	功能
EventSetTimer	用当前图表指定周期启动定时器事件生成器
EventKillTimer	依据当前图表定时器停止事件生成器
EventChartCustom	为指定图表生成自定义事件

另见

[图表事件类型](#)

EventSetTimer

该函数应用于客户端，[定时器](#)事件必须依据EA交易依附的图表指定周期生成。

```
bool EventSetTimer(  
    int seconds // 秒数  
);
```

参量

seconds

[in] 决定定时器事件发生频率的秒数。

返回值

若成功返回true，否则false。若要获得[错误代码](#)，调用函数[GetLastError\(\)](#)。

注释

一般情况下，这个函数必须从函数[OnInit\(\)](#) 或者从类[构造函数](#)中调用。若为处理定时器事件，EA交易必须有[OnTimer\(\)](#) 函数。

每个EA交易及标识符使用自己的定时器工作，且仅从那里接收事件。一旦mql5程序停止运行，如果定时器不能通过[EventKillTimer\(\)](#) 函数创建和禁止，则会被强制销毁。

对于每一个程序只能运行一个定时器。

EventKillTimer

指定停止生成[定时器](#)事件的客户端。

```
void EventKillTimer();
```

返回值

无返回值

注释

特别是，如果函数[OnInit\(\)](#)从函数[EventSetTimer\(\)](#)调用，该函数必须从函数[OnDeinit\(\)](#)调用。如果函数[EventSetTimer\(\)](#)在类构造函数中调用，那么该函数也可以从类析构函数调用。

每个EA交易及标识符使用自己的定时器工作，且仅从那里接收事件。一旦mql5程序停止运行，如果定时器不能通过[EventKillTimer\(\)](#)函数创建和禁止，则会被强制销毁。

EventChartCustom

该函数为指定图表生成自定义事件。

```
bool EventChartCustom(
    long    chart_id,           // 接收图表事件的标识符
    ushort  custom_event_id,    // 事件标识符
    long    lparam,            // 长整型参量
    double  dparam,            // 双精度型参量
    string  sparam              // 事件的字符串参量
);
```

参量

- chart_id*
[in] 图表标识符。0 意味着当前图表。
- custom_event_id*
[in] 用户事件ID。这个标识符自动添加到CHARTEVENT_CUSTOM值中且转化成整数型。
- lparam*
[in] 传递到OnChartEvent函数的长整型事件参量。
- dparam*
[in] 传递到OnChartEvent函数的双精度型事件参量。
- sparam*
[in] 传递到OnChartEvent函数的字符串型事件参量。如果字符串超过63个字符，则会被删节。

返回值

true

false,
[GetLastError\(\)](#).

注释

附在指定图表的EA交易或者指标处理使用OnChartEvent函数 (int event_ id , long& lparam , double& dparam , string& sparam) 的事件。

对于每一种事件类型，OnChartEvent() 函数的输入参量有事件需要过程的定值。通过这个参量传递的事件和值列在以下表格中。

事件	id参量值	lparam 参量值	dparam 参量值	sparam 参量值
击键事件	CHARTEVENT_KEYDOWN	击键代码	—	—
图解物件创建事件 (CHARTEVENT_OBJECT_	—	—	创建的图解物件名称

CHART_EVENT_OBJECT_CREATE=true)	CREATE			
通过属性对话框改变物件属性事件	CHARTEVENT_OBJECT_CHANGE	—	—	更改的图解物件名称
图解物件删除事件 (CHART_EVENT_OBJECT_DELETE=true)	CHARTEVENT_OBJECT_DELETE	-	-	删除的图解物件名称
图表上鼠标点击事件	CHARTEVENT_CLICK	X 坐标	Y 坐标	—
附属于图表的图解物件鼠标点击事件	CHARTEVENT_OBJECT_CLICK	X 坐标	Y 坐标	事件发生的图解物件名称
用鼠标拖拽图解物件事件	CHARTEVENT_OBJECT_DRAG	—	—	移动的图解物件名称
LabelEdit图解物件输入框中完成文本编辑事件	CHARTEVENT_OBJECT_ENDEDIT	—	—	完成文本编辑的LabelEdit图解物件名称
	CHARTEVENT_CHART_CHANGE	—	—	—
N号码下用户ID事件	CHARTEVENT_CUSTOM+N	通过 EventChartCustom() 函数设置的值	通过 EventChartCustom() 函数设置的值	通过 EventChartCustom() 函数设置的值

示例：

```
//+-----+
//|                                     ButtonClickExpert.mq5 |
//|                                     Copyright 2009, MetaQuotes Software Corp. |
//|                                     http://www.mql5.com |
//+-----+
#property copyright "2009, MetaQuotes Software Corp."
#property link      "http://www.mql5.com"
#property version   "1.00"

string buttonID="Button";
```

```

string labelID="Info";
int broadcastEventID=5000;
//+-----+
//|  专家初始化函数          |
//+-----+
int OnInit()
{
//--- 创建按钮发送自定义事件
    ObjectCreate(0,buttonID,OBJ_BUTTON,0,100,100);
    ObjectSetInteger(0,buttonID,OBJPROP_COLOR,clrWhite);
    ObjectSetInteger(0,buttonID,OBJPROP_BGCOLOR,clrGray);
    ObjectSetInteger(0,buttonID,OBJPROP_XDISTANCE,100);
    ObjectSetInteger(0,buttonID,OBJPROP_YDISTANCE,100);
    ObjectSetInteger(0,buttonID,OBJPROP_XSIZE,200);
    ObjectSetInteger(0,buttonID,OBJPROP_YSIZE,50);
    ObjectSetString(0,buttonID,OBJPROP_FONT,"Arial");
    ObjectSetString(0,buttonID,OBJPROP_TEXT,"Button");
    ObjectSetInteger(0,buttonID,OBJPROP_FONTSIZE,10);
    ObjectSetInteger(0,buttonID,OBJPROP_SELECTABLE,0);

//--- 创建标签展示信息
    ObjectCreate(0,labelID,OBJ_LABEL,0,100,100);
    ObjectSetInteger(0,labelID,OBJPROP_COLOR,clrRed);
    ObjectSetInteger(0,labelID,OBJPROP_XDISTANCE,100);
    ObjectSetInteger(0,labelID,OBJPROP_YDISTANCE,50);
    ObjectSetString(0,labelID,OBJPROP_FONT,"Trebuchet MS");
    ObjectSetString(0,labelID,OBJPROP_TEXT,"No information");
    ObjectSetInteger(0,labelID,OBJPROP_FONTSIZE,20);
    ObjectSetInteger(0,labelID,OBJPROP_SELECTABLE,0);

//---
    return(0);
}
//+-----+
//|  专家无法初始化函数          |
//+-----+
void OnDeinit(const int reason)
{
//---
    ObjectDelete(0,buttonID);
    ObjectDelete(0,labelID);
}
//+-----+
//|  专家订单号函数          |
//+-----+
void OnTick()
{
//---

```

```

    }
//+-----+
void OnChartEvent(const int id,
                  const long &lparam,
                  const double &dparam,
                  const string &sparam)
{
//--- 通过按下鼠标键检测事件
    if(id==CHARTEVENT_OBJECT_CLICK)
    {
        string clickedChartObject=sparam;
        //--- 如果点击带有按钮ID名的物件
        if(clickedChartObject==buttonID)
        {
            //--- 按钮状态-按或者不按
            bool selected=ObjectGetInteger(0,buttonID,OBJPROP_STATE);
            //--- 调试信息日志
            Print("Button pressed = ",selected);
            int customEventID; // 发送的自定义事件数
            string message;    // 事件中被发送的信息
            //--- 如果按下按钮
            if(selected)
            {
                message="Button pressed";
                customEventID=CHARTEVENT_CUSTOM+1;
            }
            else // 不按按钮
            {
                message="Button in not pressed";
                customEventID=CHARTEVENT_CUSTOM+999;
            }
            //--- 发送自定义事件 "our"图表
            EventChartCustom(0,customEventID-CHARTEVENT_CUSTOM,0,0,message);
            ///--- 发送信息到所有打开的图表
            BroadcastEvent(ChartID(),0,"Broadcast Message");
            //--- 调试信息
            Print("Sent an event with ID = ",customEventID);
        }
        ChartRedraw(); // 强制重画所有图表物件
    }

//--- 检测属于用户事件的事件
    if(id>CHARTEVENT_CUSTOM)
    {
        if(id==broadcastEventID)
        {
            Print("Got broadcast message from a chart with id = "+lparam);
        }
    }
}

```

```

    }
    else
    {
        ///--- 阅读事件中的文本信息
        string info=sparam;
        Print("Handle the user event with the ID = ",id);
        ///--- 标签中展示信息
        ObjectSetString(0,labelID,OBJPROP_TEXT,sparam);
        ChartRedraw();// 强制重画所有图表物件
    }
}

//+-----+
//| 发送广播事件到所有打开的图表 |
//+-----+
void BroadcastEvent(long lparam,double dparam,string sparam)
{
    int eventID=broadcastEventID-CHARTEVENT_CUSTOM;
    long currChart=ChartFirst();
    int i=0;
    while(i<CHARTS_MAX // 打开图表不能多于CHARTS_MAX
    {
        EventChartCustom(currChart,eventID,lparam,dparam,sparam);
        currChart=ChartNext(currChart); // 从上一个接收新图表
        if(currChart==0) break; // 达到图表列表末端
        i++; // 不要忘记增加计数器
    }
}

//+-----+

```

另见

[客户端事件](#) , [事件处理程序函数](#)

Standard Library

This group of chapters contains the technical details of the MQL5 Standard Library and descriptions of all its key components.

MQL5 Standard Library is written in MQL5 and is designed to facilitate writing programs (indicators, scripts, experts) to end users. Library provides convenient access to most of the internal functions MQL5.

MQL5 Standard Library is placed in the working directory of the terminal in the 'Include' folder.

Section	Location
Base class	Include\
Classes of data	Include\Arrays\
Classes for file operations	Include\Files\
Classes for string operations	Include\Strings\
Classes for graphic objects	Include\Objects\
Class for working with chart	Include\Charts\
Technical indicators	Include\Indicators\
Trade Classes	Include\Trade\
Trading Strategy Classes	Include\Expert\

Basic Class CObject

Class CObject is the base class for constructing a MQL5 Standard Library .

Description

Class CObject provides all its descendants to be part of a linked list. Also identifies a number of virtual methods for further implementation in descendant classes.

Declaration

```
class CObject
```

Title

```
#include <Object.mqh>
```

Class Methods

Attributes	
Prev	Gets the value of the previous item
Prev	Sets the value of the previous item
Next	Gets the value of the subsequent element
Next	Sets the next element
Compare methods	
virtual Compare	Returns the result of comparison with another object
Input/output	
virtual Save	Writes object to a file
virtual Load	Reads the object from the file
virtual Type	Returns the type of object

Derived classes:

- [CArray](#)
- [CChartObject](#)
- [CChart](#)
- [CString](#)
- [CFile](#)
- [CList](#)
- [CTreeNode](#)

Prev

Gets a pointer to the previous list item.

```
CObject* Prev()
```

Return Value

Pointer to the previous list item. If an item is listed first, then return NULL.

Example:

```
//--- example for CObject::Prev()
#include <Object.mqh>
//---
void OnStart()
{
    CObject *object_first,*object_second;
    //---
    object_first=new CObject;
    if(object_first==NULL)
    {
        printf("Object create error");
        return;
    }
    object_second=new CObject;
    if(object_second==NULL)
    {
        printf("Object create error");
        delete object_first;
        return;
    }
    //--- set interconnect
    object_first.Next(object_second);
    object_second.Prev(object_first);
    //--- use prev object
    CObject *object=object_second.Prev();
    //--- delete objects
    delete object_first;
    delete object_second;
}
```

Prev

Sets the pointer to the previous list item.

```
void Prev(  
    CObject* object    // Pointer to the previous list item  
)
```

Parameters

object

[in] New value pointer to the previous list item.

Example:

```
//--- example for CObject::Prev(CObject*)  
#include <Object.mqh>  
//---  
void OnStart()  
{  
    CObject *object_first,*object_second;  
    //---  
    object_first=new CObject;  
    if(object_first==NULL)  
    {  
        printf("Object create error");  
        return;  
    }  
    object_second=new CObject;  
    if(object_second==NULL)  
    {  
        printf("Object create error");  
        delete object_first;  
        return;  
    }  
    //--- set interconnect  
    object_first.Next(object_second);  
    object_second.Prev(object_first);  
    //--- use objects  
    //--- ...  
    //--- delete objects  
    delete object_first;  
    delete object_second;  
}
```


Next

Gets a pointer to the next element of the list.

```
CObject* Next()
```

Return Value

Pointer to the next item in the list. If the last item in the list, returns NULL.

Example:

```
//--- example for CObject::Next()
#include <Object.mqh>
//---
void OnStart()
{
    CObject *object_first,*object_second;
    //---
    object_first=new CObject;
    if(object_first==NULL)
    {
        printf("Object create error");
        return;
    }
    object_second=new CObject;
    if(object_second==NULL)
    {
        printf("Object create error");
        delete object_first;
        return;
    }
    //--- set interconnect
    object_first.Next(object_second);
    object_second.Prev(object_first);
    //--- use next object
    CObject *object=object_first.Next();
    //--- delete objects
    delete object_first;
    delete object_second;
}
```

Next

Sets the pointer to the next element of the list.

```
void Next(  
    CObject* object    // Pointer to the next list item  
)
```

Parameters

object

[in] New value pointer to the next list item.

Example:

```
//--- example for CObject::Next(CObject*)  
#include <Object.mqh>  
//---  
void OnStart()  
{  
    CObject *object_first,*object_second;  
    //---  
    object_first=new CObject;  
    if(object_first==NULL)  
    {  
        printf("Object create error");  
        return;  
    }  
    object_second=new CObject;  
    if(object_second==NULL)  
    {  
        printf("Object create error");  
        delete object_first;  
        return;  
    }  
    //--- set interconnect  
    object_first.Next(object_second);  
    object_second.Prev(object_first);  
    //--- use objects  
    //--- ...  
    //--- delete objects  
    delete object_first;  
    delete object_second;  
}
```

Compare

Compares the data item in the list with data on another element of the list.

```
virtual int Compare(
    CObject const * node,      // Node to compare with
    int mode=0                // Compare mode
) const
```

Parameters

node

[in] Pointer to a list item to compare

mode=0

[in] Variant Comparison

Return Value

0 - in case the list items are equal, -1 - if the list item is less than the item in the list for comparison (*node*) , 1 - if the list item more than item in the list for comparison (*node*) .

Note

Method Compare () in class CObject always returns 0 and does not perform any action. If you want to compare data derived class, the method Compare (...) Should be implemented. The mode parameter should be used when implementing multivariate comparison.

Example:

```
//--- example for CObject::Compare(...)
#include <Object.mqh>
//---
void OnStart()
{
    CObject *object_first,*object_second;
    //---
    object_first=new CObject;
    if(object_first==NULL)
    {
        printf("Object create error");
        return;
    }
    object_second=new CObject;
    if(object_second==NULL)
    {
        printf("Object create error");
        delete object_first;
        return;
    }
    //--- set interconnect
    object_first.Next(object_second);
```

```
object_second.Prev(object_first);  
//--- compare objects  
int result=object_first.Compare(object_second);  
//--- delete objects  
delete object_first;  
delete object_second;  
}
```

Save

Saves data element list in the file.

```
virtual bool Save(  
    int file_handle    // File handle  
)
```

Parameters

file_handle

[in] Handle to open earlier, with the function `FileOpen ()`, binary

Return Value

true - if successfully completed, false - if an error.

Note

Method `Save (int)` in class `CObject` always returns true and does not perform any action. If you want to save the data derived class in the file, the method `Save (int)` should be implemented.

Example:

```
//--- example for CObject::Save(int)  
#include <Object.mqh>  
//---  
void OnStart()  
{  
    int file_handle;  
    CObject *object=new CObject;  
    //---  
    if(object!=NULL)  
    {  
        printf("Object create error");  
        return;  
    }  
    //--- set objects data  
    //--- . . .  
    //--- open file  
    file_handle=FileOpen("MyFile.bin",FILE_WRITE|FILE_BIN|FILE_ANSI);  
    if(file_handle>=0)  
    {  
        if(!object.Save(file_handle))  
        {  
            //--- file save error  
            printf("File save: Error %d!",GetLastError());  
            delete object;  
            FileClose(file_handle);  
            //---  
            return;  
        }  
    }  
}
```

```
    }  
    FileClose(file_handle);  
}  
delete object;  
}
```

Load

Loads data item in the list from a file.

```
virtual bool Load(  
    int file_handle    // handle to file  
)
```

Parameters

file_handle

[in] Handle to open earlier, with the function `FileOpen ()`, binary

Return Value

true - if successfully completed, false - if an error.

Note

Method `Load (int)` in class `CObject` always returns true and does not perform any action. If you want to load the data derived class from a file, the method `Load (int)` should be implemented.

Example:

```
//--- example for CObject::Load(int)  
#include <Object.mqh>  
//---  
void OnStart()  
{  
    int file_handle;  
    CObject *object=new CObject;  
    //---  
    if(object!=NULL)  
    {  
        printf("Object create error");  
        return;  
    }  
    //--- open file  
    file_handle=FileOpen("MyFile.bin",FILE_READ|FILE_BIN|FILE_ANSI);  
    if(file_handle>=0)  
    {  
        if(!object.Load(file_handle))  
        {  
            //--- file load error  
            printf("File load: Error %d!",GetLastError());  
            delete object;  
            FileClose(file_handle);  
            //---  
            return;  
        }  
        FileClose(file_handle);  
    }
```

```
    }  
    //--- use object  
    //--- . . .  
    delete object;  
}
```


Type

Gets the type identifier.

```
virtual int Type() const
```

Return Value

Type identifier (for CObject - 0) .

Example:

```
//--- example for CObject::Type()
#include <Object.mqh>
//---
void OnStart()
{
    CObject *object=new CObject;
    //---
    object=new CObject;
    if(object ==NULL)
    {
        printf("Object create error");
        return;
    }
    //--- get objects type
    int type=object.Type();
    //--- delete object
    delete object;
}
```

Data Structures

This section contains the technical details of working with various data structures (arrays, linked lists, etc.) and description of the relevant components of the MQL5 Standard Library .

Using classes of data structures, will save time when creating custom data stores of various formats (including composite data structures) .

MQL5 Standard Library (in terms of data sets) is placed in the working directory of the terminal in the Include\Arrays folder.

Data Arrays

Use of classes of dynamic arrays of data will save time when creating a custom data stores of various formats (including multidimensional arrays) .

MQL5 Standard Library (in terms of arrays of data) is placed in the working directory of the terminal in the Include\Arrays folder.

Class	Description
Base class of dynamic data array CArray	Base class of dynamic data array
CArrayChar	Dynamic array of variables of type char or uchar
CArrayShort	Dynamic array of variables of type short or ushort
CArrayInt	Dynamic array of variables of type int or uint
CArrayLong	Dynamic array of variables of type long or ulong
CArrayFloat	Dynamic array of variables of type float
CArrayDouble	Dynamic array of variables of type double
CArrayString	Dynamic array of variables of type string
Base class of object array CArrayObj	Dynamic array of pointers CObject
Base class of list CList	Provides the ability to work with a list of instances of CObject and its descendant
CTreeNode	Provides the ability to work with nodes of the binary tree CTree
CTree	Provides the ability to work with the binary tree of the CTreeNode class instances and its descendants

CArray

CArray class is the base class a dynamic array of variables.

Description

Class CArray provides the ability to work with a dynamic array of variables in the part of the attributes of management of memory allocation, sorting, and working with files.

Declaration

```
class CArray : public CObject
```

Title

```
#include <Arrays\Array.mqh>
```

Class Methods

Attributes	
Step	Gets the step increment size of the array
Step	Set the increment size of the array
Total	Gets the number of elements in the array
Available	Gets the number of free elements of the array are available without additional memory allocation
Max	Gets the maximum possible size of the array without memory reallocation
IsSorted	Gets sign sorted array to the specified option
SortMode	Gets the version of the sorting array
Clear methods	
Clear	Deletes all of the array elements without memory release
Sort methods	
Sort	Sorts an array to the specified option
Input/output	
virtual Save	Saves data array in the file
virtual Load	Loads data array from a file

Derived classes:

- [CArrayChar](#)
- [CArrayShort](#)
- [CArrayInt](#)

- [CArrayLong](#)
- [CArrayFloat](#)
- [CArrayDouble](#)
- [CArrayString](#)
- [CArrayObj](#)

Step

Gets the step increment size of the array.

```
int Step() const
```

Return Value

Increment size of the array.

Example:

```
//--- example for CArray::Step()
#include <Arrays\Array.mqh>
//---
void OnStart()
{
    CArray *array=new CArray;
    //---
    if(array==NULL)
    {
        printf("Object create error");
        return;
    }
    //--- get resize step
    int step=array.Step();
    //--- use array
    //--- ...
    //--- delete array
    delete array;
}
```

Step

Sets the increment size of the array.

```
bool Step(  
    int step    // step  
)
```

Parameters

step

[in] The new value of step increments in the size of the array.

Return Value

true if successful, false - if there was an attempt to establish a step less than or equal to zero.

Example:

```
//--- example for CArray::Step(int)  
#include <Arrays\Array.mqh>  
//---  
void OnStart()  
{  
    CArray *array=new CArray;  
    //---  
    if(array==NULL)  
    {  
        printf("Object create error");  
        return;  
    }  
    //--- set resize step  
    bool result=array.Step(1024);  
    //--- use array  
    //--- ...  
    //--- delete array  
    delete array;  
}
```

Total

Gets the number of elements in the array.

```
int Total() const;
```

Return Value

Number of elements in the array.

Example:

```
//--- example for CArray::Total()
#include <Arrays\Array.mqh>
//---
void OnStart()
{
    CArray *array=new CArray;
    //---
    if(array==NULL)
    {
        printf("Object create error");
        return;
    }
    //--- check total
    int total=array.Total();
    //--- use array
    //--- ...
    //--- delete array
    delete array;
}
```

Available

Gets the number of free elements of the array are available without additional memory allocation.

```
int Available() const
```

Return Value

Number of free elements of the array are available without additional memory allocation.

Example:

```
//--- example for CArray::Available()
#include <Arrays\Array.mqh>
//---
void OnStart()
{
    CArray *array=new CArray;
    //---
    if(array==NULL)
    {
        printf("Object create error");
        return;
    }
    //--- check available
    int available=array.Available();
    //--- use array
    //--- ...
    //--- delete array
    delete array;
}
```


Max

Gets the maximum possible size of the array without memory reallocation.

```
int Max() const
```

Return Value

The maximum possible size of the array without reallocation memory.

Example:

```
//--- example for CArray::Max()
#include <Arrays\Array.mqh>
//---
void OnStart()
{
    CArray *array=new CArray;
    //---
    if(array==NULL)
    {
        printf("Object create error");
        return;
    }
    //--- check maximum size
    int max=array.Max();
    //--- use array
    //--- ...
    //--- delete array
    delete array;
}
```

IsSorted

Gets sign sorted array to the specified option.

```
bool IsSorted(  
    int mode=0      // Sorting mode  
) const
```

Parameters

mode=0

[in] Tested version of the sort.

Return Value

Flag of the sorted list. If the list is sorted by the specified version - true, otherwise - false.

Note

Symptom sorted array can not be changed directly. Symptom sorted set method Sort () and reset any methods to add / insert except InsertSort (...) .

Example:

```
//--- example for CArray::IsSorted()  
#include <Arrays\Array.mqh>  
//---  
void OnStart()  
{  
    CArray *array=new CArray;  
    //---  
    if(array==NULL)  
    {  
        printf("Object create error");  
        return;  
    }  
    //--- check sorted  
    if(array.IsSorted())  
    {  
        //--- use methods for sorted array  
        //--- ...  
    }  
    //--- delete array  
    delete array;  
}
```

SortMode

Gets the version of the sorting array.

```
int SortMode() const;
```

Return Value

Sorting mode.

Example:

```
//--- example for CArray::SortMode()
#include <Arrays\Array.mqh>
//---
void OnStart()
{
    CArray *array=new CArray;
    //---
    if(array==NULL)
    {
        printf("Object create error");
        return;
    }
    //--- check sort mode
    int sort_mode=array.SortMode();
    //--- use array
    //--- ...
    //--- delete array
    delete array;
}
```

Clear

Deletes all of the array elements without memory release.

```
void Clear()
```

Return Value

None.

Example:

```
//--- example for CArray::Clear()
#include <Arrays\Array.mqh>
//---
void OnStart()
{
    CArray *array=new CArray;
    //---
    if(array==NULL)
    {
        printf("Object create error");
        return;
    }
    //--- use array
    //--- ...
    //--- clear array
    array.Clear();
    //--- delete array
    delete array;
}
```

Sort

Sorts an array to the specified option.

```
void Sort(  
    int mode=0      // Sorting mode  
)
```

Parameters

mode=0

[in] Mode of array sorting.

Return Value

No.

Note

Sorting an array is always ascending. For arrays of primitive data types (CArrayChar, CArrayShort, etc.) , the parameter mode is not used. For the array CArrayObj, multivariate sort should be implemented in the method Sort (int) derived class.

Example:

```
//--- example for CArray::Sort(int)  
#include <Arrays\Array.mqh>  
//---  
void OnStart()  
{  
    CArray *array=new CArray;  
    //---  
    if(array==NULL)  
    {  
        printf("Object create error");  
        return;  
    }  
    //--- sorting by mode 0  
    array.Sort(0);  
    //--- use array  
    //--- ...  
    //--- delete array  
    delete array;  
}
```

Save

Saves data array in the file.

```
virtual bool Save(  
    int file_handle    // File handle  
)
```

Parameters

file_handle

[in] Handle to open earlier, with the function `FileOpen (...)`, binary file.

Return Value

true - if successfully completed, false - if an error.

Example:

```
//--- example for CArray::Save(int)  
#include <Arrays\Array.mqh>  
//---  
void OnStart()  
{  
    int file_handle;  
    CArray *array=new CArray;  
    //---  
    if(array!=NULL)  
    {  
        printf("Object create error");  
        return;  
    }  
    //--- open file  
    file_handle=FileOpen("MyFile.bin",FILE_WRITE|FILE_BIN|FILE_ANSI);  
    if(file_handle>=0)  
    {  
        if(!array.Save(file_handle))  
        {  
            //--- file save error  
            printf("File save: Error %d!",GetLastError());  
            delete array;  
            FileClose(file_handle);  
            //---  
            return;  
        }  
        FileClose(file_handle);  
    }  
    //--- delete array  
    delete array;  
}
```

Load

Loads data array from a file.

```
virtual bool Load(  
    int file_handle    // File handle  
)
```

Parameters

file_handle

[in] Handle to open earlier, with the function `FileOpen (...)`, binary file.

Return Value

true - if successfully completed, false - if an error.

Example:

```
//--- example for CArray::Load(...)  
#include <Arrays\Array.mqh>  
//---  
void OnStart()  
{  
    int file_handle;  
    CArray *array=new CArray;  
    //---  
    if(array!=NULL)  
    {  
        printf("Object create error");  
        return;  
    }  
    //--- open file  
    file_handle=FileOpen("MyFile.bin",FILE_READ|FILE_BIN|FILE_ANSI);  
    if(file_handle>=0)  
    {  
        if(!array.Load(file_handle))  
        {  
            //--- file load error  
            printf("File load: Error %d!",GetLastError());  
            delete array;  
            FileClose(file_handle);  
            //---  
            return;  
        }  
        FileClose(file_handle);  
    }  
    //--- delete array  
    delete array;  
}
```

CArrayChar

CArrayChar class is a class of dynamic array of variables of type char or uchar.

Description

Class CArrayChar provides the ability to work with a dynamic array of variables of type char or uchar. In the class implemented the ability to add / insert / delete elements in an array, sort array, searching in sorted array. In addition, the implemented methods of work with the file.

Declaration

```
class CArrayChar : public CArray
```

Title

```
#include <Arrays\ArrayChar.mqh>
```

Class Methods

Memory control	
Reserve	Allocates memory to increase the size of the array
Resize	Sets a new (smaller) size of the array
Shutdown	Clears the array with a full memory release
Add methods	
Add	Adds an element to the end of the array
AddArray	Adds to the end of the array elements from another array
AddArray	Adds to the end of the array elements from another array
Insert	Inserts an element in the array to the specified position
InsertArray	Inserts an array of elements from another array with the specified position
InsertArray	Inserts an array of elements from another array with the specified position
AssignArray	Copies the array elements from another array
AssignArray	Copies the array elements from another array
Modify methods	
Update	Changes the element at the specified position array
Shift	Moves an item from a given position in the array to the specified offset
Delete methods	
Delete	Removes the element from the specified position array
DeleteRange	Deletes a group of elements from the specified position array
Access methods	

At	Gets the element from the specified position array
Compare methods	
CompareArray	Compares array with another array
CompareArray	Compares array with another array
Sorted array methods	
InsertSort	Inserts element in a sorted array
Search	Searches for an element equal to the model in sorted array
SearchGreat	Searches for an element of more samples in sorted array
SearchLess	Searches for an element less than the sample in the sorted array
SearchGreatOrEqual	Searches for an element greater than or equal to the model in sorted array
SearchLessOrEqual	Searches for an element less than or equal to the model in sorted array
SearchFirst	Finds the first element equal to the model in sorted array
SearchLast	Finds the last element equal to the model in sorted array
Input/output	
virtual Save	Saves data array in the file
virtual Load	Loads data array from a file
virtual Type	Gets the type identifier of the array

Reserve

Allocates memory to increase the size of the array.

```
bool Reserve(  
    int size      // Number  
)
```

Parameters

size

[in] The number of additional elements of the array.

Return Value

true if successful, false - if there was an attempt to seek the amount is less than or equal to zero, or if the array did not increase.

Note

To reduce memory fragmentation, increase the size of the array is made with a step previously given through the method of Step (int) , or 16 (default) .

Example:

```
//--- example for CArrayChar::Reserve(int)  
#include <Arrays\ArrayChar.mqh>  
//---  
void OnStart()  
{  
    CArrayChar *array=new CArrayChar;  
    //---  
    if(array==NULL)  
    {  
        printf("Object create error");  
        return;  
    }  
    //--- reserve memory  
    if(!array.Reserve(1024))  
    {  
        printf("Reserve error");  
        delete array;  
        return;  
    }  
    //--- use array  
    //--- . . .  
    //--- delete array  
    delete array;  
}
```

Resize

Sets a new (smaller) size of the array.

```
bool Resize(  
    int size      // Size  
)
```

Parameters

size

[in] New size of the array.

Return Value

true if successful, false - if there was an attempt to set the size of less than or equal to zero.

Note

Changing the size of the array allows optimal use of memory. Superfluous elements on the right lost. To reduce fragmentation of memory, change the size of the array is made with a step previously given through the method of `Step (int)`, or 16 (default) .

Example:

```
//--- example for CArrayChar::Resize(int)  
#include <Arrays\ArrayChar.mqh>  
//---  
void OnStart()  
{  
    CArrayChar *array=new CArrayChar;  
    //---  
    if(array==NULL)  
    {  
        printf("Object create error");  
        return;  
    }  
    //--- add arrays elements  
    //--- . . .  
    //--- resize array  
    if(!array.Resize(10))  
    {  
        printf("Resize error");  
        delete array;  
        return;  
    }  
    //--- delete array  
    delete array;  
}
```

Shutdown

Clears the array with a full memory release.

```
bool Shutdown()
```

Return Value

true if successful, false - if an error occurred.

Example:

```
//--- example for CArrayChar::Shutdown()
#include <Arrays\ArrayChar.mqh>
//---
void OnStart()
{
    CArrayChar *array=new CArrayChar;
    //---
    if(array==NULL)
    {
        printf("Object create error");
        return;
    }
    //--- add arrays elements
    //--- . . .
    //--- shutdown array
    if(!array.Shutdown())
    {
        printf("Shutdown error");
        delete array;
        return;
    }
    //--- delete array
    delete array;
}
```

Add

Adds an element to the end of the array.

```
bool Add(  
    char element    // Element to add  
)
```

Parameters

element

[in] value of the element to add to the array.

Return Value

true if successful, false - if you can not add an element.

Example:

```
//--- example for CArrayChar::Add(char)  
#include <Arrays\ArrayChar.mqh>  
//---  
void OnStart()  
{  
    CArrayChar *array=new CArrayChar;  
    //---  
    if(array==NULL)  
    {  
        printf("Object create error");  
        return;  
    }  
    //--- add arrays elements  
    for(int i=0;i<100;i++)  
    {  
        if(!array.Add(i))  
        {  
            printf("Element addition error");  
            delete array;  
            return;  
        }  
    }  
    //--- use array  
    //--- . . .  
    //--- delete array  
    delete array;  
}
```

AddArray

Adds to the end of the array elements from another array.

```
bool AddArray(  
    const char& src[]      // Source array  
)
```

Parameters

src[]

[in] Reference to an array of source elements to add.

Return Value

true if successful, false - if you can not add items.

Example:

```
//--- example for CArrayChar::AddArray(const char &[])  
#include <Arrays\ArrayChar.mqh>  
//---  
char src[];  
//---  
void OnStart()  
{  
    CArrayChar *array=new CArrayChar;  
    //---  
    if(array==NULL)  
    {  
        printf("Object create error");  
        return;  
    }  
    //--- add another array  
    if(!array.AddArray(src))  
    {  
        printf("Array addition error");  
        delete array;  
        return;  
    }  
    //--- use array  
    //--- . . .  
    //--- delete array  
    delete array;  
}
```

AddArray

Adds to the end of the array elements from another array.

```
bool AddArray(  
    const CArrayChar* src      // Pointer to the source  
)
```

Parameters

src

[in] Pointer to an instance of class CArrayChar-source elements to add.

Return Value

true if successful, false - if you can not add items.

Example:

```
//--- example for CArrayChar::AddArray(const CArrayChar*)  
#include <Arrays\ArrayChar.mqh>  
//---  
void OnStart()  
{  
    CArrayChar *array=new CArrayChar;  
    //---  
    if(array==NULL)  
    {  
        printf("Object create error");  
        return;  
    }  
    //--- create source array  
    CArrayChar *src=new CArrayChar;  
    if(src==NULL)  
    {  
        printf("Object create error");  
        delete array;  
        return;  
    }  
    //--- add source arrays elements  
    //--- . . .  
    //--- add another array  
    if(!array.AddArray(src))  
    {  
        printf("Array addition error");  
        delete src;  
        delete array;  
        return;  
    }  
    //--- delete source array  
    delete src;
```

```
//--- use array  
//--- . . .  
//--- delete array  
delete array;  
}
```


Insert

Inserts an element in the array to the specified position.

```
bool Insert(  
    char element,    // Element to insert  
    int pos          // Position  
)
```

Parameters

element

[in] Value of the element to be inserted into an array

pos

[in] Position in the array to insert

Return Value

true if successful, false - if you can not insert the element.

Example:

```
//--- example for CArrayChar::Insert(char,int)  
#include <Arrays\ArrayChar.mqh>  
//---  
void OnStart()  
{  
    CArrayChar *array=new CArrayChar;  
    //---  
    if(array==NULL)  
    {  
        printf("Object create error");  
        return;  
    }  
    //--- insert elements  
    for(int i=0;i<100;i++)  
    {  
        if(!array.Insert(i,0))  
        {  
            printf("Insert error");  
            delete array;  
            return;  
        }  
    }  
    //--- use array  
    //--- . . .  
    //--- delete array  
    delete array;  
}
```

InsertArray

Inserts an array of elements from another array with the specified position.

```
bool  InsertArray(  
    const char&  src[],      // Source array  
    int          pos         // Position  
)
```

Parameters

src[]

[in] Reference to an array of source elements to insert

pos

[in] Position in the array to insert

Return Value

true if successful, false - if you can not paste items.

Example:

```
//--- example for CArrayChar::InsertArray(const char &[],int)  
#include <Arrays\ArrayChar.mqh>  
//---  
char src[];  
//---  
void OnStart()  
{  
    CArrayChar *array=new CArrayChar;  
    //---  
    if(array==NULL)  
    {  
        printf("Object create error");  
        return;  
    }  
    //--- insert another array  
    if(!array.InsertArray(src,0))  
    {  
        printf("Array inserting error");  
        delete array;  
        return;  
    }  
    //--- use array  
    //--- . . .  
    //--- delete array  
    delete array;  
}
```

InsertArray

Inserts an array of elements from another array with the specified position.

```
bool InsertArray(  
    CArrayChar* src,      // Pointer to the source  
    int pos              // Position  
)
```

Parameters

src

[in] Pointer to an instance of class CArrayChar-source elements to insert.

pos

[in] Position in the array to insert

Return Value

true if successful, false - if you can not paste items.

Example:

```
//--- example for CArrayChar::InsertArray(const CArrayChar*,int)  
#include <Arrays\ArrayChar.mqh>  
//---  
void OnStart()  
{  
    CArrayChar *array=new CArrayChar;  
    //---  
    if(array==NULL)  
    {  
        printf("Object create error");  
        return;  
    }  
    //--- create source array  
    CArrayChar *src=new CArrayChar;  
    if(src==NULL)  
    {  
        printf("Object create error");  
        delete array;  
        return;  
    }  
    //--- add source arrays elements  
    //--- . . .  
    //--- insert another array  
    if(!array.InsertArray(src,0))  
    {  
        printf("Array inserting error");  
        delete src;  
        delete array;  
    }  
}
```

```
        return;  
    }  
    //--- delete source array  
    delete src;  
    //--- use array  
    //--- . . .  
    //--- delete array  
    delete array;  
}
```

AssignArray

Copies the array elements from another array.

```
bool AssignArray(  
    const char& src[]      // Source array  
)
```

Parameters

src[]

[in] Reference to an array of source elements to copy.

Return Value

true if successful, false - if you can not copy the items.

Example:

```
//--- example for CArrayChar::AssignArray(const char &[])  
#include <Arrays\ArrayChar.mqh>  
//---  
char src[];  
//---  
void OnStart()  
{  
    CArrayChar *array=new CArrayChar;  
    //---  
    if(array==NULL)  
    {  
        printf("Object create error");  
        return;  
    }  
    //--- assign another array  
    if(!array.AssignArray(src))  
    {  
        printf("Array assigned error");  
        delete array;  
        return;  
    }  
    //--- use array  
    //--- . . .  
    //--- delete array  
    delete array;  
}
```

AssignArray

Copies the array elements from another array.

```
bool AssignArray(  
    const CArrayChar* src      // Pointer to the source  
)
```

Parameters

src

[in] Pointer to an instance of class CArrayChar-source element to copy.

Return Value

true if successful, false - if you can not copy the items.

Example:

```
//--- example for CArrayChar::AssignArray(const CArrayChar*)  
#include <Arrays\ArrayChar.mqh>  
//---  
void OnStart()  
{  
    CArrayChar *array=new CArrayChar;  
    //---  
    if(array==NULL)  
    {  
        printf("Object create error");  
        return;  
    }  
    //--- create source array  
    CArrayChar *src =new CArrayChar;  
    if(src==NULL)  
    {  
        printf("Object create error");  
        delete array;  
        return;  
    }  
    //--- add source arrays elements  
    //--- . . .  
    //--- assign another array  
    if(!array.AssignArray(src))  
    {  
        printf("Array assigned error");  
        delete src;  
        delete array;  
        return;  
    }  
    //--- arrays is identical  
    //--- delete source array
```

```
delete src;  
//--- use array  
//--- . . .  
//--- delete array  
delete array;  
}
```

Update

Changes the element at the specified position array.

```
bool Update(  
    int    pos,           // Position  
    char   element       // Value  
)
```

Parameters

pos

[in] Position of the element in the array to change

element

[in] New value element

Return Value

true if successful, false - if you can not change the element.

Example:

```
//--- example for CArrayChar::Update(int,char)  
#include <Arrays\ArrayChar.mqh>  
//---  
void OnStart()  
{  
    CArrayChar *array=new CArrayChar;  
    //---  
    if(array==NULL)  
    {  
        printf("Object create error");  
        return;  
    }  
    //--- add arrays elements  
    //--- . . .  
    //--- update element  
    if(!array.Update(0, 'A'))  
    {  
        printf("Update error");  
        delete array;  
        return;  
    }  
    //--- delete array  
    delete array;  
}
```


Shift

Moves an item from a given position in the array to the specified offset.

```
bool Shift(  
    int pos,          // Position  
    int shift         // Value  
)
```

Parameters

pos

[in] Position of the moved element array

shift

[in] The value of displacement (both positive and negative) .

Return Value

true if successful, false - if you can not move the item.

Example:

```
//--- example for CArrayChar::Shift(int,int)  
#include <Arrays\ArrayChar.mqh>  
//---  
void OnStart()  
{  
    CArrayChar *array=new CArrayChar;  
    //---  
    if(array==NULL)  
    {  
        printf("Object create error");  
        return;  
    }  
    //--- add arrays elements  
    //--- . . .  
    //--- shift element  
    if(!array.Shift(10,-5))  
    {  
        printf("Shift error");  
        delete array;  
        return;  
    }  
    //--- delete array  
    delete array;  
}
```

Delete

Removes the element from the given position in the array.

```
bool Delete(  
    int pos    // Position  
)
```

Parameters

pos

[in] Position removes the element in the array.

Return Value

true if successful, false - if you can not remove the element.

Example:

```
//--- example for CArrayChar::Delete(int)  
#include <Arrays\ArrayChar.mqh>  
//---  
void OnStart()  
{  
    CArrayChar *array=new CArrayChar;  
    //---  
    if(array==NULL)  
    {  
        printf("Object create error");  
        return;  
    }  
    //--- add arrays elements  
    //--- . . .  
    //--- delete element  
    if(!array.Delete(0))  
    {  
        printf("Delete error");  
        delete array;  
        return;  
    }  
    //--- delete array  
    delete array;  
}
```

DeleteRange

Deletes a group of elements from a given position in the array.

```
bool DeleteRange(  
    int from,      // Position of the first element  
    int to         // Positions of the last element  
)
```

Parameters

from

[in] Position of the first removes the element in the array.

to

[in] Position of the last deleted element in the array.

Return Value

true if successful, false - if you can not remove elements.

Example:

```
//--- example for CArrayChar::DeleteRange(int,int)  
#include <Arrays\ArrayChar.mqh>  
//---  
void OnStart()  
{  
    CArrayChar *array=new CArrayChar;  
    //---  
    if(array==NULL)  
    {  
        printf("Object create error");  
        return;  
    }  
    //--- add arrays elements  
    //--- . . .  
    //--- delete elements  
    if(!array.DeleteRange(0,10))  
    {  
        printf("Delete error");  
        delete array;  
        return;  
    }  
    //--- delete array  
    delete array;  
}
```

At

Gets the element from the given position in the array.

```
char At(  
    int pos    // Position  
) const
```

Parameters

pos

[in] Position of the desired element in the array.

Return Value

The value of the element in case of success, CHAR_MAX-if there was an attempt to get an element of not existing positions (the last error ERR_OUT_OF_RANGE) .

Note

Of course, CHAR_MAX may be a valid value of an array element, so having a value, always check the last error code.

Example:

```
//--- example for CArrayChar::At(int)  
#include <Arrays\ArrayChar.mqh>  
//---  
void OnStart()  
{  
    CArrayChar *array=new CArrayChar;  
    //---  
    if(array==NULL)  
    {  
        printf("Object create error");  
        return;  
    }  
    //--- add arrays elements  
    //--- . . .  
    for(int i=0;i<array.Total();i++)  
    {  
        char result=array.At(i);  
        if(result==CHAR_MAX && GetLastError()==ERR_OUT_OF_RANGE)  
        {  
            //--- error of reading from array  
            printf("Get element error");  
            delete array;  
            return;  
        }  
        //--- use element  
        //--- . . .  
    }  
}
```

```
//--- delete array  
delete array;  
}
```

CompareArray

Compares array with another array.

```
bool CompareArray(  
    const char& src[]      // Source array  
) const
```

Parameters

src[]

[in] Reference to an array of source elements for comparison.

Return Value

true if arrays are equal, false - if not.

Example:

```
///--- example for CArrayChar::CompareArray(const char &[])  
#include <Arrays\ArrayChar.mqh>  
///---  
char src[];  
///---  
void OnStart()  
{  
    CArrayChar *array=new CArrayChar;  
    ///---  
    if(array==NULL)  
    {  
        printf("Object create error");  
        return;  
    }  
    ///--- compare with another array  
    int result=array.CompareArray(src);  
    ///--- delete array  
    delete array;  
}
```

CompareArray

Compares array with another array.

```
bool CompareArray(  
    const CArrayChar* src      // Pointer to the sources  
    ) const
```

Parameters

src

[in] Pointer to an instance of class CArrayChar-source elements for comparison.

Return Value

true if arrays are equal, false - if not.

Example:

```
///--- example for CArrayChar::CompareArray(const CArrayChar*)  
#include <Arrays\ArrayChar.mqh>  
///---  
void OnStart()  
{  
    CArrayChar *array=new CArrayChar;  
    ///---  
    if(array==NULL)  
    {  
        printf("Object create error");  
        return;  
    }  
    ///--- create source array  
    CArrayChar *src=new CArrayChar;  
    if(src==NULL)  
    {  
        printf("Object create error");  
        delete array;  
        return;  
    }  
    ///--- add source arrays elements  
    ///--- . . .  
    ///--- compare with another array  
    int result=array.CompareArray(src);  
    ///--- delete arrays  
    delete src;  
    delete array;  
}
```

InsertSort

Inserts element in a sorted array.

```
bool  InsertSort(  
    char  element      // Element to insert  
)
```

Parameters

element

[in] Value of the element to be inserted into a sorted array

Return Value

true if successful, false - if you can not insert the element.

Example:

```
//--- example for CArrayChar::InsertSort(char)  
#include <Arrays\ArrayChar.mqh>  
//---  
void OnStart()  
{  
    CArrayChar *array=new CArrayChar;  
    //---  
    if(array==NULL)  
    {  
        printf("Object create error");  
        return;  
    }  
    //--- add arrays elements  
    //--- . . .  
    //--- sort array  
    array.Sort();  
    //--- insert element  
    if(!array.InsertSort('A'))  
    {  
        printf("Insert error");  
        delete array;  
        return;  
    }  
    //--- delete array  
    delete array;  
}
```


Search

Searches for an element equal to the sample in the sorted array.

```
int Search(  
    char element    // Sample  
    ) const
```

Parameters

element

[in] The sample element to search in the array.

Return Value

The position of the found element, if successful, -1 - if the item was not found.

Example:

```
//--- example for CArrayChar::Search(char)  
#include <Arrays\ArrayChar.mqh>  
//---  
void OnStart()  
{  
    CArrayChar *array=new CArrayChar;  
    //---  
    if(array==NULL)  
    {  
        printf("Object create error");  
        return;  
    }  
    //--- add arrays elements  
    //--- . . .  
    //--- sort array  
    array.Sort();  
    //--- search element  
    if(array.Search('A')!=-1) printf("Element found");  
    else                      printf("Element not found");  
    //--- delete array  
    delete array;  
}
```

SearchGreat

Searches for an element of more samples in sorted array.

```
int SearchGreat(  
    char element    // Sample  
    ) const
```

Parameters

element

[in] The sample element to search in the array.

Return Value

The position of the found element, if successful, -1 - if the item was not found.

Example:

```
//--- example for CArrayChar::SearchGreat(char)  
#include <Arrays\ArrayChar.mqh>  
//---  
void OnStart()  
{  
    CArrayChar *array=new CArrayChar;  
    //---  
    if(array==NULL)  
    {  
        printf("Object create error");  
        return;  
    }  
    //--- add arrays elements  
    //--- . . .  
    //--- sort array  
    array.Sort();  
    //--- search element  
    if(array.SearchGreat('A')!=-1) printf("Element found");  
    else                          printf("Element not found");  
    //--- delete array  
    delete array;  
}
```

SearchLess

Searches for an element less than the sample in the sorted array.

```
int SearchLess(  
    char element    // Sample  
    ) const
```

Parameters

element

[in] The sample element to search in the array.

Return Value

The position of the found element, if successful, -1 - if the item was not found.

Example:

```
//--- example for CArrayChar::SearchLess(char)  
#include <Arrays\ArrayChar.mqh>  
//---  
void OnStart()  
{  
    CArrayChar *array=new CArrayChar;  
    //---  
    if(array==NULL)  
    {  
        printf("Object create error");  
        return;  
    }  
    //--- add arrays elements  
    //--- . . .  
    //--- sort array  
    array.Sort();  
    //--- search element  
    if(array.SearchLess('A')!=-1) printf("Element found");  
    else                          printf("Element not found");  
    //--- delete array  
    delete array;  
}
```

SearchGreatOrEqual

Searches for an element greater than or equal to the sample in the sorted array

```
int SearchGreatOrEqual(  
    char element    // Sample  
) const
```

Parameters

element

[in] The sample element to search in the array.

Return Value

The position of the found element, if successful, -1 - if the item was not found.

Example:

```
//--- example for CArrayChar::SearchGreatOrEqual(char)  
#include <Arrays\ArrayChar.mqh>  
//---  
void OnStart()  
{  
    CArrayChar *array=new CArrayChar;  
    //---  
    if(array==NULL)  
    {  
        printf("Object create error");  
        return;  
    }  
    //--- add arrays elements  
    //--- . . .  
    //--- sort array  
    array.Sort();  
    //--- search element  
    if(array.SearchGreatOrEqual('A')!=-1) printf("Element found");  
    else                                printf("Element not found");  
    //--- delete array  
    delete array;  
}
```

SearchLessOrEqual

Searches for an element less than or equal to the sample in the sorted array.

```
int SearchLessOrEqual(  
    char element    // Sample  
    ) const
```

Parameters

element

[in] The sample element to search in the array.

Return Value

The position of the found element, if successful, -1 - if the item was not found.

Example:

```
//--- example for CArrayChar::SearchLessOrEqual(char)  
#include <Arrays\ArrayChar.mqh>  
//---  
void OnStart()  
{  
    CArrayChar *array=new CArrayChar;  
    //---  
    if(array==NULL)  
    {  
        printf("Object create error");  
        return;  
    }  
    //--- add arrays elements  
    //--- . . .  
    //--- sort array  
    array.Sort();  
    //--- search element  
    if(array.SearchLessOrEqual('A')!=-1) printf("Element found");  
    else                                printf("Element not found");  
    //--- delete array  
    delete array;  
}
```

SearchFirst

Finds the first element equal to the sample in the sorted array.

```
int SearchFirst(  
    char element    // Sample  
    ) const
```

Parameters

element

[in] The sample element to search in the array.

Return Value

The position of the found element, if successful, -1 - if the item was not found.

Example:

```
//--- example for CArrayChar::SearchFirst(char)  
#include <Arrays\ArrayChar.mqh>  
//---  
void OnStart()  
{  
    CArrayChar *array=new CArrayChar;  
    //---  
    if(array==NULL)  
    {  
        printf("Object create error");  
        return;  
    }  
    //--- add arrays elements  
    //--- . . .  
    //--- sort array  
    array.Sort();  
    //--- search element  
    if(array.SearchFirst('A')!=-1) printf("Element found");  
    else                          printf("Element not found");  
    //--- delete array  
    delete array;  
}
```

SearchLast

Finds the last element equal to the model in sorted array.

```
int SearchLast(  
    char element    // Sample  
    ) const
```

Parameters

element

[in] The sample element to search in the array.

Return Value

The position of the found element, if successful, -1 - if the item was not found.

Example:

```
///--- example for CArrayChar::SearchLast(char)  
#include <Arrays\ArrayChar.mqh>  
///---  
void OnStart()  
{  
    CArrayChar *array=new CArrayChar;  
    ///---  
    if(array==NULL)  
    {  
        printf("Object create error");  
        return;  
    }  
    ///--- add arrays elements  
    ///--- . . .  
    ///--- sort array  
    array.Sort();  
    ///--- search element  
    if(array.SearchLast('A')!=-1) printf("Element found");  
    else                          printf("Element not found");  
    ///--- delete array  
    delete array;  
}
```

Save

Saves data array in the file.

```
virtual bool Save(  
    int file_handle    // File handle  
)
```

Parameters

file_handle

[in] Handle to open earlier, with the function `FileOpen (...)`, binary file.

Return Value

true - if successfully completed, false - if an error.

Example:

```
//--- example for CArrayChar::Save(int)  
#include <Arrays\ArrayChar.mqh>  
//---  
void OnStart()  
{  
    int file_handle;  
    CArrayChar *array=new CArrayChar;  
    //---  
    if(array!=NULL)  
    {  
        printf("Object create error");  
        return;  
    }  
    //--- open file  
    file_handle=FileOpen("MyFile.bin",FILE_WRITE|FILE_BIN|FILE_ANSI);  
    if(file_handle>=0)  
    {  
        if(!array.Save(file_handle))  
        {  
            //--- file save error  
            printf("File save: Error %d!",GetLastError());  
            delete array;  
            FileClose(file_handle);  
            //---  
            return;  
        }  
        FileClose(file_handle);  
    }  
    //--- delete array  
    delete array;  
}
```


Load

Loads data array from a file.

```
virtual bool Load(
    int file_handle    // File handle
)
```

Parameters

file_handle

[in] Handle to open earlier, with the function `FileOpen (...)`, binary file.

Return Value

true - if successfully completed, false - if an error.

Example:

```
//--- example for CArrayChar::Load(int)
#include <Arrays\ArrayChar.mqh>
//---
void OnStart()
{
    int file_handle;
    CArrayChar *array=new CArrayChar;
    //---
    if(array!=NULL)
    {
        printf("Object create error");
        return;
    }
    //--- open file
    file_handle=FileOpen("MyFile.bin",FILE_READ|FILE_BIN|FILE_ANSI);
    if(file_handle>=0)
    {
        if(!array.Load(file_handle))
        {
            //--- file load error
            printf("File load: Error %d!",GetLastError());
            delete array;
            FileClose(file_handle);
            //---
            return;
        }
        FileClose(file_handle);
    }
    //--- use arrays elements
    for(int i=0;i<array.Total();i++)
    {
        printf("Element[%d] = '%c'",i,array.At(i));
```

```
    }  
    ///--- delete array  
    delete array;  
}
```

Type

Gets the type identifier of the array.

```
virtual int Type() const
```

Return Value

ID type of the array (for CArrayChar - 77) .

Example:

```
//--- example for CArrayChar::Type()
#include <Arrays\ArrayChar.mqh>
//---
void OnStart()
{
    CArrayChar *array=new CArrayChar;
    //---
    if(array==NULL)
    {
        printf("Object create error");
        return;
    }
    //--- get array type
    int type=array.Type();
    //--- delete array
    delete array;
}
```

CArrayShort

CArrayShort class is a class of dynamic array of variables of type short or ushort.

Description

Class CArrayShort provides the ability to work with a dynamic array of variables of type short or ushort. In the class implemented the ability to add / insert / delete elements in an array, sort array, searching in sorted array. In addition, the implemented methods of work with the file.

Declaration

```
class CArrayShort : public CArray
```

Title

```
#include <Arrays\ArrayShort.mqh>
```

Class Methods

Memory control	
Reserve	Allocates memory to increase the size of the array
Resize	Sets a new (smaller) size of the array
Shutdown	Clears the array with a full memory release
Add methods	
Add	Adds an element to the end of the array
AddArray	Adds to the end of the array elements from another array
AddArray	Adds to the end of the array elements from another array
Insert	Inserts an element in the array to the specified position
InsertArray	Inserts an array of elements from another array with the specified position
InsertArray	Inserts an array of elements from another array with the specified position
AssignArray	Copies the array elements from another array
AssignArray	Copies the array elements from another array
Update methods	
Update	Changes the element at the specified position array
Shift	Moves an item from a given position in the array to the specified offset
Delete methods	
Delete	Removes the element from the specified position array

DeleteRange	Deletes a group of elements from the specified position array
Access methods	
At	Gets the element from the specified position array
Compare methods	
CompareArray	Compares array with another array
CompareArray	Compares array with another array
Sorted array operations	
InsertSort	Inserts element in a sorted array
Search	Searches for an element equal to the model in sorted array
SearchGreat	Searches for an element of more samples in sorted array
SearchLess	Searches for an element less than the sample in the sorted array
SearchGreatOrEqual	Searches for an element greater than or equal to the model in sorted array
SearchLessOrEqual	Searches for an element less than or equal to the model in sorted array
SearchFirst	Finds the first element equal to the model in sorted array
SearchLast	Finds the last element equal to the model in sorted array
Input/output	
virtual Save	Saves data array in the file
virtual Load	Loads data array from a file
virtual Type	Gets the type identifier of the array

Reserve

Allocates memory to increase the size of the array.

```
bool Reserve(  
    int size      // Number  
)
```

Parameters

size

[in] The number of additional elements of the array.

Return Value

true if successful, false - if there was an attempt to seek the amount is less than or equal to zero, or if the array did not increase.

Note

To reduce memory fragmentation, increase the size of the array is made with a step previously given through the method of Step (int) , or 16 (default) .

Example:

```
//--- example for CArrayShort::Reserve(int)  
#include <Arrays\ArrayShort.mqh>  
//---  
void OnStart()  
{  
    CArrayShort *array=new CArrayShort;  
    //---  
    if(array==NULL)  
    {  
        printf("Object create error");  
        return;  
    }  
    //--- reserve memory  
    if(!array.Reserve(1024))  
    {  
        printf("Reserve error");  
        delete array;  
        return;  
    }  
    //--- use array  
    //--- . . .  
    //--- delete array  
    delete array;  
}
```

Resize

Sets a new (smaller) size of the array.

```
bool Resize(  
    int size      // Size  
)
```

Parameters

size

[in] New size of the array.

Return Value

true if successful, false - if there was an attempt to set the size of less than or equal to zero.

Note

Changing the size of the array allows optimal use of memory. Superfluous elements on the right lost. To reduce fragmentation of memory, change the size of the array is made with a step previously given through the method of Step (int) , or 16 (default) .

Example:

```
//--- example for CArrayShort::Resize(int)  
#include <Arrays\ArrayShort.mqh>  
//---  
void OnStart()  
{  
    CArrayShort *array=new CArrayShort;  
    //---  
    if(array==NULL)  
    {  
        printf("Object create error");  
        return;  
    }  
    //--- add arrays elements  
    //--- . . .  
    //--- resize array  
    if(!array.Resize(10))  
    {  
        printf("Resize error");  
        delete array;  
        return;  
    }  
    //--- delete array  
    delete array;  
}
```

Shutdown

Clears the array with a full memory release.

```
bool Shutdown()
```

Return Value

true if successful, false - if an error occurred.

Example:

```
//--- example for CArrayShort::Shutdown()
#include <Arrays\ArrayShort.mqh>
//---
void OnStart()
{
    CArrayShort *array=new CArrayShort;
    //---
    if(array==NULL)
    {
        printf("Object create error");
        return;
    }
    //--- add arrays elements
    //--- . . .
    //--- shutdown array
    if(!array.Shutdown())
    {
        printf("Shutdown error");
        delete array;
        return;
    }
    //--- delete array
    delete array;
}
```


Add

Adds an element to the end of the array.

```
bool Add(  
    short element    // Element to add  
)
```

Parameters

element

[in] Value of the element to add to the array.

Return Value

true if successful, false - if you can not add an element.

Example:

```
//--- example for CArrayShort::Add(short)  
#include <Arrays\ArrayShort.mqh>  
//---  
void OnStart()  
{  
    CArrayShort *array=new CArrayShort;  
    //---  
    if(array==NULL)  
    {  
        printf("Object create error");  
        return;  
    }  
    //--- add arrays elements  
    for(int i=0;i<100;i++)  
    {  
        if(!array.Add(i))  
        {  
            printf("Element addition error");  
            delete array;  
            return;  
        }  
    }  
    //--- use array  
    //--- . . .  
    //--- delete array  
    delete array;  
}
```

AddArray

Adds to the end of the array elements from another array.

```
bool AddArray(  
    const short& src[]    // Source array  
)
```

Parameters

src[]

[in] Reference to an array of source elements to add.

Return Value

true if successful, false - if you can not add items.

Example:

```
///--- example for CArrayShort::AddArray(const short &[])  
#include <Arrays\ArrayShort.mqh>  
///---  
short src[];  
///---  
void OnStart()  
{  
    CArrayShort *array=new CArrayShort;  
    ///---  
    if(array==NULL)  
    {  
        printf("Object create error");  
        return;  
    }  
    ///--- add another array  
    if(!array.AddArray(src))  
    {  
        printf("Array addition error");  
        delete array;  
        return;  
    }  
    ///--- use array  
    ///--- . . .  
    ///--- delete array  
    delete array;  
}
```

AddArray

Adds to the end of the array elements from another array.

```
bool AddArray(  
    const CArrayShort* src      // Pointer to the source  
)
```

Parameters

src

[in] Pointer to an instance of class CArrayShort-source elements to add.

Return Value

true if successful, false - if you can not add items.

Example:

```
///--- example for CArrayShort::AddArray(const CArrayShort*)  
#include <Arrays\ArrayShort.mqh>  
///---  
void OnStart()  
{  
    CArrayShort *array=new CArrayShort;  
    ///---  
    if(array==NULL)  
    {  
        printf("Object create error");  
        return;  
    }  
    ///--- create source array  
    CArrayShort *src=new CArrayShort;  
    if(src==NULL)  
    {  
        printf("Object create error");  
        delete array;  
        return;  
    }  
    ///--- add source arrays elements  
    ///--- . . .  
    ///--- add another array  
    if(!array.AddArray(src))  
    {  
        printf("Array addition error");  
        delete src;  
        delete array;  
        return;  
    }  
    ///--- delete source array  
    delete src;
```

```
//--- use array  
//--- . . .  
//--- delete array  
delete array;  
}
```

Insert

Inserts an element in the array to the specified position.

```
bool Insert(  
    short element,    // Element to insert  
    int pos           // Position  
)
```

Parameters

element

[in] Value of the element to be inserted into an array

pos

[in] Position in the array to insert

Return Value

true if successful, false - if you can not insert the element.

Example:

```
//--- example for CArrayShort::Insert(short,int)  
#include <Arrays\ArrayShort.mqh>  
//---  
void OnStart()  
{  
    CArrayShort *array=new CArrayShort;  
    //---  
    if(array==NULL)  
    {  
        printf("Object create error");  
        return;  
    }  
    //--- insert elements  
    for(int i=0;i<100;i++)  
    {  
        if(!array.Insert(i,0))  
        {  
            printf("Insert error");  
            delete array;  
            return;  
        }  
    }  
    //--- use array  
    //--- . . .  
    //--- delete array  
    delete array;  
}
```

InsertArray

Inserts an array of elements from another array specified position.

```
bool  InsertArray(  
    const short&  src[],      // Source array  
    int           pos         // Position  
)
```

Parameters

src[]

[in] Reference to an array of source elements to insert

pos

[in] Position in the array to insert

Return Value

true if successful, false - if you can not paste items.

Example:

```
//--- example for CArrayShort::InsertArray(const short &[],int)  
#include <Arrays\ArrayShort.mqh>  
//---  
short src[];  
//---  
void OnStart()  
{  
    CArrayShort *array=new CArrayShort;  
    //---  
    if(array==NULL)  
    {  
        printf("Object create error");  
        return;  
    }  
    //--- insert another array  
    if(!array.InsertArray(src,0))  
    {  
        printf("Array inserting error");  
        delete array;  
        return;  
    }  
    //--- use array  
    //--- . . .  
    //--- delete array  
    delete array;  
}
```

InsertArray

Inserts an array of elements from another array ukazannogy position.

```
bool InsertArray(  
    CArrayShort* src,      // Pointer to the source  
    int pos               // Position  
)
```

Parameters

src

[in] Pointer to an instance of class CArrayShort-source elements to insert.

pos

[in] Position in the array to insert

Return Value

true if successful, false - if you can not paste items.

Example:

```
//--- example for CArrayShort::InsertArray(const CArrayShort*,int)  
#include <Arrays\ArrayShort.mqh>  
//---  
void OnStart()  
{  
    CArrayShort *array=new CArrayShort;  
    //---  
    if(array==NULL)  
    {  
        printf("Object create error");  
        return;  
    }  
    //--- create source array  
    CArrayShort *src=new CArrayShort;  
    if(src==NULL)  
    {  
        printf("Object create error");  
        delete array;  
        return;  
    }  
    //--- add source arrays elements  
    //--- . . .  
    //--- insert another array  
    if(!array.InsertArray(src,0))  
    {  
        printf("Array inserting error");  
        delete src;  
        delete array;  
    }  
}
```

```
        return;  
    }  
    //--- delete source array  
    delete src;  
    //--- use array  
    //--- . . .  
    //--- delete array  
    delete array;  
}
```


AssignArray

Copies the array elements from another array.

```
bool AssignArray(  
    const short& src[]      // Source array  
)
```

Parameters

src[]

[in] Reference to an array of source elements to copy.

Return Value

true if successful, false - if you can not copy the items.

Example:

```
//--- example for CArrayShort::AssignArray(const short &[])  
#include <Arrays\ArrayShort.mqh>  
//---  
short src[];  
//---  
void OnStart()  
{  
    CArrayShort *array=new CArrayShort;  
    //---  
    if(array==NULL)  
    {  
        printf("Object create error");  
        return;  
    }  
    //--- assign another array  
    if(!array.AssignArray(src))  
    {  
        printf("Array assigned error");  
        delete array;  
        return;  
    }  
    //--- use array  
    //--- . . .  
    //--- delete array  
    delete array;  
}
```

AssignArray

Copies the array elements from another array.

```
bool AssignArray(  
    const CArrayShort* src      // Pointer to the source  
)
```

Parameters

src

[in] Pointer to an instance of class CArrayShort-source element to copy.

Return Value

true if successful, false - if you can not copy the items.

Example:

```
//--- example for CArrayShort::AssignArray(const CArrayShort*)  
#include <Arrays\ArrayShort.mqh>  
//---  
void OnStart()  
{  
    CArrayShort *array=new CArrayShort;  
    //---  
    if(array==NULL)  
    {  
        printf("Object create error");  
        return;  
    }  
    //--- create source array  
    CArrayShort *src =new CArrayShort;  
    if(src==NULL)  
    {  
        printf("Object create error");  
        delete array;  
        return;  
    }  
    //--- add source arrays elements  
    //--- . . .  
    //--- assign another array  
    if(!array.AssignArray(src))  
    {  
        printf("Array assigned error");  
        delete src;  
        delete array;  
        return;  
    }  
    //--- arrays is identical  
    //--- delete source array
```

```
delete src;  
//--- use array  
//--- . . .  
//--- delete array  
delete array;  
}
```

Update

Changes the element at the specified position array.

```
bool Update(  
    int    pos,          // Position  
    short  element       // Value  
)
```

Parameters

pos

[in] Position of the element in the array to change

element

[in] New value element

Return Value

true if successful, false - if you can not change the element.

Example:

```
//--- example for CArrayShort::Update(int,short)  
#include <Arrays\ArrayShort.mqh>  
//---  
void OnStart()  
{  
    CArrayShort *array=new CArrayShort;  
    //---  
    if(array==NULL)  
    {  
        printf("Object create error");  
        return;  
    }  
    //--- add arrays elements  
    //--- . . .  
    //--- update element  
    if(!array.Update(0,100))  
    {  
        printf("Update error");  
        delete array;  
        return;  
    }  
    //--- delete array  
    delete array;  
}
```

Shift

Moves an item from a given position in the array to the specified offset.

```
bool Shift(  
    int pos,          // Positions  
    int shift         // Shift  
)
```

Parameters

pos

[in] Position of the moved element array

shift

[in] The value of displacement (both positive and negative) .

Return Value

true if successful, false - if you can not move the item.

Example:

```
//--- example for CArrayShort::Shift(int,int)  
#include <Arrays\ArrayShort.mqh>  
//---  
void OnStart()  
{  
    CArrayShort *array=new CArrayShort;  
    //---  
    if(array==NULL)  
    {  
        printf("Object create error");  
        return;  
    }  
    //--- add arrays elements  
    //--- . . .  
    //--- shift element  
    if(!array.Shift(10,-5))  
    {  
        printf("Shift error");  
        delete array;  
        return;  
    }  
    //--- delete array  
    delete array;  
}
```

Delete

Removes the element from the given position in the array.

```
bool Delete(  
    int pos    // Position  
)
```

Parameters

pos

[in] Position removes the element in the array.

Return Value

true if successful, false - if you can not remove the element.

Example:

```
///--- example for CArrayShort::Delete(int)  
#include <Arrays\ArrayShort.mqh>  
///---  
void OnStart()  
{  
    CArrayShort *array=new CArrayShort;  
    ///---  
    if(array==NULL)  
    {  
        printf("Object create error");  
        return;  
    }  
    ///--- add arrays elements  
    ///--- . . .  
    ///--- delete element  
    if(!array.Delete(0))  
    {  
        printf("Delete error");  
        delete array;  
        return;  
    }  
    ///--- delete array  
    delete array;  
}
```

DeleteRange

Deletes a group of elements from a given position in the array.

```
bool DeleteRange(  
    int from,      // Position of the first element  
    int to         // Position of the last element  
)
```

Parameters

from

[in] Position of the first removed element in the array.

to

[in] Position of the last deleted element in the array.

Return Value

true if successful, false - if you can not remove elements.

Example:

```
//--- example for CArrayShort::DeleteRange(int,int)  
#include <Arrays\ArrayShort.mqh>  
//---  
void OnStart()  
{  
    CArrayShort *array=new CArrayShort;  
    //---  
    if(array==NULL)  
    {  
        printf("Object create error");  
        return;  
    }  
    //--- add arrays elements  
    //--- . . .  
    //--- delete elements  
    if(!array.DeleteRange(0,10))  
    {  
        printf("Delete error");  
        delete array;  
        return;  
    }  
    //--- delete array  
    delete array;  
}
```

At

Gets the element from the given position in the array.

```
short At(
    int pos    // Position
) const
```

Parameters

pos

[in] Position of the desired element in the array.

Return Value

The value of the element in case of success, `SHORT_MAX`-if there was an attempt to get an element of not existing positions (the last error `ERR_OUT_OF_RANGE`) .

Note

Of course, `SHORT_MAX` may be a valid value of an array element, so having a value, always check the last error code.

Example:

```
//--- example for CArrayShort::At(int)
#include <Arrays\ArrayShort.mqh>
//---
void OnStart()
{
    CArrayShort *array=new CArrayShort;
    //---
    if(array==NULL)
    {
        printf("Object create error");
        return;
    }
    //--- add arrays elements
    //--- . . .
    for(int i=0;i<array.Total();i++)
    {
        short result=array.At(i);
        if(result==SHORT_MAX && GetLastError()==ERR_OUT_OF_RANGE)
        {
            //--- error of reading from array
            printf("Get element error");
            delete array;
            return;
        }
        //--- use element
        //--- . . .
    }
}
```



```
//--- delete array  
delete array;  
}
```

CompareArray

Compares array with another array.

```
bool CompareArray(  
    const short& src[]      // Source array  
    ) const
```

Parameters

src[]

[in] Reference to an array of source elements for comparison.

Return Value

true if arrays are equal, false - if not.

Example:

```
//--- example for CArrayShort::CompareArray(const short &[])  
#include <Arrays\ArrayShort.mqh>  
//---  
short src[];  
//---  
void OnStart()  
{  
    CArrayShort *array=new CArrayShort;  
    //---  
    if(array==NULL)  
    {  
        printf("Object create error");  
        return;  
    }  
    //--- compare with another array  
    int result=array.CompareArray(src);  
    //--- delete array  
    delete array;  
}
```

CompareArray

Compares array with another array.

```
bool CompareArray(  
    const CArrayShort* src      // Pointer to the source  
    ) const
```

Parameters

src

[in] Pointer to an instance of class CArrayShort-source elements for comparison.

Return Value

true if arrays are equal, false - if not.

Example:

```
//--- example for CArrayShort::CompareArray(const CArrayShort*)  
#include <Arrays\ArrayShort.mqh>  
//---  
void OnStart()  
{  
    CArrayShort *array=new CArrayShort;  
    //---  
    if(array==NULL)  
    {  
        printf("Object create error");  
        return;  
    }  
    //--- create source array  
    CArrayShort *src=new CArrayShort;  
    if(src==NULL)  
    {  
        printf("Object create error");  
        delete array;  
        return;  
    }  
    //--- add source arrays elements  
    //--- . . .  
    //--- compare with another array  
    int result=array.CompareArray(src);  
    //--- delete arrays  
    delete src;  
    delete array;  
}
```

InsertSort

Inserts element in a sorted array.

```
bool  InsertSort(  
    short  element      // Element to insert  
)
```

Parameters

element

[in] Value of the element to be inserted into a sorted array

Return Value

true if successful, false - if you can not insert the element.

Example:

```
//--- example for CArrayShort::InsertSort(short)  
#include <Arrays\ArrayShort.mqh>  
//---  
void OnStart()  
{  
    CArrayShort *array=new CArrayShort;  
    //---  
    if(array==NULL)  
    {  
        printf("Object create error");  
        return;  
    }  
    //--- add arrays elements  
    //--- . . .  
    //--- sort array  
    array.Sort();  
    //--- insert element  
    if(!array.InsertSort(100))  
    {  
        printf("Insert error");  
        delete array;  
        return;  
    }  
    //--- delete array  
    delete array;  
}
```

Search

Searches for an element equal to the sample in the sorted array.

```
int Search(  
    short element    // Sample  
    ) const
```

Parameters

element

[in] The sample element to search in the array.

Return Value

The position of the found element, if successful, -1 - if the item was not found.

Example:

```
//--- example for CArrayShort::Search(short)  
#include <Arrays\ArrayShort.mqh>  
//---  
void OnStart()  
{  
    CArrayShort *array=new CArrayShort;  
    //---  
    if(array==NULL)  
    {  
        printf("Object create error");  
        return;  
    }  
    //--- add arrays elements  
    //--- . . .  
    //--- sort array  
    array.Sort();  
    //--- search element  
    if(array.Search(100)!=-1) printf("Element found");  
    else                     printf("Element not found");  
    //--- delete array  
    delete array;  
}
```

SearchGreat

Searches for an element of more samples in sorted array.

```
int SearchGreat(  
    short element    // Sample  
) const
```

Parameters

element

[in] The sample element to search in the array.

Return Value

The position of the found element, if successful, -1 - if the item was not found.

Example:

```
//--- example for CArrayShort::SearchGreat(short)  
#include <Arrays\ArrayShort.mqh>  
//---  
void OnStart()  
{  
    CArrayShort *array=new CArrayShort;  
    //---  
    if(array==NULL)  
    {  
        printf("Object create error");  
        return;  
    }  
    //--- add arrays elements  
    //--- . . .  
    //--- sort array  
    array.Sort();  
    //--- search element  
    if(array.SearchGreat(100)!=-1) printf("Element found");  
    else                          printf("Element not found");  
    //--- delete array  
    delete array;  
}
```

SearchLess

Searches for an element less than the sample in the sorted array.

```
int SearchLess(  
    short element    // Sample  
    ) const
```

Parameters

element

[in] The sample element to search in the array.

Return Value

The position of the found element, if successful, -1 - if the item was not found.

Example:

```
//--- example for CArrayShort::SearchLess(short)  
#include <Arrays\ArrayShort.mqh>  
//---  
void OnStart()  
{  
    CArrayShort *array=new CArrayShort;  
    //---  
    if(array==NULL)  
    {  
        printf("Object create error");  
        return;  
    }  
    //--- add arrays elements  
    //--- . . .  
    //--- sort array  
    array.Sort();  
    //--- search element  
    if(array.SearchLess(100)!=-1) printf("Element found");  
    else                          printf("Element not found");  
    //--- delete array  
    delete array;  
}
```

SearchGreatOrEqual

Searches for an element greater than or equal to the sample in the sorted array.

```
int SearchGreatOrEqual(  
    short element    // Sample  
) const
```

Parameters

element

[in] The sample element to search in the array.

Return Value

The position of the found element, if successful, -1 - if the item was not found.

Example:

```
//--- example for CArrayShort::SearchGreatOrEqual(short)  
#include <Arrays\ArrayShort.mqh>  
//---  
void OnStart()  
{  
    CArrayShort *array=new CArrayShort;  
    //---  
    if(array==NULL)  
    {  
        printf("Object create error");  
        return;  
    }  
    //--- add arrays elements  
    //--- . . .  
    //--- sort array  
    array.Sort();  
    //--- search element  
    if(array.SearchGreatOrEqual(100)!=-1) printf("Element found");  
    else                                printf("Element not found");  
    //--- delete array  
    delete array;  
}
```


SearchLessOrEqual

Searches for an element less than or equal to the sample in the sorted array.

```
int SearchLessOrEqual(  
    short element    // Sample  
) const
```

Parameters

element

[in] The sample element to search in the array.

Return Value

The position of the found element, if successful, -1 - if the item was not found.

Example:

```
//--- example for CArrayShort::SearchLessOrEqual(short)  
#include <Arrays\ArrayShort.mqh>  
//---  
void OnStart()  
{  
    CArrayShort *array=new CArrayShort;  
    //---  
    if(array==NULL)  
    {  
        printf("Object create error");  
        return;  
    }  
    //--- add arrays elements  
    //--- . . .  
    //--- sort array  
    array.Sort();  
    //--- search element  
    if(array.SearchLessOrEqual(100)!=-1) printf("Element found");  
    else                                printf("Element not found");  
    //--- delete array  
    delete array;  
}
```

SearchFirst

Finds the first element equal to the sample in the sorted array.

```
int SearchFirst(  
    short element    // Sample  
) const
```

Parameters

element

[in] The sample element to search in the array.

Return Value

The position of the found element, if successful, -1 - if the item was not found.

Example:

```
//--- example for CArrayShort::SearchFirst(short)  
#include <Arrays\ArrayShort.mqh>  
//---  
void OnStart()  
{  
    CArrayShort *array=new CArrayShort;  
    //---  
    if(array==NULL)  
    {  
        printf("Object create error");  
        return;  
    }  
    //--- add arrays elements  
    //--- . . .  
    //--- sort array  
    array.Sort();  
    //--- search element  
    if(array.SearchFirst(100)!=-1) printf("Element found");  
    else                          printf("Element not found");  
    //--- delete array  
    delete array;  
}
```

SearchLast

Finds the last element equal to the model in sorted array.

```
int SearchLast(  
    short element    // Sample  
    ) const
```

Parameters

element

[in] The sample element to search in the array.

Return Value

The position of the found element, if successful, -1 - if the item was not found.

Example:

```
//--- example for CArrayShort::SearchLast(short)  
#include <Arrays\ArrayShort.mqh>  
//---  
void OnStart()  
{  
    CArrayShort *array=new CArrayShort;  
    //---  
    if(array==NULL)  
    {  
        printf("Object create error");  
        return;  
    }  
    //--- add arrays elements  
    //--- . . .  
    //--- sort array  
    array.Sort();  
    //--- search element  
    if(array.SearchLast(100)!=-1) printf("Element found");  
    else                          printf("Element not found");  
    //--- delete array  
    delete array;  
}
```

Save

Saves data array in the file.

```
virtual bool Save(  
    int file_handle    // File handle  
)
```

Parameters

file_handle

[in] Handle to open earlier, with the function `FileOpen (...)`, binary file.

Return Value

true - if successfully completed, false - if an error.

Example:

```
//--- example for CArrayShort::Save(int)  
#include <Arrays\ArrayShort.mqh>  
//---  
void OnStart()  
{  
    int file_handle;  
    CArrayShort *array=new CArrayShort;  
    //---  
    if(array!=NULL)  
    {  
        printf("Object create error");  
        return;  
    }  
    //--- add 100 arrays elements  
    for(int i=0;i<100;i++)  
    {  
        array.Add(i);  
    }  
    //--- open file  
    file_handle=FileOpen("MyFile.bin", FILE_WRITE|FILE_BIN|FILE_ANSI);  
    if(file_handle>=0)  
    {  
        if(!array.Save(file_handle))  
        {  
            //--- file save error  
            printf("File save: Error %d!",GetLastError());  
            delete array;  
            FileClose(file_handle);  
            //---  
            return;  
        }  
        FileClose(file_handle);  
    }
```

```
    }  
    delete array;  
}
```

Load

Loads data array from a file.

```
virtual bool Load(
    int file_handle    // File handle
)
```

Parameters

file_handle

[in] Handle to open earlier, with the function `FileOpen (...)`, binary file.

Return Value

true - if successfully completed, false - if an error.

Example:

```
//--- example for CArrayShort::Load(int)
#include <Arrays\ArrayShort.mqh>
//---
void OnStart()
{
    int file_handle;
    CArrayShort *array=new CArrayShort;
    //---
    if(array!=NULL)
    {
        printf("Object create error");
        return;
    }
    //--- open file
    file_handle=FileOpen("MyFile.bin",FILE_READ|FILE_BIN|FILE_ANSI);
    if(file_handle>=0)
    {
        if(!array.Load(file_handle))
        {
            //--- file load error
            printf("File load: Error %d!",GetLastError());
            delete array;
            FileClose(file_handle);
            //---
            return;
        }
        FileClose(file_handle);
    }
    //--- use arrays elements
    for(int i=0;i<array.Total();i++)
    {
        printf("Element[%d] = %d",i,array.At(i));
    }
}
```

```
    }  
    delete array;  
}
```

Type

Gets the type identifier of the array.

```
virtual int Type() const
```

Return Value

ID type of the array (for CArrayShort - 82) .

Example:

```
//--- example for CArrayShort::Type()
#include <Arrays\ArrayShort.mqh>
//---
void OnStart()
{
    CArrayShort *array=new CArrayShort;
    //---
    if(array==NULL)
    {
        printf("Object create error");
        return;
    }
    //--- get array type
    int type=array.Type();
    //--- delete array
    delete array;
}
```


CArrayInt

CArrayInt class is a class of dynamic array of variables of type int or uint.

Description

Class CArrayInt provides the ability to work with a dynamic array of variables of type int or uint. In the class implemented the ability to add / insert / delete elements in an array, sort array, searching in sorted array. In addition, the implemented methods of work with the file.

Declaration

```
class CArrayInt : public CArray
```

Title

```
#include <Arrays\ArrayInt.mqh>
```

Class Methods

Memory control	
Reserve	Allocates memory to increase the size of the array
Resize	Sets a new (smaller) size of the array
Shutdown	Clears the array with a full memory release
Add methods	
Add	Adds an element to the end of the array
AddArray	Adds to the end of the array elements from another array
AddArray	Adds to the end of the array elements from another array
Insert	Inserts an element in the array to the specified position
InsertArray	Inserts an array of elements from another array with the specified position
InsertArray	Inserts an array of elements from another array with the specified position
AssignArray	Copies the array elements from another array
AssignArray	Copies the array elements from another array
Update methods	
Update	Changes the element at the specified position array
Shift	Moves an item from a given position in the array to the specified offset
Delete methods	
Delete	Removes the element from the specified position array

DeleteRange	Deletes a group of elements from the specified position array
Access methods	
At	Gets the element from the specified position array
Compare methods	
CompareArray	Compares array with another array
CompareArray	Compares array with another array
Sorted array operations	
InsertSort	Inserts element in a sorted array
Search	Searches for an element equal to the model in sorted array
SearchGreat	Searches for an element of more samples in sorted array
SearchLess	Searches for an element less than the sample in the sorted array
SearchGreatOrEqual	Searches for an element greater than or equal to the model in sorted array
SearchLessOrEqual	Searches for an element less than or equal to the model in sorted array
SearchFirst	Finds the first element equal to the model in sorted array
SearchLast	Finds the last element equal to the model in sorted array
Input/output	
virtual Save	Saves data array in the file
virtual Load	Loads data array from a file
virtual Type	Gets the type identifier of the array

Reserve

Allocates memory to increase the size of the array.

```
bool Reserve(  
    int size      // Number  
)
```

Parameters

size

[in] The number of additional elements of the array.

Return Value

true if successful, false - if there was an attempt to seek the amount is less than or equal to zero, or if the array did not increase.

Note

To reduce memory fragmentation, increase the size of the array is made with a step previously given through the method of Step (int) , or 16 (default) .

Example:

```
//--- example for CArrayInt::Reserve(int)  
#include <Arrays\ArrayInt.mqh>  
//---  
void OnStart()  
{  
    CArrayInt *array=new CArrayInt;  
    //---  
    if(array==NULL)  
    {  
        printf("Object create error");  
        return;  
    }  
    //--- reserve memory  
    if(!array.Reserve(1024))  
    {  
        printf("Reserve error");  
        delete array;  
        return;  
    }  
    //--- use array  
    //--- . . .  
    //--- delete array  
    delete array;  
}
```

Resize

Sets a new (smaller) size of the array.

```
bool Resize(  
    int size      // Number  
)
```

Parameters

size

[in] New size of the array.

Return Value

true if successful, false - if there was an attempt to set the size of less than or equal to zero.

Note

Changing the size of the array allows optimal use of memory. Superfluous elements on the right lost. To reduce fragmentation of memory, change the size of the array is made with a step previously given through the method of Step (int) , or 16 (default) .

Example:

```
//--- example for CArrayInt::Resize(int)  
#include <Arrays\ArrayInt.mqh>  
//---  
void OnStart()  
{  
    CArrayInt *array=new CArrayInt;  
    //---  
    if(array==NULL)  
    {  
        printf("Object create error");  
        return;  
    }  
    //--- add arrays elements  
    //--- . . .  
    //--- resize array  
    if(!array.Resize(10))  
    {  
        printf("Resize error");  
        delete array;  
        return;  
    }  
    //--- delete array  
    delete array;  
}
```

Shutdown

Clears the array with a full memory release.

```
bool Shutdown()
```

Return Value

true if successful, false - if an error occurred.

Example:

```
//--- example for CArrayInt::Shutdown()
#include <Arrays\ArrayInt.mqh>
//---
void OnStart()
{
    CArrayInt *array=new CArrayInt;
    //---
    if(array==NULL)
    {
        printf("Object create error");
        return;
    }
    //--- add arrays elements
    //--- . . .
    //--- shutdown array
    if(!array.Shutdown())
    {
        printf("Shutdown error");
        delete array;
        return;
    }
    //--- delete array
    delete array;
}
```

Add

Adds an element to the end of the array.

```
bool Add(  
    int element    // Element to add  
)
```

Parameters

element

[in] value of the element to add to the array.

Return Value

true if successful, false - if you can not add an element.

Example:

```
//--- example for CArrayInt::Add(int)  
#include <Arrays\ArrayInt.mqh>  
//---  
void OnStart()  
{  
    CArrayInt *array=new CArrayInt;  
    //---  
    if(array==NULL)  
    {  
        printf("Object create error");  
        return;  
    }  
    //--- add arrays elements  
    for(int i=0;i<100;i++)  
    {  
        if(!array.Add(i))  
        {  
            printf("Element addition error");  
            delete array;  
            return;  
        }  
    }  
    //--- use array  
    //--- . . .  
    //--- delete array  
    delete array;  
}
```

AddArray

Adds to the end of the array elements from another array.

```
bool AddArray(  
    const int& src[]    // Source array  
)
```

Parameters

src[]

[in] Reference to an array of source elements to add.

Return Value

true if successful, false - if you can not add items.

Example:

```
///--- example for CArrayInt::AddArray(const int &[])  
#include <Arrays\ArrayInt.mqh>  
///---  
int src[];  
///---  
void OnStart()  
{  
    CArrayInt *array=new CArrayInt;  
    ///---  
    if(array==NULL)  
    {  
        printf("Object create error");  
        return;  
    }  
    ///--- add another array  
    if(!array.AddArray(src))  
    {  
        printf("Array addition error");  
        delete array;  
        return;  
    }  
    ///--- use array  
    ///--- . . .  
    ///--- delete array  
    delete array;  
}
```

AddArray

Adds to the end of the array elements from another array.

```
bool AddArray(  
    const CArrayInt* src      // Pointer to the source  
)
```

Parameters

src

[in] Pointer to an instance of class CArrayInt-source elements to add.

Return Value

true if successful, false - if you can not add items.

Example:

```
//--- example for CArrayInt::AddArray(const CArrayInt*)  
#include <Arrays\ArrayInt.mqh>  
//---  
void OnStart()  
{  
    CArrayInt *array=new CArrayInt;  
    //---  
    if(array==NULL)  
    {  
        printf("Object create error");  
        return;  
    }  
    //--- create source array  
    CArrayInt *src=new CArrayInt;  
    if(src==NULL)  
    {  
        printf("Object create error");  
        delete array;  
        return;  
    }  
    //--- add source arrays elements  
    //--- . . .  
    //--- add another array  
    if(!array.AddArray(src))  
    {  
        printf("Array addition error");  
        delete src;  
        delete array;  
        return;  
    }  
    //--- delete source array  
    delete src;
```



```
//--- use array  
//--- . . .  
//--- delete array  
delete array;  
}
```

Insert

Inserts an element in the array to the specified position.

```
bool Insert(  
    int element,    // Element to insert  
    int pos         // Position  
)
```

Parameters

element
[in] Value of the element to be inserted into an array

pos
[in] Position in the array to insert

Return Value

true if successful, false - if you can not insert the element.

Example:

```
//--- example for CArrayInt::Insert(int,int)  
#include <Arrays\ArrayInt.mqh>  
//---  
void OnStart()  
{  
    CArrayInt *array=new CArrayInt;  
    //---  
    if(array==NULL)  
    {  
        printf("Object create error");  
        return;  
    }  
    //--- insert elements  
    for(int i=0;i<100;i++)  
    {  
        if(!array.Insert(i,0))  
        {  
            printf("Insert error");  
            delete array;  
            return;  
        }  
    }  
    //--- use array  
    //--- . . .  
    //--- delete array  
    delete array;  
}
```

InsertArray

Inserts an array of elements from another array specified position.

```
bool  InsertArray(  
    const int&  src[],      // Source array  
    int         pos        // Position  
)
```

Parameters

src[]

[in] Reference to an array of source elements to insert

pos

[in] Position in the array to insert

Return Value

true if successful, false - if you can not paste items.

Example:

```
//--- example for CArrayInt::InsertArray(const int &[],int)  
#include <Arrays\ArrayInt.mqh>  
//---  
int src[];  
//---  
void OnStart()  
{  
    CArrayInt *array=new CArrayInt;  
    //---  
    if(array==NULL)  
    {  
        printf("Object create error");  
        return;  
    }  
    //--- insert another array  
    if(!array.InsertArray(src,0))  
    {  
        printf("Array inserting error");  
        delete array;  
        return;  
    }  
    //--- use array  
    //--- . . .  
    //--- delete array  
    delete array;  
}
```

InsertArray

Inserts an array of elements from another array specified position.

```
bool  InsertArray(  
    CArrayInt*  src,      // Pointer to the source  
    int         pos       // Position  
)
```

Parameters

src

[in] Pointer to an instance of class CArrayInt-source elements to insert.

pos

[in] Position in the array to insert.

Return Value

true if successful, false - if you can not paste items.

Example:

```
//--- example for CArrayInt::InsertArray(const CArrayInt*,int)  
#include <Arrays\ArrayInt.mqh>  
//---  
void OnStart()  
{  
    CArrayInt *array=new CArrayInt;  
    //---  
    if(array==NULL)  
    {  
        printf("Object create error");  
        return;  
    }  
    //--- create source array  
    CArrayInt *src=new CArrayInt;  
    if(src==NULL)  
    {  
        printf("Object create error");  
        delete array;  
        return;  
    }  
    //--- add source arrays elements  
    //--- . . .  
    //--- insert another array  
    if(!array.InsertArray(src,0))  
    {  
        printf("Array inserting error");  
        delete src;  
        delete array;  
    }  
}
```

```
        return;  
    }  
    //--- delete source array  
    delete src;  
    //--- use array  
    //--- . . .  
    //--- delete array  
    delete array;  
}
```

AssignArray

Copies the array elements from another array.

```
bool AssignArray(  
    const int& src[]    // Source array  
)
```

Parameters

src[]

[in] Reference to an array of source elements to copy.

Return Value

true if successful, false - if you can not copy the items.

Example:

```
//--- example for CArrayInt::AssignArray(const int &[])  
#include <Arrays\ArrayInt.mqh>  
//---  
int src[];  
//---  
void OnStart()  
{  
    CArrayInt *array=new CArrayInt;  
    //---  
    if(array==NULL)  
    {  
        printf("Object create error");  
        return;  
    }  
    //--- assign another array  
    if(!array.AssignArray(src))  
    {  
        printf("Array assigned error");  
        delete array;  
        return;  
    }  
    //--- use array  
    //--- . . .  
    //--- delete array  
    delete array;  
}
```

AssignArray

Copies the array elements from another array.

```
bool AssignArray(  
    const CArrayInt* src      // Pointer to the source  
)
```

Parameters

src

[in] Pointer to an instance of class CArrayInt-source element to copy.

Return Value

true if successful, false - if you can not copy the items.

Example:

```
///--- example for CArrayInt::AssignArray(const CArrayInt*)  
#include <Arrays\ArrayInt.mqh>  
///---  
void OnStart()  
{  
    CArrayInt *array=new CArrayInt;  
    ///---  
    if(array==NULL)  
    {  
        printf("Object create error");  
        return;  
    }  
    ///--- create source array  
    CArrayInt *src =new CArrayInt;  
    if(src==NULL)  
    {  
        printf("Object create error");  
        delete array;  
        return;  
    }  
    ///--- add source arrays elements  
    ///--- . . .  
    ///--- assign another array  
    if(!array.AssignArray(src))  
    {  
        printf("Array assigned error");  
        delete src;  
        delete array;  
        return;  
    }  
}
```

```
//--- arrays is identical
//--- delete source array
delete src;
//--- use array
//--- . . .
//--- delete array
delete array;
}
```


Update

Changes the element at the specified position array.

```
bool Update(  
    int pos,           // Position  
    int element        // Value  
)
```

Parameters

pos

[in] Position of the element in the array to change.

element

[in] New value element

Return Value

true if successful, false - if you can not change the element.

Example:

```
//--- example for CArrayInt::Update(int,int)  
#include <Arrays\ArrayInt.mqh>  
//---  
void OnStart()  
{  
    CArrayInt *array=new CArrayInt;  
    //---  
    if(array==NULL)  
    {  
        printf("Object create error");  
        return;  
    }  
    //--- add arrays elements  
    //--- . . .  
    //--- update element  
    if(!array.Update(0,10000))  
    {  
        printf("Update error");  
        delete array;  
        return;  
    }  
    //--- delete array  
    delete array;  
}
```

Shift

Moves an item from a given position in the array to the specified offset.

```
bool Shift(  
    int pos,          // Position  
    int shift         // Shift  
)
```

Parameters

pos

[in] Position of the moved element array

shift

[in] The value of displacement (both positive and negative) .

Return Value

true if successful, false - if you can not move the item.

Example:

```
//--- example for CArrayInt::Shift(int,int)  
#include <Arrays\ArrayInt.mqh>  
//---  
void OnStart()  
{  
    CArrayInt *array=new CArrayInt;  
    //---  
    if(array==NULL)  
    {  
        printf("Object create error");  
        return;  
    }  
    //--- add arrays elements  
    //--- . . .  
    //--- shift element  
    if(!array.Shift(10,-5))  
    {  
        printf("Shift error");  
        delete array;  
        return;  
    }  
    //--- delete array  
    delete array;  
}
```

Delete

Removes the element from the given position in the array.

```
bool Delete(  
    int pos    // Position  
)
```

Parameters

pos

[in] Position removes the element in the array.

Return Value

true if successful, false - if you can not remove the element.

Example:

```
///--- example for CArrayInt::Delete(int)  
#include <Arrays\ArrayInt.mqh>  
///---  
void OnStart()  
{  
    CArrayInt *array=new CArrayInt;  
    ///---  
    if(array==NULL)  
    {  
        printf("Object create error");  
        return;  
    }  
    ///--- add arrays elements  
    ///--- . . .  
    ///--- delete element  
    if(!array.Delete(0))  
    {  
        printf("Delete error");  
        delete array;  
        return;  
    }  
    ///--- delete array  
    delete array;  
}
```

DeleteRange

Deletes a group of elements from a given position in the array.

```
bool DeleteRange(  
    int from,      // Position of the first element  
    int to         // Position of the last element  
)
```

Parameters

from

[in] Position of the first removes the element in the array.

to

[in] Position of the last deleted element in the array.

Return Value

true if successful, false - if you can not remove elements.

Example:

```
//--- example for CArrayInt::DeleteRange(int,int)  
#include <Arrays\ArrayInt.mqh>  
//---  
void OnStart()  
{  
    CArrayInt *array=new CArrayInt;  
    //---  
    if(array==NULL)  
    {  
        printf("Object create error");  
        return;  
    }  
    //--- add arrays elements  
    //--- . . .  
    //--- delete elements  
    if(!array.DeleteRange(0,10))  
    {  
        printf("Delete error");  
        delete array;  
        return;  
    }  
    //--- delete array  
    delete array;  
}
```

At

Gets the element from the given position in the array.

```
int At(  
    int pos    // Position  
) const
```

Parameters

pos

[in] Position of the desired element in the array.

Return Value

The value of the element in case of success, INT_MAX-if there was an attempt to get an element of not existing positions (the last error ERR_OUT_OF_RANGE) .

Note

Of course, INT_MAX could be a valid value of an array element, so having a value, always check the last error code.

Example:

```
//--- example for CArrayInt::At(int)  
#include <Arrays\ArrayInt.mqh>  
//---  
void OnStart()  
{  
    CArrayInt *array=new CArrayInt;  
    //---  
    if(array==NULL)  
    {  
        printf("Object create error");  
        return;  
    }  
    //--- add arrays elements  
    //--- . . .  
    for(int i=0;i<array.Total();i++)  
    {  
        int result=array.At(i);  
        if(result==INT_MAX && GetLastError()==ERR_OUT_OF_RANGE)  
        {  
            //--- error of reading from array  
            printf("Get element error");  
            delete array;  
            return;  
        }  
        //--- use element  
        //--- . . .  
    }  
}
```

```
//--- delete array  
delete array;  
}
```

CompareArray

Compares array with another array.

```
bool CompareArray(  
    const int& src[]    // Source array  
    ) const
```

Parameters

src[]

[in] Reference to an array of source elements for comparison.

Return Value

true if arrays are equal, false - if not.

Example:

```
//--- example for CArrayInt::CompareArray(const int &[])  
#include <Arrays\ArrayInt.mqh>  
//---  
int src[];  
//---  
void OnStart()  
{  
    CArrayInt *array=new CArrayInt;  
    //---  
    if(array==NULL)  
    {  
        printf("Object create error");  
        return;  
    }  
    //--- compare with another array  
    int result=array.CompareArray(src);  
    //--- delete array  
    delete array;  
}
```

CompareArray

Compares array with another array.

```
bool CompareArray(  
    const CArrayInt* src      // Pointer to the source  
    ) const
```

Parameters

src

[in] Pointer to an instance of class CArrayInt-source elements for comparison.

Return Value

true if arrays are equal, false - if not.

Example:

```
//--- example for CArrayInt::CompareArray(const CArrayInt*)  
#include <Arrays\ArrayInt.mqh>  
//---  
void OnStart()  
{  
    CArrayInt *array=new CArrayInt;  
    //---  
    if(array==NULL)  
    {  
        printf("Object create error");  
        return;  
    }  
    //--- create source array  
    CArrayInt *src=new CArrayInt;  
    if(src==NULL)  
    {  
        printf("Object create error");  
        delete array;  
        return;  
    }  
    //--- add source arrays elements  
    //--- . . .  
    //--- compare with another array  
    int result=array.CompareArray(src);  
    //--- delete arrays  
    delete src;  
    delete array;  
}
```


InsertSort

Inserts element in a sorted array.

```
bool InsertSort(  
    int element    // Element to insert  
)
```

Parameters

element

[in] value of the element to be inserted into a sorted array

Return Value

true if successful, false - if you can not insert the element.

Example:

```
//--- example for CArrayInt::InsertSort(int)  
#include <Arrays\ArrayInt.mqh>  
//---  
void OnStart()  
{  
    CArrayInt *array=new CArrayInt;  
    //---  
    if(array==NULL)  
    {  
        printf("Object create error");  
        return;  
    }  
    //--- add arrays elements  
    //--- . . .  
    //--- sort array  
    array.Sort();  
    //--- insert element  
    if(!array.InsertSort(10000))  
    {  
        printf("Insert error");  
        delete array;  
        return;  
    }  
    //--- delete array  
    delete array;  
}
```

Search

Searches for an element equal to the sample in the sorted array.

```
int Search(  
    int element    // Sample  
    ) const
```

Parameters

element

[in] The sample element to search in the array.

Return Value

The position of the found element, if successful, -1 - if the item was not found.

Example:

```
//--- example for CArrayInt::Search(int)  
#include <Arrays\ArrayInt.mqh>  
//---  
void OnStart()  
{  
    CArrayInt *array=new CArrayInt;  
    //---  
    if(array==NULL)  
    {  
        printf("Object create error");  
        return;  
    }  
    //--- add arrays elements  
    //--- . . .  
    //--- sort array  
    array.Sort();  
    //--- search element  
    if(array.Search(10000)!=-1) printf("Element found");  
    else                        printf("Element not found");  
    //--- delete array  
    delete array;  
}
```

SearchGreat

Searches for an element of more samples in sorted array.

```
int SearchGreat(  
    int element    // Sample  
    ) const
```

Parameters

element

[in] The sample element to search in the array.

Return Value

The position of the found element, if successful, -1 - if the item was not found.

Example:

```
//--- example for CArrayInt::SearchGreat(int)  
#include <Arrays\ArrayInt.mqh>  
//---  
void OnStart()  
{  
    CArrayInt *array=new CArrayInt;  
    //---  
    if(array==NULL)  
    {  
        printf("Object create error");  
        return;  
    }  
    //--- add arrays elements  
    //--- . . .  
    //--- sort array  
    array.Sort();  
    //--- search element  
    if(array.SearchGreat(10000)!=-1) printf("Element found");  
    else                             printf("Element not found");  
    //--- delete array  
    delete array;  
}
```

SearchLess

Searches for an element less than the sample in the sorted array.

```
int SearchLess(  
    int element    // Sample  
    ) const
```

Parameters

element

[in] The sample element to search in the array.

Return Value

The position of the found element, if successful, -1 - if the item was not found.

Example:

```
//--- example for CArrayInt::SearchLess(int)  
#include <Arrays\ArrayInt.mqh>  
//---  
void OnStart()  
{  
    CArrayInt *array=new CArrayInt;  
    //---  
    if(array==NULL)  
    {  
        printf("Object create error");  
        return;  
    }  
    //--- add arrays elements  
    //--- . . .  
    //--- sort array  
    array.Sort();  
    //--- search element  
    if(array.SearchLess(10000)!=-1) printf("Element found");  
    else                             printf("Element not found");  
    //--- delete array  
    delete array;  
}
```

SearchGreatOrEqual

Searches for an element greater than or equal to the sample in the sorted array.

```
int SearchGreatOrEqual(  
    int element    // Element to search  
) const
```

Parameters

element

[in] The sample element to search in the array.

Return Value

The position of the found element, if successful, -1 - if the item was not found.

Example:

```
//--- example for CArrayInt::SearchGreatOrEqual(int)  
#include <Arrays\ArrayInt.mqh>  
//---  
void OnStart()  
{  
    CArrayInt *array=new CArrayInt;  
    //---  
    if(array==NULL)  
    {  
        printf("Object create error");  
        return;  
    }  
    //--- add arrays elements  
    //--- . . .  
    //--- sort array  
    array.Sort();  
    //--- search element  
    if(array.SearchGreatOrEqual(10000)!=-1) printf("Element found");  
    else                                     printf("Element not found");  
    //--- delete array  
    delete array;  
}
```

SearchLessOrEqual

Searches for an element less than or equal to the sample in the sorted array.

```
int SearchLessOrEqual(  
    int element    // Sample  
) const
```

Parameters

element

[in] The sample element to search in the array.

Return Value

The position of the found element, if successful, -1 - if the item was not found.

Example:

```
//--- example for CArrayInt::SearchLessOrEqual(int)  
#include <Arrays\ArrayInt.mqh>  
//---  
void OnStart()  
{  
    CArrayInt *array=new CArrayInt;  
    //---  
    if(array==NULL)  
    {  
        printf("Object create error");  
        return;  
    }  
    //--- add arrays elements  
    //--- . . .  
    //--- sort array  
    array.Sort();  
    //--- search element  
    if(array.SearchLessOrEqual(10000)!=-1) printf("Element found");  
    else                                printf("Element not found");  
    //--- delete array  
    delete array;  
}
```

SearchFirst

Finds the first element equal to the sample in the sorted array.

```
int SearchFirst(  
    int element    // Sample  
    ) const
```

Parameters

element

[in] The sample element to search in the array.

Return Value

The position of the found element, if successful, -1 - if the item was not found.

Example:

```
//--- example for CArrayInt:: SearchFirst(int)  
#include <Arrays\ArrayInt.mqh>  
//---  
void OnStart()  
{  
    CArrayInt *array=new CArrayInt;  
    //---  
    if(array==NULL)  
    {  
        printf("Object create error");  
        return;  
    }  
    //--- add arrays elements  
    //--- . . .  
    //--- sort array  
    array.Sort();  
    //--- search element  
    if(array.SearchFirst(10000)!=-1) printf("Element found");  
    else                             printf("Element not found");  
    //--- delete array  
    delete array;  
}
```

SearchLast

Finds the last element equal to the model in sorted array.

```
int SearchLast(  
    int element    // Sample  
    ) const
```

Parameters

element

[in] The sample element to search in the array.

Return Value

The position of the found element, if successful, -1 - if the item was not found.

Example:

```
//--- example for CArrayInt::SearchLast(int)  
#include <Arrays\ArrayInt.mqh>  
//---  
void OnStart()  
{  
    CArrayInt *array=new CArrayInt;  
    //---  
    if(array==NULL)  
    {  
        printf("Object create error");  
        return;  
    }  
    //--- add arrays elements  
    //--- . . .  
    //--- sort array  
    array.Sort();  
    //--- search element  
    if(array.SearchLast(10000)!=-1) printf("Element found");  
    else                           printf("Element not found");  
    //--- delete array  
    delete array;  
}
```


Save

Saves data array in the file.

```
virtual bool Save(  
    int file_handle    // File handle  
)
```

Parameters

file_handle

[in] Handle to open earlier, with the function `FileOpen (...)`, binary file.

Return Value

true - if successfully completed, false - if an error.

Example:

```
//--- example for CArrayInt::Save(int)  
#include <Arrays\ArrayInt.mqh>  
//---  
void OnStart()  
{  
    int file_handle;  
    CArrayInt *array=new CArrayInt;  
    //---  
    if(array!=NULL)  
    {  
        printf("Object create error");  
        return;  
    }  
    //--- add 100 arrays elements  
    for(int i=0;i<100;i++)  
    {  
        array.Add(i);  
    }  
    //--- open file  
    file_handle=FileOpen("MyFile.bin", FILE_WRITE|FILE_BIN|FILE_ANSI);  
    if(file_handle>=0)  
    {  
        if(!array.Save(file_handle))  
        {  
            //--- file save error  
            printf("File save: Error %d!",GetLastError());  
            delete array;  
            FileClose(file_handle);  
            //---  
            return;  
        }  
        FileClose(file_handle);  
    }
```

```
    }  
    delete array;  
}
```

Load

Loads data array from a file.

```
virtual bool Load(
    int file_handle    // File handle
)
```

Parameters

file_handle

[in] Handle to open earlier, with the function `FileOpen (...)`, binary file.

Return Value

true - if successfully completed, false - if an error.

Example:

```
//--- example for CArrayInt::Load(int)
#include <Arrays\ArrayInt.mqh>
//---
void OnStart()
{
    int file_handle;
    CArrayInt *array=new CArrayInt;
    //---
    if(array!=NULL)
    {
        printf("Object create error");
        return;
    }
    //--- open file
    file_handle=FileOpen("MyFile.bin",FILE_READ|FILE_BIN|FILE_ANSI);
    if(file_handle>=0)
    {
        if(!array.Load(file_handle))
        {
            //--- file load error
            printf("File load: Error %d!",GetLastError());
            delete array;
            FileClose(file_handle);
            //---
            return;
        }
        FileClose(file_handle);
    }
    //--- use arrays elements
    for(int i=0;i<array.Total();i++)
    {
        printf("Element[%d] = %d",i,array.At(i));
    }
}
```

```
    }  
    delete array;  
}
```

Type

Gets the type identifier of the array.

```
virtual int Type() const
```

Return Value

ID type of the array (for CArrayInt - 82) .

Example:

```
//--- example for CArrayInt::Type()
#include <Arrays\ArrayInt.mqh>
//---
void OnStart()
{
    CArrayInt *array=new CArrayInt;
    //---
    if(array==NULL)
    {
        printf("Object create error");
        return;
    }
    //--- get array type
    int type=array.Type();
    //--- delete array
    delete array;
}
```

CArrayLong

CArrayLong class is a class of dynamic array of variables of type long or ulong.

Description

Class CArrayLong provides the ability to work with a dynamic array of variables of type long or ulong. In the class implemented the ability to add / insert / delete elements in an array, sort array, searching in sorted array. In addition, the implemented methods of work with the file.

Declaration

```
class CArrayLong : public CArray
```

Title

```
#include <Arrays\ArrayLong.mqh>
```

Class Methods

Memory control	
Reserve	Allocates memory to increase the size of the array
Resize	Sets a new (smaller) size of the array
Shutdown	Clears the array with a full memory release
Add methods	
Add	Adds an element to the end of the array
AddArray	Adds to the end of the array elements from another array
AddArray	Adds to the end of the array elements from another array
Insert	Inserts an element in the array to the specified position
InsertArray	Inserts an array of elements from another array with the specified position
InsertArray	Inserts an array of elements from another array with the specified position
AssignArray	Copies the array elements from another array
AssignArray	Copies the array elements from another array
Update methods	
Update	Changes the element at the specified position array
Shift	Moves an item from a given position in the array to the specified offset
Delete methods	
Delete	Removes the element from the specified position array

DeleteRange	Deletes a group of elements from the specified position array
Access methods	
At	Gets the element from the specified position array
Compare methods	
CompareArray	Compares array with another array
CompareArray	Compares array with another array
Sorted array operations	
InsertSort	Inserts element in a sorted array
Search	Searches for an element equal to the model in sorted array
SearchGreat	Searches for an element of more samples in sorted array
SearchLess	Searches for an element less than the sample in the sorted array
SearchGreatOrEqual	Searches for an element greater than or equal to the model in sorted array
SearchLessOrEqual	Searches for an element less than or equal to the model in sorted array
SearchFirst	Finds the first element equal to the model in sorted array
SearchLast	Finds the last element equal to the model in sorted array
Input/output	
virtual Save	Saves data array in the file
virtual Load	Loads data array from a file
virtual Type	Gets the type identifier of the array

Reserve

Allocates memory to increase the size of the array.

```
bool Reserve(  
    int size      // Number  
)
```

Parameters

size

[in] The number of additional elements of the array.

Return Value

true if successful, false - if there was an attempt to seek the amount is less than or equal to zero, or if the array did not increase.

Note

To reduce memory fragmentation, increase the size of the array is made with a step previously given through the method of Step (int) , or 16 (default) .

Example:

```
//--- example for CArrayLong::Reserve(int)  
#include <Arrays\ArrayLong.mqh>  
//---  
void OnStart()  
{  
    CArrayLong *array=new CArrayLong;  
    //---  
    if(array==NULL)  
    {  
        printf("Object create error");  
        return;  
    }  
    //--- reserve memory  
    if(!array.Reserve(1024))  
    {  
        printf("Reserve error");  
        delete array;  
        return;  
    }  
    //--- use array  
    //--- . . .  
    //--- delete array  
    delete array;  
}
```


Resize

Sets a new (smaller) size of the array.

```
bool Resize(  
    int size      // Size  
)
```

Parameters

size

[in] New size of the array.

Return Value

true if successful, false - if there was an attempt to set the size of less than or equal to zero.

Note

Changing the size of the array allows optimal use of memory. Superfluous elements on the right lost. To reduce fragmentation of memory, change the size of the array is made with a step previously given through the method of Step (int) , or 16 (default) .

Example:

```
//--- example for CArrayLong::Resize(int)  
#include <Arrays\ArrayLong.mqh>  
//---  
void OnStart()  
{  
    CArrayLong *array=new CArrayLong;  
    //---  
    if(array==NULL)  
    {  
        printf("Object create error");  
        return;  
    }  
    //--- add arrays elements  
    //--- . . .  
    //--- resize array  
    if(!array.Resize(10))  
    {  
        printf("Resize error");  
        delete array;  
        return;  
    }  
    //--- delete array  
    delete array;  
}
```

Shutdown

Clears the array with a full memory release.

```
bool Shutdown()
```

Return Value

true if successful, false - if an error occurred.

Example:

```
//--- example for CArrayLong::Shutdown()
#include <Arrays\ArrayLong.mqh>
//---
void OnStart()
{
    CArrayLong *array=new CArrayLong;
    //---
    if(array==NULL)
    {
        printf("Object create error");
        return;
    }
    //--- add arrays elements
    //--- . . .
    //--- shutdown array
    if(!array.Shutdown())
    {
        printf("Shutdown error");
        delete array;
        return;
    }
    //--- delete array
    delete array;
}
```

Add

Adds an element to the end of the array.

```
bool Add(  
    long element    // Element to add  
)
```

Parameters

element

[in] Value of the element to add to the array.

Return Value

true if successful, false - if you can not add an element.

Example:

```
//--- example for CArrayLong::Add(long)  
#include <Arrays\ArrayLong.mqh>  
//---  
void OnStart()  
{  
    CArrayLong *array=new CArrayLong;  
    //---  
    if(array==NULL)  
    {  
        printf("Object create error");  
        return;  
    }  
    //--- add arrays elements  
    for(int i=0;i<100;i++)  
    {  
        if(!array.Add(i))  
        {  
            printf("Element addition error");  
            delete array;  
            return;  
        }  
    }  
    //--- use array  
    //--- . . .  
    //--- delete array  
    delete array;  
}
```

AddArray

Adds to the end of the array elements from another array.

```
bool AddArray(  
    const long& src[]      // Source array  
)
```

Parameters

src[]

[in] Reference to an array of source elements to add.

Return Value

true if successful, false - if you can not add items.

Example:

```
///--- example for CArrayLong::AddArray(const long &[])  
#include <Arrays\ArrayLong.mqh>  
///---  
long src[];  
///---  
void OnStart()  
{  
    CArrayLong *array=new CArrayLong;  
    ///---  
    if(array==NULL)  
    {  
        printf("Object create error");  
        return;  
    }  
    ///--- add another array  
    if(!array.AddArray(src))  
    {  
        printf("Array addition error");  
        delete array;  
        return;  
    }  
    ///--- use array  
    ///--- . . .  
    ///--- delete array  
    delete array;  
}
```

AddArray

Adds to the end of the array elements from another array.

```
bool AddArray(  
    const CArrayLong* src      // Pointer to the source  
)
```

Parameters

src

[in] Pointer to an instance of class CArrayLong-source elements to add.

Return Value

true if successful, false - if you can not add items.

Example:

```
//--- example for CArrayLong::AddArray(const CArrayLong*)  
#include <Arrays\ArrayLong.mqh>  
//---  
void OnStart()  
{  
    CArrayLong *array=new CArrayLong;  
    //---  
    if(array==NULL)  
    {  
        printf("Object create error");  
        return;  
    }  
    //--- create source array  
    CArrayLong *src=new CArrayLong;  
    if(src==NULL)  
    {  
        printf("Object create error");  
        delete array;  
        return;  
    }  
    //--- add source arrays elements  
    //--- . . .  
    //--- add another array  
    if(!array.AddArray(src))  
    {  
        printf("Array addition error");  
        delete src;  
        delete array;  
        return;  
    }  
    //--- delete source array  
    delete src;
```

```
//--- use array  
//--- . . .  
//--- delete array  
delete array;  
}
```

Insert

Inserts an element in the array to the specified position.

```
bool Insert(  
    long element,    // Element to insert  
    int pos          // Position  
)
```

Parameters

element

[in] Value of the element to be inserted into an array

pos

[in] Position in the array to insert

Return Value

true if successful, false - if you can not insert the element.

Example:

```
//--- example for CArrayLong::Insert(long,int)  
#include <Arrays\ArrayLong.mqh>  
//---  
void OnStart()  
{  
    CArrayLong *array=new CArrayLong;  
    //---  
    if(array==NULL)  
    {  
        printf("Object create error");  
        return;  
    }  
    //--- insert elements  
    for(int i=0;i<100;i++)  
    {  
        if(!array.Insert(i,0))  
        {  
            printf("Insert error");  
            delete array;  
            return;  
        }  
    }  
    //--- use array  
    //--- . . .  
    //--- delete array  
    delete array;  
}
```

InsertArray

Inserts an array of elements from another array ukazannogy position.

```
bool  InsertArray(  
    const long&  src[],      // Source array  
    int          pos         // Position  
)
```

Parameters

src[]

[in] Reference to an array of source elements to insert

pos

[in] Position in the array to insert

Return Value

true if successful, false - if you can not paste items.

Example:

```
//--- example for CArrayLong::InsertArray(const long &[],int)  
#include <Arrays\ArrayLong.mqh>  
//---  
long src[];  
//---  
void OnStart()  
{  
    CArrayLong *array=new CArrayLong;  
    //---  
    if(array==NULL)  
    {  
        printf("Object create error");  
        return;  
    }  
    //--- insert another array  
    if(!array.InsertArray(src,0))  
    {  
        printf("Array inserting error");  
        delete array;  
        return;  
    }  
    //--- use array  
    //--- . . .  
    //--- delete array  
    delete array;  
}
```


InsertArray

Inserts an array of elements from another array ukazannogy position.

```
bool  InsertArray(  
    CArrayLong*  src,      // Pointer to the source  
    int          pos       // Position  
)
```

Parameters

src

[in] Pointer to an instance of class CArrayLong-source elements to insert.

pos

[in] Position in the array to insert

Return Value

true if successful, false - if you can not paste items.

Example:

```
//--- example for CArrayLong::InsertArray(const CArrayLong*,int)  
#include <Arrays\ArrayLong.mqh>  
//---  
void OnStart()  
{  
    CArrayLong *array=new CArrayLong;  
    //---  
    if(array==NULL)  
    {  
        printf("Object create error");  
        return;  
    }  
    //--- create source array  
    CArrayLong *src=new CArrayLong;  
    if(src==NULL)  
    {  
        printf("Object create error");  
        delete array;  
        return;  
    }  
    //--- add source arrays elements  
    //--- . . .  
    //--- insert another array  
    if(!array.InsertArray(src,0))  
    {  
        printf("Array inserting error");  
        delete src;  
        delete array;  
    }  
}
```

```
        return;  
    }  
    //--- delete source array  
    delete src;  
    //--- use array  
    //--- . . .  
    //--- delete array  
    delete array;  
}
```

AssignArray

Copies the array elements from another array.

```
bool AssignArray(  
    const long& src[]      // Source array  
)
```

Parameters

src[]

[in] Reference to an array of source elements to copy.

Return Value

true if successful, false - if you can not copy the items.

Example:

```
//--- example for CArrayLong::AssignArray(const long &[])  
#include <Arrays\ArrayLong.mqh>  
//---  
long src[];  
//---  
void OnStart()  
{  
    CArrayLong *array=new CArrayLong;  
    //---  
    if(array==NULL)  
    {  
        printf("Object create error");  
        return;  
    }  
    //--- assign another array  
    if(!array.AssignArray(src))  
    {  
        printf("Array assigned error");  
        delete array;  
        return;  
    }  
    //--- use array  
    //--- . . .  
    //--- delete array  
    delete array;  
}
```

AssignArray

Copies the array elements from another array.

```
bool AssignArray(  
    const CArrayLong* src      // Pointer to the source  
)
```

Parameters

src

[in] Pointer to an instance of class CArrayLong-source element to copy.

Return Value

true if successful, false - if you can not copy the items.

Example:

```
//--- example for CArrayLong::AssignArray(const CArrayLong*)  
#include <Arrays\ArrayLong.mqh>  
//---  
void OnStart()  
{  
    CArrayLong *array=new CArrayLong;  
    //---  
    if(array==NULL)  
    {  
        printf("Object create error");  
        return;  
    }  
    //--- create source array  
    CArrayLong *src =new CArrayLong;  
    if(src==NULL)  
    {  
        printf("Object create error");  
        delete array;  
        return;  
    }  
    //--- add source arrays elements  
    //--- . . .  
    //--- assign another array  
    if(!array.AssignArray(src))  
    {  
        printf("Array assigned error");  
        delete src;  
        delete array;  
        return;  
    }  
    //--- arrays is identical  
    //--- delete source array
```

```
delete src;  
//--- use array  
//--- . . .  
//--- delete array  
delete array;  
}
```

Update

Changes the element at the specified position array.

```
bool Update(  
    int    pos,           // Position  
    long   element       // Value  
)
```

Parameters

pos

[in] Position of the element in the array to change

element

[in] New value element

Return Value

true if successful, false - if you can not change the element.

Example:

```
//--- example for CArrayLong::Update(int,long)  
#include <Arrays\ArrayLong.mqh>  
//---  
void OnStart()  
{  
    CArrayLong *array=new CArrayLong;  
    //---  
    if(array==NULL)  
    {  
        printf("Object create error");  
        return;  
    }  
    //--- add arrays elements  
    //--- . . .  
    //--- update element  
    if(!array.Update(0,1000000))  
    {  
        printf("Update error");  
        delete array;  
        return;  
    }  
    //--- delete array  
    delete array;  
}
```

Shift

Moves an item from a given position in the array to the specified offset.

```
bool Shift(  
    int pos,          // Position  
    int shift         // Shift  
)
```

Parameters

pos

[in] Position of the moved element array

shift

[in] The value of displacement (both positive and negative) .

Return Value

true if successful, false - if you can not move the item.

Example:

```
//--- example for CArrayLong::Shift(int,int)  
#include <Arrays\ArrayLong.mqh>  
//---  
void OnStart()  
{  
    CArrayLong *array=new CArrayLong;  
    //---  
    if(array==NULL)  
    {  
        printf("Object create error");  
        return;  
    }  
    //--- add arrays elements  
    //--- . . .  
    //--- shift element  
    if(!array.Shift(10,-5))  
    {  
        printf("Shift error");  
        delete array;  
        return;  
    }  
    //--- delete array  
    delete array;  
}
```

Delete

Removes the element from the given position in the array.

```
bool Delete(  
    int pos    // Position  
)
```

Parameters

pos

[in] Position removes the element in the array.

Return Value

true if successful, false - if you can not remove the element.

Example:

```
//--- example for CArrayLong::Delete(int)  
#include <Arrays\ArrayLong.mqh>  
//---  
void OnStart()  
{  
    CArrayLong *array=new CArrayLong;  
    //---  
    if(array==NULL)  
    {  
        printf("Object create error");  
        return;  
    }  
    //--- add arrays elements  
    //--- . . .  
    //--- delete element  
    if(!array.Delete(0))  
    {  
        printf("Delete error");  
        delete array;  
        return;  
    }  
    //--- delete array  
    delete array;  
}
```


DeleteRange

Deletes a group of elements from a given position in the array.

```
bool DeleteRange(  
    int from,      // Position of the first element  
    int to         // Position of the last element  
)
```

Parameters

from

[in] Position of the first removes the element in the array.

to

[in] Position of the last deleted element in the array.

Return Value

true if successful, false - if you can not remove elements.

Example:

```
//--- example for CArrayLong::DeleteRange(int,int)  
#include <Arrays\ArrayLong.mqh>  
//---  
void OnStart()  
{  
    CArrayLong *array=new CArrayLong;  
    //---  
    if(array==NULL)  
    {  
        printf("Object create error");  
        return;  
    }  
    //--- add arrays elements  
    //--- . . .  
    //--- delete elements  
    if(!array.DeleteRange(0,10))  
    {  
        printf("Delete error");  
        delete array;  
        return;  
    }  
    //--- delete array  
    delete array;  
}
```

At

Gets the element from the given position in the array.

```
long At(
    int pos    // Position
) const
```

Parameters

pos

[in] Position of the desired element in the array.

Return Value

The value of the element in case of success, LONG_MAX if there was an attempt to get an element of not existing positions (the last error ERR_OUT_OF_RANGE) .

Note

Of course, LONG_MAX may be a valid value of an array element, so having a value, always check the last error code.

Example:

```
//--- example for CArrayLong::At(int)
#include <Arrays\ArrayLong.mqh>
//---
void OnStart()
{
    CArrayLong *array=new CArrayLong;
    //---
    if(array==NULL)
    {
        printf("Object create error");
        return;
    }
    //--- add arrays elements
    //--- . . .
    for(int i=0;i<array.Total();i++)
    {
        long result=array.At(i);
        if(result==LONG_MAX && GetLastError()==ERR_OUT_OF_RANGE)
        {
            //--- Error reading from the array
            printf("Get element error");
            delete array;
            return;
        }
        //--- use element
        //--- . . .
    }
}
```

```
//--- delete array  
delete array;  
}
```

CompareArray

Compares array with another array.

```
bool CompareArray(  
    const long& src[]      // Source array  
    ) const
```

Parameters

src[]

[in] Reference to an array of source elements for comparison.

Return Value

true if arrays are equal, false - if not.

Example:

```
//--- example for CArrayLong::CompareArray(const long &[])  
#include <Arrays\ArrayLong.mqh>  
//---  
long src[];  
//---  
void OnStart()  
{  
    CArrayLong *array=new CArrayLong;  
    //---  
    if(array==NULL)  
    {  
        printf("Object create error");  
        return;  
    }  
    //--- compare with another array  
    int result=array.CompareArray(src);  
    //--- delete array  
    delete array;  
}
```

CompareArrayconst

Compares array with another array.

```
bool CompareArrayconst(  
    const CArrayLong* src      // Pointer to the source  
    ) const
```

Parameters

src

[in] Pointer to an instance of class CArrayLong-source elements for comparison.

Return Value

true if arrays are equal, false - if not.

Example:

```
///--- example for CArrayLong::CompareArray(const CArrayLong*)  
#include <Arrays\ArrayLong.mqh>  
///---  
void OnStart()  
{  
    CArrayLong *array=new CArrayLong;  
    ///---  
    if(array==NULL)  
    {  
        printf("Object create error");  
        return;  
    }  
    ///--- create source array  
    CArrayLong *src=new CArrayLong;  
    if(src==NULL)  
    {  
        printf("Object create error");  
        delete array;  
        return;  
    }  
    ///--- add source arrays elements  
    ///--- . . .  
    ///--- compare with another array  
    int result=array.CompareArray(src);  
    ///--- delete arrays  
    delete src;  
    delete array;  
}
```

InsertSort

Inserts element in a sorted array.

```
bool InsertSort(  
    long element    // Element to insert  
)
```

Parameters

element

[in] Value of the element to be inserted into a sorted array

Return Value

true if successful, false - if you can not insert the element.

Example:

```
///--- example for CArrayLong::InsertSort(long)  
#include <Arrays\ArrayLong.mqh>  
///---  
void OnStart()  
{  
    CArrayLong *array=new CArrayLong;  
    ///---  
    if(array==NULL)  
    {  
        printf("Object create error");  
        return;  
    }  
    ///--- add arrays elements  
    ///--- . . .  
    ///--- sort array  
    array.Sort();  
    ///--- insert element  
    if(!array.InsertSort(1000000))  
    {  
        printf("Insert error");  
        delete array;  
        return;  
    }  
    ///--- delete array  
    delete array;  
}
```

Search

Searches for an element equal to the model in sorted array.

```
int Search(  
    long element    // Sample  
    ) const
```

Parameters

element

[in] The sample element to search in the array.

Return Value

The position of the found element, if successful, -1 - if the item was not found.

Example:

```
//--- example for CArrayLong::Search(long)  
#include <Arrays\ArrayLong.mqh>  
//---  
void OnStart()  
{  
    CArrayLong *array=new CArrayLong;  
    //---  
    if(array==NULL)  
    {  
        printf("Object create error");  
        return;  
    }  
    //--- add arrays elements  
    //--- . . .  
    //--- sort array  
    array.Sort();  
    //--- search element  
    if(array.Search(1000000)!=-1) printf("Element found");  
    else                          printf("Element not found");  
    //--- delete array  
    delete array;  
}
```

SearchGreat

Searches for an element of larger sample, in sorted array.

```
int SearchGreat(  
    long element    // Sample  
    ) const
```

Parameters

element

[in] The sample element to search in the array.

Return Value

The position of the found element, if successful, -1 - if the item was not found.

Example:

```
//--- example for CArrayLong::SearchGreat(long)  
#include <Arrays\ArrayLong.mqh>  
//---  
void OnStart()  
{  
    CArrayLong *array=new CArrayLong;  
    //---  
    if(array==NULL)  
    {  
        printf("Object create error");  
        return;  
    }  
    //--- add arrays elements  
    //--- . . .  
    //--- sort array  
    array.Sort();  
    //--- search element  
    if(array.SearchGreat(1000000)!=-1) printf("Element found");  
    else                             printf("Element not found");  
    //--- delete array  
    delete array;  
}
```


SearchLess

Searches for an element less than the sample in the sorted array.

```
int SearchLess(  
    long element    // Sample  
    ) const
```

Parameters

element

[in] The sample element to search in the array.

Return Value

The position of the found element, if successful, -1 - if the item was not found.

Example:

```
//--- example for CArrayLong::SearchLess(long)  
#include <Arrays\ArrayLong.mqh>  
//---  
void OnStart()  
{  
    CArrayLong *array=new CArrayLong;  
    //---  
    if(array==NULL)  
    {  
        printf("Object create error");  
        return;  
    }  
    //--- add arrays elements  
    //--- . . .  
    //--- sort array  
    array.Sort();  
    //--- search element  
    if(array.SearchLess(1000000)!=-1) printf("Element found");  
    else                             printf("Element not found");  
    //--- delete array  
    delete array;  
}
```

SearchGreatOrEqual

Searches for an element greater than or equal to the model in sorted array.

```
int SearchGreatOrEqual(  
    long element    // Sample  
) const
```

Parameters

element

[in] The sample element to search in the array.

Return Value

The position of the found element, if successful, -1 - if the item was not found.

Example:

```
//--- example for CArrayLong::SearchGreatOrEqual(long)  
#include <Arrays\ArrayLong.mqh>  
//---  
void OnStart()  
{  
    CArrayLong *array=new CArrayLong;  
    //---  
    if(array==NULL)  
    {  
        printf("Object create error");  
        return;  
    }  
    //--- add arrays elements  
    //--- . . .  
    //--- sort array  
    array.Sort();  
    //--- search element  
    if(array.SearchGreatOrEqual(1000000)!=-1) printf("Element found");  
    else                                     printf("Element not found");  
    //--- delete array  
    delete array;  
}
```

SearchLessOrEqual

Searches for an element less than or equal to the model in sorted array.

```
int SearchLessOrEqual(  
    long element    // Sample  
    ) const
```

Parameters

element

[in] The sample element to search in the array.

Return Value

The position of the found element, if successful, -1 - if the item was not found.

Example:

```
//--- example for CArrayLong::SearchLessOrEqual(long)  
#include <Arrays\ArrayLong.mqh>  
//---  
void OnStart()  
{  
    CArrayLong *array=new CArrayLong;  
    //---  
    if(array==NULL)  
    {  
        printf("Object create error");  
        return;  
    }  
    //--- add arrays elements  
    //--- . . .  
    //--- sort array  
    array.Sort();  
    //--- search element  
    if(array.SearchLessOrEqual(1000000)!=-1) printf("Element found");  
    else                                     printf("Element not found");  
    //--- delete array  
    delete array;  
}
```

SearchFirst

Finds the first element equal to the model in sorted array.

```
int SearchFirst(  
    long element    // Sample  
    ) const
```

Parameters

element

[in] The sample element to search in the array.

Return Value

The position of the found element, if successful, -1 - if the item was not found.

Example:

```
//--- example for CArrayLong::SearchFirst(long)  
#include <Arrays\ArrayLong.mqh>  
//---  
void OnStart()  
{  
    CArrayLong *array=new CArrayLong;  
    //---  
    if(array==NULL)  
    {  
        printf("Object create error");  
        return;  
    }  
    //--- add arrays elements  
    //--- . . .  
    //--- sort array  
    array.Sort();  
    //--- search element  
    if(array.SearchFirst(1000000)!=-1) printf("Element found");  
    else                             printf("Element not found");  
    //--- delete array  
    delete array;  
}
```

SearchLast

Finds the last element equal to the model in sorted array.

```
int SearchLast(  
    long element    // Sample  
    ) const
```

Parameters

element

[in] The sample element to search in the array.

Return Value

The position of the found element, if successful, -1 - if the item was not found.

Example:

```
//--- example for CArrayLong::SearchLast(long)  
#include <Arrays\ArrayLong.mqh>  
//---  
void OnStart()  
{  
    CArrayLong *array=new CArrayLong;  
    //---  
    if(array==NULL)  
    {  
        printf("Object create error");  
        return;  
    }  
    //--- add arrays elements  
    //--- . . .  
    //--- sort array  
    array.Sort();  
    //--- search element  
    if(array.SearchLast(1000000)!=-1) printf("Element found");  
    else                             printf("Element not found");  
    //--- delete array  
    delete array;  
}
```

Save

Saves data array in the file.

```
virtual bool Save(  
    int file_handle    // File handle  
)
```

Parameters

file_handle

[in] Handle to open earlier, with the function `FileOpen (...)`, binary file.

Return Value

true - if successfully completed, false - if an error.

Example:

```
//--- example for CArrayLong::Save(int)  
#include <Arrays\ArrayLong.mqh>  
//---  
void OnStart()  
{  
    int file_handle;  
    CArrayLong *array=new CArrayLong;  
    //---  
    if(array!=NULL)  
    {  
        printf("Object create error");  
        return;  
    }  
    //--- add 100 arrays elements  
    for(int i=0;i<100;i++)  
    {  
        array.Add(i);  
    }  
    //--- open file  
    file_handle=FileOpen("MyFile.bin", FILE_WRITE|FILE_BIN|FILE_ANSI);  
    if(file_handle>=0)  
    {  
        if(!array.Save(file_handle))  
        {  
            //--- file save error  
            printf("File save: Error %d!",GetLastError());  
            delete array;  
            FileClose(file_handle);  
            //---  
            return;  
        }  
        FileClose(file_handle);  
    }
```

```
    }  
    delete array;  
}
```

Load

Loads data array from a file.

```
virtual bool Load(
    int file_handle    // File handle
)
```

Parameters

file_handle

[in] Handle to open earlier, with the function `FileOpen (...)`, binary file.

Return Value

true - if successfully completed, false - if an error.

Example:

```
//--- example for CArrayLong::Load(int)
#include <Arrays\ArrayLong.mqh>
//---
void OnStart()
{
    int file_handle;
    CArrayLong *array=new CArrayLong;
    //---
    if(array!=NULL)
    {
        printf("Object create error");
        return;
    }
    //--- open file
    file_handle=FileOpen("MyFile.bin",FILE_READ|FILE_BIN|FILE_ANSI);
    if(file_handle>=0)
    {
        if(!array.Load(file_handle))
        {
            //--- file load error
            printf("File load: Error %d!",GetLastError());
            delete array;
            FileClose(file_handle);
            //---
            return;
        }
        FileClose(file_handle);
    }
    //--- use arrays elements
    for(int i=0;i<array.Total();i++)
    {
        printf("Element[%d] = %I64",i,array.At(i));
    }
}
```



```
    }  
    delete array;  
}
```

Type

Gets the type identifier of the array.

```
virtual int Type() const
```

Return Value

ID type of the array (for CArrayLong - 84) .

Example:

```
//--- example for CArrayLong::Type()
#include <Arrays\ArrayLong.mqh>
//---
void OnStart()
{
    CArrayLong *array=new CArrayLong;
    //---
    if(array==NULL)
    {
        printf("Object create error");
        return;
    }
    //--- get array type
    int type=array.Type();
    //--- delete array
    delete array;
}
```

CArrayFloat

CArrayFloat class is a class of dynamic array of variables of type float.

Description

Class CArrayFloat provides the ability to work with a dynamic array of variables of type float. In the class implemented the ability to add / insert / delete elements in an array, sort array, searching in sorted array. In addition, the implemented methods of work with the file.

Declaration

```
class CArrayFloat : public CArray
```

Title

```
#include <Arrays\ArrayFloat.mqh>
```

Class Methods

Attributes	
Delta	Set the comparison tolerance
Memory control	
Reserve	Allocates memory to increase the size of the array
Resize	Sets a new (smaller) size of the array
Shutdown	Clears the array with a full memory release
Add methods	
Add	Adds an element to the end of the array
AddArray	Adds to the end of the array elements from another array
AddArray	Adds to the end of the array elements from another array
Insert	Inserts an element in the array to the specified position
InsertArray	Inserts an array of elements from another array with the specified position
InsertArray	Inserts an array of elements from another array with the specified position
AssignArray	Copies the array elements from another array
AssignArray	Copies the array elements from another array
Update methods	
Update	Changes the element at the specified position array
Shift	Moves an item from a given position in the array to the specified offset

Delete	Removes the element from the specified position array
DeleteRange	Deletes a group of elements from the specified position array
Access methods	
At	Gets the element from the specified position array
Compare methods	
CompareArray	Compares array with another array
CompareArray	Compares array with another array
Sorted array operations	
InsertSort	Inserts element in a sorted array
Search	Searches for an element equal to the model in sorted array
SearchGreat	Searches for an element of more samples in sorted array
SearchLess	Searches for an element less than the sample in the sorted array
SearchGreatOrEqual	Searches for an element greater than or equal to the model in sorted array
SearchLessOrEqual	Searches for an element less than or equal to the model in sorted array
SearchFirst	Finds the first element equal to the model in sorted array
SearchLast	Finds the last element equal to the model in sorted array
Input/output	
virtual Save	Saves data array in the file
virtual Load	Loads data array from a file
virtual Type	Gets the type identifier of the array

Delta

Set tolerance comparison.

```
void Delta(  
    float delta    // Tolerance  
)
```

Parameters

delta

[in] the new value of the admission of comparison.

Return Value

None

Note

Admission of comparison used in the search. Values are considered equal if their difference is less than or equal to tolerance. The default tolerance is 0.0.

Example:

```
///--- example for CArrayFloat::Delta(float)  
#include <Arrays\ArrayFloat.mqh>  
///---  
void OnStart()  
{  
    CArrayFloat *array=new CArrayFloat;  
    ///---  
    if(array==NULL)  
    {  
        printf("Object create error");  
        return;  
    }  
    ///--- set compare variation  
    array.Delta(0.001);  
    ///--- use array  
    ///--- . . .  
    ///--- delete array  
    delete array;  
}
```

Reserve

Allocates memory to increase the size of the array.

```
bool Reserve(  
    int size      // Number  
)
```

Parameters

size

[in] The number of additional elements of the array.

Return Value

true if successful, false - if there was an attempt to seek the amount is less than or equal to zero, or if the array did not increase.

Note

To reduce memory fragmentation, increase the size of the array is made with a step previously given through the method of Step (int) , or 16 (default) .

Example:

```
//--- example for CArrayFloat::Reserve(int)  
#include <Arrays\ArrayFloat.mqh>  
//---  
void OnStart()  
{  
    CArrayFloat *array=new CArrayFloat;  
    //---  
    if(array==NULL)  
    {  
        printf("Object create error");  
        return;  
    }  
    //--- reserve memory  
    if(!array.Reserve(1024))  
    {  
        printf("Reserve error");  
        delete array;  
        return;  
    }  
    //--- use array  
    //--- . . .  
    //--- delete array  
    delete array;  
}
```

Resize

Sets a new (smaller) size of the array.

```
bool Resize(  
    int size      // Size  
)
```

Parameters

size

[in] New size of the array.

Return Value

true if successful, false - if there was an attempt to set the size of less than or equal to zero.

Note

Changing the size of the array allows optimal use of memory. Superfluous elements on the right lost. To reduce fragmentation of memory, change the size of the array is made with a step previously given through the method of Step (int) , or 16 (default) .

Example:

```
//--- example for CArrayFloat::Resize(int)  
#include <Arrays\ArrayFloat.mqh>  
//---  
void OnStart()  
{  
    CArrayFloat *array=new CArrayFloat;  
    //---  
    if(array==NULL)  
    {  
        printf("Object create error");  
        return;  
    }  
    //--- add arrays elements  
    //--- . . .  
    //--- resize array  
    if(!array.Resize(10))  
    {  
        printf("Resize error");  
        delete array;  
        return;  
    }  
    //--- delete array  
    delete array;  
}
```

Shutdown

Clears the array with a full memory release.

```
bool Shutdown()
```

Return Value

true if successful, false - if an error occurred.

Example:

```
//--- example for CArrayFloat::Shutdown()
#include <Arrays\ArrayFloat.mqh>
//---
void OnStart()
{
    CArrayFloat *array=new CArrayFloat;
    //---
    if(array==NULL)
    {
        printf("Object create error");
        return;
    }
    //--- add arrays elements
    //--- . . .
    //--- shutdown array
    if(!array.Shutdown())
    {
        printf("Shutdown error");
        delete array;
        return;
    }
    //--- delete array
    delete array;
}
```


Add

Adds an element to the end of the array.

```
bool Add(  
    float element    // Element to add  
)
```

Parameters

element

[in] Value of the element to add to the array.

Return Value

true if successful, false - if you can not add an element.

Example:

```
//--- example for CArrayFloat::Add(float)  
#include <Arrays\ArrayFloat.mqh>  
//---  
void OnStart()  
{  
    CArrayFloat *array=new CArrayFloat;  
    //---  
    if(array==NULL)  
    {  
        printf("Object create error");  
        return;  
    }  
    //--- add arrays elements  
    for(int i=0;i<100;i++)  
    {  
        if(!array.Add(i))  
        {  
            printf("Element addition error");  
            delete array;  
            return;  
        }  
    }  
    //--- use array  
    //--- . . .  
    //--- delete array  
    delete array;  
}
```

AddArray

Adds to the end of the array elements from another array.

```
bool AddArray(  
    const float& src[]    // Source array  
)
```

Parameters

src[]

[in] Reference to an array of source elements to add.

Return Value

true if successful, false - if you can not add items.

Example:

```
///--- example for CArrayFloat::AddArray(const float &[])  
#include <Arrays\ArrayFloat.mqh>  
///---  
float src[];  
///---  
void OnStart()  
{  
    CArrayFloat *array=new CArrayFloat;  
    ///---  
    if(array==NULL)  
    {  
        printf("Object create error");  
        return;  
    }  
    ///--- add another array  
    if(!array.AddArray(src))  
    {  
        printf("Array addition error");  
        delete array;  
        return;  
    }  
    ///--- use array  
    ///--- . . .  
    ///--- delete array  
    delete array;  
}
```

AddArray

Adds to the end of the array elements from another array.

```
bool AddArray(  
    const CArrayFloat* src      // Pointer to the source  
)
```

Parameters

src

[in] Pointer to an instance of class CArrayFloat-source elements to add.

Return Value

true if successful, false - if you can not add items.

Example:

```
///--- example for CArrayFloat::AddArray(const CArrayFloat*)  
#include <Arrays\ArrayFloat.mqh>  
///---  
void OnStart()  
{  
    CArrayFloat *array=new CArrayFloat;  
    ///---  
    if(array==NULL)  
    {  
        printf("Object create error");  
        return;  
    }  
    ///--- create source array  
    CArrayFloat *src=new CArrayFloat;  
    if(src==NULL)  
    {  
        printf("Object create error");  
        delete array;  
        return;  
    }  
    ///--- add source arrays elements  
    ///--- . . .  
    ///--- add another array  
    if(!array.AddArray(src))  
    {  
        printf("Array addition error");  
        delete src;  
        delete array;  
        return;  
    }  
    ///--- delete source array  
    delete src;
```

```
//--- use array  
//--- . . .  
//--- delete array  
delete array;  
}
```

Insert

Inserts an element in the array to the specified position.

```
bool Insert(  
    float element,    // Element to insert  
    int pos           // Position  
)
```

Parameters

element

[in] Value of the element to be inserted into an array

pos

[in] Position in the array to insert

Return Value

true if successful, false - if you can not insert the element.

Example:

```
//--- example for CArrayFloat::Insert(float,int)  
#include <Arrays\ArrayFloat.mqh>  
//---  
void OnStart()  
{  
    CArrayFloat *array=new CArrayFloat;  
    //---  
    if(array==NULL)  
    {  
        printf("Object create error");  
        return;  
    }  
    //--- insert elements  
    for(int i=0;i<100;i++)  
    {  
        if(!array.Insert(i,0))  
        {  
            printf("Insert error");  
            delete array;  
            return;  
        }  
    }  
    //--- use array  
    //--- . . .  
    //--- delete array  
    delete array;  
}
```

InsertArray

Inserts an array of elements from another array with the specified position.

```
bool  InsertArray(  
    const float&  src[],      // Source array  
    int           pos         // Position  
)
```

Parameters

src[]

[in] Reference to an array of source elements to insert

pos

[in] Position in the array to insert

Return Value

true if successful, false - if you can not paste items.

Example:

```
//--- example for CArrayFloat::InsertArray(const float &[],int)  
#include <Arrays\ArrayFloat.mqh>  
//---  
float src[];  
//---  
void OnStart()  
{  
    CArrayFloat *array=new CArrayFloat;  
    //---  
    if(array==NULL)  
    {  
        printf("Object create error");  
        return;  
    }  
    //--- insert another array  
    if(!array.InsertArray(src,0))  
    {  
        printf("Array inserting error");  
        delete array;  
        return;  
    }  
    //--- use array  
    //--- . . .  
    //--- delete array  
    delete array;  
}
```

InsertArray

Inserts an array of elements from another array specified position.

```
bool InsertArray(  
    CArrayFloat* src,      // Pointer to the source  
    int pos               // Position  
)
```

Parameters

src

[in] Pointer to an instance of class CArrayFloat-source elements to insert.

pos

[in] Position in the array to insert

Return Value

true if successful, false - if you can not paste items.

Example:

```
//--- example for CArrayFloat::InsertArray(const CArrayFloat*,int)  
#include <Arrays\ArrayFloat.mqh>  
//---  
void OnStart()  
{  
    CArrayFloat *array=new CArrayFloat;  
    //---  
    if(array==NULL)  
    {  
        printf("Object create error");  
        return;  
    }  
    //--- create source array  
    CArrayFloat *src=new CArrayFloat;  
    if(src==NULL)  
    {  
        printf("Object create error");  
        delete array;  
        return;  
    }  
    //--- add source arrays elements  
    //--- . . .  
    //--- insert another array  
    if(!array.InsertArray(src,0))  
    {  
        printf("Array inserting error");  
        delete src;  
        delete array;  
    }  
}
```

```
        return;  
    }  
    //--- delete source array  
    delete src;  
    //--- use array  
    //--- . . .  
    //--- delete array  
    delete array;  
}
```


AssignArray

Copies the array elements from another array.

```
bool AssignArray(  
    const float& src[]    // Source array  
)
```

Parameters

src[]

[in] Reference to an array of source elements to copy.

Return Value

true if successful, false - if you can not copy the items.

Example:

```
//--- example for CArrayFloat::AssignArray(const float &[])  
#include <Arrays\ArrayFloat.mqh>  
//---  
float src[];  
//---  
void OnStart()  
{  
    CArrayFloat *array=new CArrayFloat;  
    //---  
    if(array==NULL)  
    {  
        printf("Object create error");  
        return;  
    }  
    //--- assign another array  
    if(!array.AssignArray(src))  
    {  
        printf("Array assigned error");  
        delete array;  
        return;  
    }  
    //--- use array  
    //--- . . .  
    //--- delete array  
    delete array;  
}
```

AssignArray

Copies the array elements from another array.

```
bool AssignArray(  
    const CArrayFloat* src      // Pointer to the source  
)
```

Parameters

src

[in] Pointer to an instance of class CArrayFloat-source element to copy.

Return Value

true if successful, false - if you can not copy the items.

Example:

```
//--- example for CArrayFloat::AssignArray(const CArrayFloat*)  
#include <Arrays\ArrayFloat.mqh>  
//---  
void OnStart()  
{  
    CArrayFloat *array=new CArrayFloat;  
    //---  
    if(array==NULL)  
    {  
        printf("Object create error");  
        return;  
    }  
    //--- create source array  
    CArrayFloat *src =new CArrayFloat;  
    if(src==NULL)  
    {  
        printf("Object create error");  
        delete array;  
        return;  
    }  
    //--- add source arrays elements  
    //--- . . .  
    //--- assign another array  
    if(!array.AssignArray(src))  
    {  
        printf("Array assigned error");  
        delete src;  
        delete array;  
        return;  
    }  
    //--- arrays is identical  
    //--- delete source array
```

```
delete src;  
//--- use array  
//--- . . .  
//--- delete array  
delete array;  
}
```

Update

Changes the element at the specified position array.

```
bool Update(  
    int    pos,          // Position  
    float  element       // Value  
)
```

Parameters

pos

[in] Position of the element in the array to change

element

[in] New value element

Return Value

true if successful, false - if you can not change the element.

Example:

```
//--- example for CArrayFloat::Update(int,float)  
#include <Arrays\ArrayFloat.mqh>  
//---  
void OnStart()  
{  
    CArrayFloat *array=new CArrayFloat;  
    //---  
    if(array==NULL)  
    {  
        printf("Object create error");  
        return;  
    }  
    //--- add arrays elements  
    //--- . . .  
    //--- update element  
    if(!array.Update(0,100.0))  
    {  
        printf("Update error");  
        delete array;  
        return;  
    }  
    //--- delete array  
    delete array;  
}
```

Shift

Moves an item from a given position in the array to the specified offset.

```
bool Shift(  
    int pos,          // Position  
    int shift         // Shift  
)
```

Parameters

pos

[in] Position of the moved element array

shift

[in] The value of displacement (both positive and negative) .

Return Value

true if successful, false - if you can not move the item.

Example:

```
//--- example for CArrayFloat::Shift(int,int)  
#include <Arrays\ArrayFloat.mqh>  
//---  
void OnStart()  
{  
    CArrayFloat *array=new CArrayFloat;  
    //---  
    if(array==NULL)  
    {  
        printf("Object create error");  
        return;  
    }  
    //--- add arrays elements  
    //--- . . .  
    //--- shift element  
    if(!array.Shift(10,-5))  
    {  
        printf("Shift error");  
        delete array;  
        return;  
    }  
    //--- delete array  
    delete array;  
}
```

Delete

Removes the element from the given position in the array.

```
bool Delete(  
    int pos    // Position  
)
```

Parameters

pos

[in] Position removes the element in the array.

Return Value

true if successful, false - if you can not remove the element.

Example:

```
///--- example for CArrayFloat::Delete(int)  
#include <Arrays\ArrayFloat.mqh>  
///---  
void OnStart()  
{  
    CArrayFloat *array=new CArrayFloat;  
    ///---  
    if(array==NULL)  
    {  
        printf("Object create error");  
        return;  
    }  
    ///--- add arrays elements  
    ///--- . . .  
    ///--- delete element  
    if(!array.Delete(0))  
    {  
        printf("Delete error");  
        delete array;  
        return;  
    }  
    ///--- delete array  
    delete array;  
}
```

DeleteRange

Deletes a group of elements from a given position in the array.

```
bool DeleteRange(  
    int from,      // Position of the first element  
    int to         // Position of last element  
)
```

Parameters

from

[in] Position of the first removes the element in the array.

to

[in] Position of the last deleted element in the array.

Return Value

true if successful, false - if you can not remove elements.

Example:

```
//--- example for CArrayFloat::DeleteRange(int,int)  
#include <Arrays\ArrayFloat.mqh>  
//---  
void OnStart()  
{  
    CArrayFloat *array=new CArrayFloat;  
    //---  
    if(array==NULL)  
    {  
        printf("Object create error");  
        return;  
    }  
    //--- add arrays elements  
    //--- . . .  
    //--- delete elements  
    if(!array.DeleteRange(0,10))  
    {  
        printf("Delete error");  
        delete array;  
        return;  
    }  
    //--- delete array  
    delete array;  
}
```

At

Gets the element from the given position in the array.

```
float At(  
    int pos      // Position  
) const
```

Parameters

pos

[in] Position of the desired element in the array.

Return Value

The value of the element in case of success, FLT_MAX if there was an attempt to get an element of not existing positions (the last error ERR_OUT_OF_RANGE) .

Note

Of course, FLT_MAX may be a valid value of an array element, so having a value, always check the last error code.

Example:

```
///--- example for CArrayFloat::At(int)  
#include <Arrays\ArrayFloat.mqh>  
///---  
void OnStart()  
{  
    CArrayFloat *array=new CArrayFloat;  
    ///---  
    if(array==NULL)  
    {  
        printf("Object create error");  
        return;  
    }  
    ///--- add arrays elements  
    ///--- . . .  
    for(int i=0;i<array.Total();i++)  
    {  
        float result=array.At(i);  
        if(result==FLT_MAX && GetLastError()==ERR_OUT_OF_RANGE)  
        {  
            ///--- error reading from array  
            printf("Get element error");  
            delete array;  
            return;  
        }  
        ///--- use element  
        ///--- . . .  
    }  
}
```



```
//--- delete array  
delete array;  
}
```

CompareArray

Compares array with another array.

```
bool CompareArray(  
    const float& src[]      // Source array  
    ) const
```

Parameters

src[]

[in] Reference to an array of source elements for comparison.

Return Value

true if arrays are equal, false - if not.

Example:

```
//--- example for CArrayFloat::CompareArray(const float &[])  
#include <Arrays\ArrayFloat.mqh>  
//---  
float src[];  
//---  
void OnStart()  
{  
    CArrayFloat *array=new CArrayFloat;  
    //---  
    if(array==NULL)  
    {  
        printf("Object create error");  
        return;  
    }  
    //--- compare with another array  
    int result=array.CompareArray(src);  
    //--- delete array  
    delete array;  
}
```

AssignArrayconst

Copies the array elements from another array.

```
bool AssignArrayconst(  
    const CArrayFloat* src      // Pointer to the source  
    ) const
```

Parameters

src

[in] Pointer to an instance of class CArrayFloat-source element to copy.

Return Value

true if successful, false - if you can not copy the items.

Example:

```
//--- example for CArrayFloat::CompareArray(const CArrayFloat*)  
#include <Arrays\ArrayFloat.mqh>  
//---  
void OnStart()  
{  
    CArrayFloat *array=new CArrayFloat;  
    //---  
    if(array==NULL)  
    {  
        printf("Object create error");  
        return;  
    }  
    //--- create source array  
    CArrayFloat *src=new CArrayFloat;  
    if(src==NULL)  
    {  
        printf("Object create error");  
        delete array;  
        return;  
    }  
    //--- add source arrays elements  
    //--- . . .  
    //--- compare with another array  
    int result=array.CompareArray(src);  
    //--- delete arrays  
    delete src;  
    delete array;  
}
```

InsertSort

Inserts element in a sorted array.

```
bool InsertSort(  
    float element    // Element to insert  
)
```

Parameters

element

[in] Value of the element to be inserted into a sorted array

Return Value

true if successful, false - if you can not insert the element.

Example:

```
//--- example for CArrayFloat::InsertSort(float)  
#include <Arrays\ArrayFloat.mqh>  
//---  
void OnStart()  
{  
    CArrayFloat *array=new CArrayFloat;  
    //---  
    if(array==NULL)  
    {  
        printf("Object create error");  
        return;  
    }  
    //--- add arrays elements  
    //--- . . .  
    //--- sort array  
    array.Sort();  
    //--- insert element  
    if(!array.InsertSort(100.0))  
    {  
        printf("Insert error");  
        delete array;  
        return;  
    }  
    //--- delete array  
    delete array;  
}
```

Search

Searches for an element equal to the sample in the sorted array.

```
int Search(  
    float element    // Sample  
    ) const
```

Parameters

element

[in] The sample element to search in the array.

Return Value

The position of the found element, if successful, -1 - if the item was not found.

Example:

```
//--- example for CArrayFloat::Search(float)  
#include <Arrays\ArrayFloat.mqh>  
//---  
void OnStart()  
{  
    CArrayFloat *array=new CArrayFloat;  
    //---  
    if(array==NULL)  
    {  
        printf("Object create error");  
        return;  
    }  
    //--- add arrays elements  
    //--- . . .  
    //--- sort array  
    array.Sort();  
    //--- search element  
    if(array.Search(100.0)!=-1) printf("Element found");  
    else                        printf("Element not found");  
    //--- delete array  
    delete array;  
}
```

SearchGreat

Searches for an element of more samples in sorted array.

```
int SearchGreat(  
    float element    // Sample  
) const
```

Parameters

element

[in] The sample element to search in the array.

Return Value

The position of the found element, if successful, -1 - if the item was not found.

Example:

```
//--- example for CArrayFloat::SearchGreat(float)  
#include <Arrays\ArrayFloat.mqh>  
//---  
void OnStart()  
{  
    CArrayFloat *array=new CArrayFloat;  
    //---  
    if(array==NULL)  
    {  
        printf("Object create error");  
        return;  
    }  
    //--- add arrays elements  
    //--- . . .  
    //--- sort array  
    array.Sort();  
    //--- search element  
    if(array.SearchGreat(100.0)!=-1) printf("Element found");  
    else                             printf("Element not found");  
    //--- delete array  
    delete array;  
}
```

SearchLess

Searches for an element less than the sample in the sorted array.

```
int SearchLess(  
    float element    // Sample  
    ) const
```

Parameters

element

[in] The sample element to search in the array.

Return Value

The position of the found element, if successful, -1 - if the item was not found.

Example:

```
//--- example for CArrayFloat:: SearchLess(float)  
#include <Arrays\ArrayFloat.mqh>  
//---  
void OnStart()  
{  
    CArrayFloat *array=new CArrayFloat;  
    //---  
    if(array==NULL)  
    {  
        printf("Object create error");  
        return;  
    }  
    //--- add arrays elements  
    //--- . . .  
    //--- sort array  
    array.Sort();  
    //--- search element  
    if(array.SearchLess(100.0)!=-1) printf("Element found");  
    else                          printf("Element not found");  
    //--- delete array  
    delete array;  
}
```

SearchGreatOrEqual

Searches for an element greater than or equal to the sample in the sorted array.

```
int SearchGreatOrEqual(  
    float element    // Sample  
    ) const
```

Parameters

element

[in] The sample element to search in the array.

Return Value

The position of the found element, if successful, -1 - if the item was not found.

Example:

```
//--- example for CArrayFloat::SearchGreatOrEqual(float)  
#include <Arrays\ArrayFloat.mqh>  
//---  
void OnStart()  
{  
    CArrayFloat *array=new CArrayFloat;  
    //---  
    if(array==NULL)  
    {  
        printf("Object create error");  
        return;  
    }  
    //--- add arrays elements  
    //--- . . .  
    //--- sort array  
    array.Sort();  
    //--- search element  
    if(array.SearchGreatOrEqual(100.0)!=-1) printf("Element found");  
    else                                  printf("Element not found");  
    //--- delete array  
    delete array;  
}
```


SearchLessOrEqual

Searches for an element less than or equal to the sample in the sorted array.

```
int SearchLessOrEqual(  
    float element    // Sample  
) const
```

Parameters

element

[in] The sample element to search in the array.

Return Value

The position of the found element, if successful, -1 - if the item was not found.

Example:

```
//--- example for CArrayFloat::SearchLessOrEqual(float)  
#include <Arrays\ArrayFloat.mqh>  
//---  
void OnStart()  
{  
    CArrayFloat *array=new CArrayFloat;  
    //---  
    if(array==NULL)  
    {  
        printf("Object create error");  
        return;  
    }  
    //--- add arrays elements  
    //--- . . .  
    //--- sort array  
    array.Sort();  
    //--- search element  
    if(array.SearchLessOrEqual(100.0)!=-1) printf("Element found");  
    else                                printf("Element not found");  
    //--- delete array  
    delete array;  
}
```

SearchFirst

Finds the first element equal to the sample in the sorted array.

```
int SearchFirst(  
    float element    // Sample  
) const
```

Parameters

element

[in] The sample element to search in the array.

Return Value

The position of the found element, if successful, -1 - if the item was not found.

Example:

```
//--- example for CArrayFloat::SearchFirst(float)  
#include <Arrays\ArrayFloat.mqh>  
//---  
void OnStart()  
{  
    CArrayFloat *array=new CArrayFloat;  
    //---  
    if(array==NULL)  
    {  
        printf("Object create error");  
        return;  
    }  
    //--- add arrays elements  
    //--- . . .  
    //--- sort array  
    array.Sort();  
    //--- search element  
    if(array.SearchFirst(100.0)!=-1) printf("Element found");  
    else                             printf("Element not found");  
    //--- delete array  
    delete array;  
}
```

SearchLast

Finds the last element equal to the model in sorted array.

```
int SearchLast(  
    float element    // Sample  
    ) const
```

Parameters

element

[in] The sample element to search in the array.

Return Value

The position of the found element, if successful, -1 - if the item was not found.

Example:

```
//--- example for CArrayFloat::SearchLast(float)  
#include <Arrays\ArrayFloat.mqh>  
//---  
void OnStart()  
{  
    CArrayFloat *array=new CArrayFloat;  
    //---  
    if(array==NULL)  
    {  
        printf("Object create error");  
        return;  
    }  
    //--- add arrays elements  
    //--- . . .  
    //--- sort array  
    array.Sort();  
    //--- search element  
    if(array.SearchLast(100.0)!=-1) printf("Element found");  
    else                             printf("Element not found");  
    //--- delete array  
    delete array;  
}
```

Save

Saves data array in the file.

```
virtual bool Save(  
    int file_handle    // File handle  
)
```

Parameters

file_handle

[in] Handle to open earlier, with the function FileOpen (...) , binary file.

Return Value

true - if successfully completed, false - if an error.

Example:

```
//--- example for CArrayFloat::Save(int)  
#include <Arrays\ArrayFloat.mqh>  
//---  
void OnStart()  
{  
    int file_handle;  
    CArrayFloat *array=new CArrayFloat;  
    //---  
    if(array!=NULL)  
    {  
        printf("Object create error");  
        return;  
    }  
    //--- add 100 arrays elements  
    for(int i=0;i<100;i++)  
    {  
        array.Add(i);  
    }  
    //--- open file  
    file_handle=FileOpen("MyFile.bin",FILE_WRITE|FILE_BIN|FILE_ANSI);  
    if(file_handle>=0)  
    {  
        if(!array.Save(file_handle))  
        {  
            //--- file save error  
            printf("File save: Error %d!",GetLastError());  
            delete array;  
            FileClose(file_handle);  
            //---  
            return;  
        }  
        FileClose(file_handle);  
    }
```

```
    }  
    delete array;  
}
```

Load

Loads data array from a file.

```
virtual bool Load(
    int file_handle    // File handle
)
```

Parameters

file_handle

[in] Handle to open earlier, with the function FileOpen (...) , binary file.

Return Value

true - if successfully completed, false - if an error.

Example:

```
//--- example for CArrayFloat::Load(int)
#include <Arrays\ArrayFloat.mqh>
//---
void OnStart()
{
    int file_handle;
    CArrayFloat *array=new CArrayFloat;
    //---
    if(array!=NULL)
    {
        printf("Object create error");
        return;
    }
    //--- open file
    file_handle=FileOpen("MyFile.bin",FILE_READ|FILE_BIN|FILE_ANSI);
    if(file_handle>=0)
    {
        if(!array.Load(file_handle))
        {
            //--- file load error
            printf("File load: Error %d!",GetLastError());
            delete array;
            FileClose(file_handle);
            //---
            return;
        }
        FileClose(file_handle);
    }
    //--- use arrays elements
    for(int i=0;i<array.Total();i++)
    {
        printf("Element[%d] = %f",i,array.At(i));
    }
}
```

```
    }  
    delete array;  
}
```

Type

Gets the type identifier of the array.

```
virtual int Type() const
```

Return Value

ID type of the array (for CArrayFloat - 87) .

Example:

```
//--- example for CArrayFloat::Type()
#include <Arrays\ArrayFloat.mqh>
//---
void OnStart()
{
    CArrayFloat *array=new CArrayFloat;
    //---
    if(array==NULL)
    {
        printf("Object create error");
        return;
    }
    //--- get array type
    int type=array.Type();
    //--- delete array
    delete array;
}
```


CArrayDouble

CArrayDouble class is a class of dynamic array of variables of type double.

Description

Class CArrayDouble provides the ability to work with a dynamic array of variables of type double. In the class implemented the ability to add / insert / delete elements in an array, sort array, searching in sorted array. In addition, the implemented methods of work with the file.

Declaration

```
class CArrayDouble : public CArray
```

Title

```
#include <Arrays\ArrayDouble.mqh>
```

Class Methods

Attributes	
Delta	Set the comparison tolerance
Memory control	
Reserve	Allocates memory to increase the size of the array
Resize	Sets a new (smaller) size of the array
Shutdown	Clears the array with a full memory release
Add methods	
Add	Adds an element to the end of the array
AddArray	Adds to the end of the array elements from another array
AddArray	Adds to the end of the array elements from another array
Insert	Inserts an element in the array to the specified position
InsertArray	Inserts an array of elements from another array with the specified position
InsertArray	Inserts an array of elements from another array with the specified position
AssignArray	Copies the array elements from another array
AssignArray	Copies the array elements from another array
Update methods	
Update	Changes the element at the specified position array
Shift	Moves an item from a given position in the array to the specified offset

Delete methods	
Delete	Removes the element from the specified position array
DeleteRange	Deletes a group of elements from the specified position array
Access methods	
At	Gets the element from the specified position array
Compare methods	
CompareArray	Compares array with another array
CompareArray	Compares array with another array
Sorted array operations	
InsertSort	Inserts element in a sorted array
Search	Searches for an element equal to the model in sorted array
SearchGreat	Searches for an element of more samples in sorted array
SearchLess	Searches for an element less than the sample in the sorted array
SearchGreatOrEqual	Searches for an element greater than or equal to the model in sorted array
SearchLessOrEqual	Searches for an element less than or equal to the model in sorted array
SearchFirst	Finds the first element equal to the model in sorted array
SearchLast	Finds the last element equal to the model in sorted array
Input/output	
virtual Save	Saves data array in the file
virtual Load	Loads data array from a file
virtual Type	Gets the type identifier of the array

Derived classes:

- [CIndicatorBuffer](#)

Delta

Set tolerance comparison.

```
void Delta(  
    double delta    // Tolerance  
)
```

Parameters

delta

[in] The new value of the admission of comparison.

Return Value

No

Note

Admission of comparison used in the search. Values are considered equal if their difference is less than or equal to tolerance. The default tolerance is 0.0.

Example:

```
//--- example for CArrayDouble::Delta(double)  
#include <Arrays\ArrayDouble.mqh>  
//---  
void OnStart()  
{  
    CArrayDouble *array=new CArrayDouble;  
    //---  
    if(array==NULL)  
    {  
        printf("Object create error");  
        return;  
    }  
    //--- set compare variation  
    array.Delta(0.001);  
    //--- use array  
    //--- . . .  
    //--- delete array  
    delete array;  
}
```

Reserve

Allocates memory to increase the size of the array.

```
bool Reserve(  
    int size      // Number  
)
```

Parameters

size

[in] The number of additional elements of the array.

Return Value

true if successful, false - if there was an attempt to seek the amount is less than or equal to zero, or if the array did not increase.

Note

To reduce memory fragmentation, increase the size of the array is made with a step previously given through the method of Step (int) , or 16 (default) .

Example:

```
//--- example for CArrayDouble::Reserve(int)  
#include <Arrays\ArrayDouble.mqh>  
//---  
void OnStart()  
{  
    CArrayDouble *array=new CArrayDouble;  
    //---  
    if(array==NULL)  
    {  
        printf("Object create error");  
        return;  
    }  
    //--- reserve memory  
    if(!array.Reserve(1024))  
    {  
        printf("Reserve error");  
        delete array;  
        return;  
    }  
    //--- use array  
    //--- . . .  
    //--- delete array  
    delete array;  
}
```

Resize

Sets a new (smaller) size of the array.

```
bool Resize(  
    int size      // Size  
)
```

Parameters

size

[in] New size of the array.

Return Value

true if successful, false - if there was an attempt to set the size of less than or equal to zero.

Note

Changing the size of the array allows optimal use of memory. Superfluous elements on the right lost. To reduce fragmentation of memory, change the size of the array is made with a step previously given through the method of Step (int) , or 16 (default) .

Example:

```
//--- example for CArrayDouble::Resize(int)  
#include <Arrays\ArrayDouble.mqh>  
//---  
void OnStart()  
{  
    CArrayDouble *array=new CArrayDouble;  
    //---  
    if(array==NULL)  
    {  
        printf("Object create error");  
        return;  
    }  
    //--- add arrays elements  
    //--- . . .  
    //--- resize array  
    if(!array.Resize(10))  
    {  
        printf("Resize error");  
        delete array;  
        return;  
    }  
    //--- delete array  
    delete array;  
}
```

Shutdown

Clears the array with a full memory release.

```
bool Shutdown()
```

Return Value

true if successful, false - if an error occurred.

Example:

```
//--- example for CArrayDouble::Shutdown()
#include <Arrays\ArrayDouble.mqh>
//---
void OnStart()
{
    CArrayDouble *array=new CArrayDouble;
    //---
    if(array==NULL)
    {
        printf("Object create error");
        return;
    }
    //--- add arrays elements
    //--- . . .
    //--- shutdown array
    if(!array.Shutdown())
    {
        printf("Shutdown error");
        delete array;
        return;
    }
    //--- delete array
    delete array;
}
```

Add

Adds an element to the end of the array.

```
bool Add(  
    double element    // Element to add  
)
```

Parameters

element

[in] value of the element to add to the array.

Return Value

true if successful, false - if you can not add an element.

Example:

```
///--- example for CArrayDouble::Add(double)  
#include <Arrays\ArrayDouble.mqh>  
///---  
void OnStart()  
{  
    CArrayDouble *array=new CArrayDouble;  
    ///---  
    if(array==NULL)  
    {  
        printf("Object create error");  
        return;  
    }  
    ///--- add arrays elements  
    for(int i=0;i<100;i++)  
    {  
        if(!array.Add(i))  
        {  
            printf("Element addition error");  
            delete array;  
            return;  
        }  
    }  
    ///--- use array  
    ///--- . . .  
    ///--- delete array  
    delete array;  
}
```

AddArray

Adds to the end of the array elements from another array.

```
bool AddArray(  
    const double& src[]    // Source array  
)
```

Parameters

src[]

[in] Reference to an array of source elements to add.

Return Value

true if successful, false - if you can not add items.

Example:

```
///--- example for CArrayDouble::AddArray(const double &[])  
#include <Arrays\ArrayDouble.mqh>  
///---  
double src[];  
///---  
void OnStart()  
{  
    CArrayDouble *array=new CArrayDouble;  
    ///---  
    if(array==NULL)  
    {  
        printf("Object create error");  
        return;  
    }  
    ///--- add another array  
    if(!array.AddArray(src))  
    {  
        printf("Array addition error");  
        delete array;  
        return;  
    }  
    ///--- use array  
    ///--- . . .  
    ///--- delete array  
    delete array;  
}
```


AddArray

Adds to the end of the array elements from another array.

```
bool AddArray(  
    const CArrayDouble* src      // Pointer to the source  
)
```

Parameters

src

[in] Pointer to an instance of class CArrayDouble-source elements to add.

Return Value

true if successful, false - if you can not add items.

Example:

```
///--- example for CArrayDouble::AddArray(const CArrayDouble*)  
#include <Arrays\ArrayDouble.mqh>  
///---  
void OnStart()  
{  
    CArrayDouble *array=new CArrayDouble;  
    ///---  
    if(array==NULL)  
    {  
        printf("Object create error");  
        return;  
    }  
    ///--- create source array  
    CArrayDouble *src=new CArrayDouble;  
    if(src==NULL)  
    {  
        printf("Object create error");  
        delete array;  
        return;  
    }  
    ///--- add source arrays elements  
    ///--- . . .  
    ///--- add another array  
    if(!array.AddArray(src))  
    {  
        printf("Array addition error");  
        delete src;  
        delete array;  
        return;  
    }  
    ///--- delete source array  
    delete src;
```

```
//--- use array  
//--- . . .  
//--- delete array  
delete array;  
}
```

Insert

Inserts an element in the array to the specified position.

```
bool Insert(  
    double element,    // Element to insert  
    int pos            // Position  
)
```

Parameters

element

[in] Value of the element to be inserted into an array

pos

[in] Position in the array to insert

Return Value

true if successful, false - if you can not insert the element.

Example:

```
//--- example for CArrayDouble::Insert(double,int)  
#include <Arrays\ArrayDouble.mqh>  
//---  
void OnStart()  
{  
    CArrayDouble *array=new CArrayDouble;  
    //---  
    if(array==NULL)  
    {  
        printf("Object create error");  
        return;  
    }  
    //--- insert elements  
    for(int i=0;i<100;i++)  
    {  
        if(!array.Insert(i,0))  
        {  
            printf("Insert error");  
            delete array;  
            return;  
        }  
    }  
    //--- use array  
    //--- . . .  
    //--- delete array  
    delete array;  
}
```

InsertArray

Inserts an array of elements from another array specified position.

```
bool  InsertArray(  
    const double&  src[],      // Source array  
    int            pos         // Position  
)
```

Parameters

src[]

[in] Reference to an array of source elements to insert

pos

[in] Position in the array to insert

Return Value

true if successful, false - if you can not paste items.

Example:

```
//--- example for CArrayDouble::InsertArray(const double &[],int)  
#include <Arrays\ArrayDouble.mqh>  
//---  
double src[];  
//---  
void OnStart()  
{  
    CArrayDouble *array=new CArrayDouble;  
    //---  
    if(array==NULL)  
    {  
        printf("Object create error");  
        return;  
    }  
    //--- insert another array  
    if(!array.InsertArray(src,0))  
    {  
        printf("Array inserting error");  
        delete array;  
        return;  
    }  
    //--- use array  
    //--- . . .  
    //--- delete array  
    delete array;  
}
```

InsertArray

Inserts an array of elements from another array specified position.

```
bool  InsertArray(  
    CArrayDouble*  src,      // Pointer to the source  
    int            pos       // Position  
)
```

Parameters

src

[in] Pointer to an instance of class CArrayDouble-source elements to insert.

pos

[in] Position in the array to insert

Return Value

true if successful, false - if you can not paste items.

Example:

```
//--- example for CArrayDouble::InsertArray(const CArrayDouble*,int)  
#include <Arrays\ArrayDouble.mqh>  
//---  
void OnStart()  
{  
    CArrayDouble *array=new CArrayDouble;  
    //---  
    if(array==NULL)  
    {  
        printf("Object create error");  
        return;  
    }  
    //--- create source array  
    CArrayDouble *src=new CArrayDouble;  
    if(src==NULL)  
    {  
        printf("Object create error");  
        delete array;  
        return;  
    }  
    //--- add source arrays elements  
    //--- . . .  
    //--- insert another array  
    if(!array.InsertArray(src,0))  
    {  
        printf("Array inserting error");  
        delete src;  
        delete array;  
    }  
}
```

```
        return;  
    }  
    //--- delete source array  
    delete src;  
    //--- use array  
    //--- . . .  
    //--- delete array  
    delete array;  
}
```

AssignArray

Copies the array elements from another array.

```
bool AssignArray(  
    const double& src[]      // Source array  
)
```

Parameters

src[]

[in] Reference to an array of source elements to copy.

Return Value

true if successful, false - if you can not copy the items.

Example:

```
///--- example for CArrayDouble::AssignArray(const double &[])  
#include <Arrays\ArrayDouble.mqh>  
///---  
double src[];  
///---  
void OnStart()  
{  
    CArrayDouble *array=new CArrayDouble;  
    ///---  
    if(array==NULL)  
    {  
        printf("Object create error");  
        return;  
    }  
    ///--- assign another array  
    if(!array.AssignArray(src))  
    {  
        printf("Array assigned error");  
        delete array;  
        return;  
    }  
    ///--- use array  
    ///--- . . .  
    ///--- delete array  
    delete array;  
}
```

AssignArray

Copies the array elements from another array.

```
bool AssignArray(  
    const CArrayDouble* src      // Pointer to the source  
)
```

Parameters

src

[in] Pointer to an instance of class CArrayDouble-source element to copy.

Return Value

true if successful, false - if you can not copy the items.

Example:

```
//--- example for CArrayDouble::AssignArray(const CArrayDouble*)  
#include <Arrays\ArrayDouble.mqh>  
//---  
void OnStart()  
{  
    CArrayDouble *array=new CArrayDouble;  
    //---  
    if(array==NULL)  
    {  
        printf("Object create error");  
        return;  
    }  
    //--- create source array  
    CArrayDouble *src =new CArrayDouble;  
    if(src==NULL)  
    {  
        printf("Object create error");  
        delete array;  
        return;  
    }  
    //--- add source arrays elements  
    //--- . . .  
    //--- assign another array  
    if(!array.AssignArray(src))  
    {  
        printf("Array assigned error");  
        delete src;  
        delete array;  
        return;  
    }  
    //--- arrays is identical  
    //--- delete source array
```



```
delete src;  
//--- use array  
//--- . . .  
//--- delete array  
delete array;  
}
```

Update

Changes the element at the specified position array.

```
bool Update(  
    int    pos,           // Position  
    double element       // Value  
)
```

Parameters

pos

[in] Position of the element in the array to change

element

[in] New value.

Return Value

true if successful, false - if you can not change the element.

Example:

```
//--- example for CArrayDouble::Update(int,double)  
#include <Arrays\ArrayDouble.mqh>  
//---  
void OnStart()  
{  
    CArrayDouble *array=new CArrayDouble;  
    //---  
    if(array==NULL)  
    {  
        printf("Object create error");  
        return;  
    }  
    //--- add arrays elements  
    //--- . . .  
    //--- update element  
    if(!array.Update(0,100.0))  
    {  
        printf("Update error");  
        delete array;  
        return;  
    }  
    //--- delete array  
    delete array;  
}
```

Shift

Moves an item from a given position in the array to the specified offset.

```
bool Shift(  
    int pos,          // Position  
    int shift         // Shift  
)
```

Parameters

pos

[in] Position of the moved element array

shift

[in] The value of displacement (both positive and negative) .

Return Value

true if successful, false - if you can not move the item.

Example:

```
//--- example for CArrayDouble::Shift(int,int)  
#include <Arrays\ArrayDouble.mqh>  
//---  
void OnStart()  
{  
    CArrayDouble *array=new CArrayDouble;  
    //---  
    if(array==NULL)  
    {  
        printf("Object create error");  
        return;  
    }  
    //--- add arrays elements  
    //--- . . .  
    //--- shift element  
    if(!array.Shift(10,-5))  
    {  
        printf("Shift error");  
        delete array;  
        return;  
    }  
    //--- delete array  
    delete array;  
}
```

Delete

Removes the element from the given position in the array.

```
bool Delete(  
    int pos    // Position  
)
```

Parameters

pos

[in] position removes the element in the array.

Return Value

true if successful, false - if you can not remove the element.

Example:

```
///--- example for CArrayDouble::Delete(int)  
#include <Arrays\ArrayDouble.mqh>  
///---  
void OnStart()  
{  
    CArrayDouble *array=new CArrayDouble;  
    ///---  
    if(array==NULL)  
    {  
        printf("Object create error");  
        return;  
    }  
    ///--- add arrays elements  
    ///--- . . .  
    ///--- delete element  
    if(!array.Delete(0))  
    {  
        printf("Delete error");  
        delete array;  
        return;  
    }  
    ///--- delete array  
    delete array;  
}
```

DeleteRange

Deletes a group of elements from a given position in the array.

```
bool DeleteRange(  
    int from,      // Position of the first element  
    int to         // Position of last element  
)
```

Parameters

from

[in] Position of the first removes the element in the array.

to

[in] Position of the last deleted element in the array.

Return Value

true if successful, false - if you can not remove elements.

Example:

```
//--- example for CArrayDouble::DeleteRange(int,int)  
#include <Arrays\ArrayDouble.mqh>  
//---  
void OnStart()  
{  
    CArrayDouble *array=new CArrayDouble;  
    //---  
    if(array==NULL)  
    {  
        printf("Object create error");  
        return;  
    }  
    //--- add arrays elements  
    //--- . . .  
    //--- delete elements  
    if(!array.DeleteRange(0,10))  
    {  
        printf("Delete error");  
        delete array;  
        return;  
    }  
    //--- delete array  
    delete array;  
}
```

At

Gets the element from the given position in the array.

```
double At(  
    int pos      // Position  
) const
```

Parameters

pos

[in] Position of the desired element in the array.

Return Value

The value of the element in case of success, DBL_MAX if there was an attempt to get an element of not existing positions (the last error ERR_OUT_OF_RANGE) .

Note

Of course, DBL_MAX may be a valid value of an array element, so having a value, always check the last error code.

Example:

```
//--- example for CArrayDouble::At(int)  
#include <Arrays\ArrayDouble.mqh>  
//---  
void OnStart()  
{  
    CArrayDouble *array=new CArrayDouble;  
    //---  
    if(array==NULL)  
    {  
        printf("Object create error");  
        return;  
    }  
    //--- add arrays elements  
    //--- . . .  
    for(int i=0;i<array.Total();i++)  
    {  
        double result=array.At(i);  
        if(result==DBL_MAX && GetLastError()==ERR_OUT_OF_RANGE)  
        {  
            //--- Error reading from the array  
            printf("Get element error");  
            delete array;  
            return;  
        }  
        //--- use element  
        //--- . . .  
    }  
}
```

```
//--- delete array  
delete array;  
}
```

CompareArray

Compares array with another array.

```
bool CompareArray(  
    const double& src[]      // Source array  
    ) const
```

Parameters

src[]

[in] Reference to an array of source elements for comparison.

Return Value

true if arrays are equal, false - if not.

Example:

```
//--- example for CArrayDouble::CompareArray(const double &[])  
#include <Arrays\ArrayDouble.mqh>  
//---  
double src[];  
//---  
void OnStart()  
{  
    CArrayDouble *array=new CArrayDouble;  
    //---  
    if(array==NULL)  
    {  
        printf("Object create error");  
        return;  
    }  
    //--- compare with another array  
    int result=array.CompareArray(src);  
    //--- delete array  
    delete array;  
}
```


CompareArray

Compares array with another array.

```
bool CompareArray(  
    const CArrayDouble* src      // Pointer to the source  
    ) const
```

Parameters

src

[in] Pointer to an instance of class CArrayDouble-source elements for comparison.

Return Value

true if arrays are equal, false - if not.

Example:

```
///--- example for CArrayDouble::CompareArray(const CArrayDouble*)  
#include <Arrays\ArrayDouble.mqh>  
///---  
void OnStart()  
{  
    CArrayDouble *array=new CArrayDouble;  
    ///---  
    if(array==NULL)  
    {  
        printf("Object create error");  
        return;  
    }  
    ///--- create source array  
    CArrayDouble *src=new CArrayDouble;  
    if(src==NULL)  
    {  
        printf("Object create error");  
        delete array;  
        return;  
    }  
    ///--- add source arrays elements  
    ///--- . . .  
    ///--- compare with another array  
    int result=array.CompareArray(src);  
    ///--- delete arrays  
    delete src;  
    delete array;  
}
```

InsertSort

Inserts element in a sorted array.

```
bool  InsertSort(  
    double  element      // Element to insert  
)
```

Parameters

element

[in] value of the element to be inserted into a sorted array

Return Value

true if successful, false - if you can not insert the element.

Example:

```
///--- example for CArrayDouble::InsertSort(double)  
#include <Arrays\ArrayDouble.mqh>  
///---  
void OnStart()  
{  
    CArrayDouble *array=new CArrayDouble;  
    ///---  
    if(array==NULL)  
    {  
        printf("Object create error");  
        return;  
    }  
    ///--- add arrays elements  
    ///--- . . .  
    ///--- sort array  
    array.Sort();  
    ///--- insert element  
    if(!array.InsertSort(100.0))  
    {  
        printf("Insert error");  
        delete array;  
        return;  
    }  
    ///--- delete array  
    delete array;  
}
```

Search

Searches for an element equal to the sample in the sorted array.

```
int Search(  
    double element    // Sample  
    ) const
```

Parameters

element

[in] The sample element to search in the array.

Return Value

The position of the found element, if successful, -1 - if the item was not found.

Example:

```
//--- example for CArrayDouble::Search(double)  
#include <Arrays\ArrayDouble.mqh>  
//---  
void OnStart()  
{  
    CArrayDouble *array=new CArrayDouble;  
    //---  
    if(array==NULL)  
    {  
        printf("Object create error");  
        return;  
    }  
    //--- add arrays elements  
    //--- . . .  
    //--- sort array  
    array.Sort();  
    //--- search element  
    if(array.Search(100.0)!=-1) printf("Element found");  
    else                        printf("Element not found");  
    //--- delete array  
    delete array;  
}
```

SearchGreat

Searches for an element of more samples in sorted array.

```
int SearchGreat(  
    double element    // Sample  
    ) const
```

Parameters

element

[in] The sample element to search in the array.

Return Value

The position of the found element, if successful, -1 - if the item was not found.

Example:

```
//--- example for CArrayDouble::SearchGreat(double)  
#include <Arrays\ArrayDouble.mqh>  
//---  
void OnStart()  
{  
    CArrayDouble *array=new CArrayDouble;  
    //---  
    if(array==NULL)  
    {  
        printf("Object create error");  
        return;  
    }  
    //--- add arrays elements  
    //--- . . .  
    //--- sort array  
    array.Sort();  
    //--- search element  
    if(array.SearchGreat(100.0)!=-1) printf("Element found");  
    else                             printf("Element not found");  
    //--- delete array  
    delete array;  
}
```

SearchLess

Searches for an element less than the sample in the sorted array.

```
int SearchLess(  
    double element    // Sample  
    ) const
```

Parameters

element

[in] The sample element to search in the array.

Return Value

The position of the found element, if successful, -1 - if the item was not found.

Example:

```
//--- example for CArrayDouble:: SearchLess(double)  
#include <Arrays\ArrayDouble.mqh>  
//---  
void OnStart()  
{  
    CArrayDouble *array=new CArrayDouble;  
    //---  
    if(array==NULL)  
    {  
        printf("Object create error");  
        return;  
    }  
    //--- add arrays elements  
    //--- . . .  
    //--- sort array  
    array.Sort();  
    //--- search element  
    if(array.SearchLess(100.0)!=-1) printf("Element found");  
    else                          printf("Element not found");  
    //--- delete array  
    delete array;  
}
```

SearchGreatOrEqual

Searches for an element greater than or equal to the sample in the sorted array.

```
int SearchGreatOrEqual(  
    double element    // Sample  
    ) const
```

Parameters

element

[in] The sample element to search in the array.

Return Value

The position of the found element, if successful, -1 - if the item was not found.

Example:

```
///--- example for CArrayDouble::SearchGreatOrEqual(double)  
#include <Arrays\ArrayDouble.mqh>  
///---  
void OnStart()  
{  
    CArrayDouble *array=new CArrayDouble;  
    ///---  
    if(array==NULL)  
    {  
        printf("Object create error");  
        return;  
    }  
    ///--- add arrays elements  
    ///--- . . .  
    ///--- sort array  
    array.Sort();  
    ///--- search element  
    if(array.SearchGreatOrEqual(100.0)!=-1) printf("Element found");  
    else                                  printf("Element not found");  
    ///--- delete array  
    delete array;  
}
```

SearchLessOrEqual

Searches for an element less than or equal to the sample in the sorted array.

```
int SearchLessOrEqual(  
    double element // Sample  
    ) const
```

Parameters

element

[in] The sample element to search in the array.

Return Value

The position of the found element, if successful, -1 - if the item was not found.

Example:

```
//--- example for CArrayDouble::SearchLessOrEqual(double)  
#include <Arrays\ArrayDouble.mqh>  
//---  
void OnStart()  
{  
    CArrayDouble *array=new CArrayDouble;  
    //---  
    if(array==NULL)  
    {  
        printf("Object create error");  
        return;  
    }  
    //--- add arrays elements  
    //--- . . .  
    //--- sort array  
    array.Sort();  
    //--- search element  
    if(array.SearchLessOrEqual(100.0)!=-1) printf("Element found");  
    else printf("Element not found");  
    //--- delete array  
    delete array;  
}
```

SearchFirst

Finds the first element equal to the sample in the sorted array.

```
int SearchFirst(  
    double element    // Sample  
    ) const
```

Parameters

element

[in] The sample element to search in the array.

Return Value

The position of the found element, if successful, -1 - if the item was not found.

Example:

```
//--- example for CArrayDouble::SearchFirst(double)  
#include <Arrays\ArrayDouble.mqh>  
//---  
void OnStart()  
{  
    CArrayDouble *array=new CArrayDouble;  
    //---  
    if(array==NULL)  
    {  
        printf("Object create error");  
        return;  
    }  
    //--- add arrays elements  
    //--- . . .  
    //--- sort array  
    array.Sort();  
    //--- search element  
    if(array.SearchFirst(100.0)!=-1) printf("Element found");  
    else                             printf("Element not found");  
    //--- delete array  
    delete array;  
}
```


SearchLast

Finds the last element equal to the model in sorted array.

```
int SearchLast(  
    double element    // Sample  
    ) const
```

Parameters

element

[in] The sample element to search in the array.

Return Value

The position of the found element, if successful, -1 - if the item was not found.

Example:

```
//--- example for CArrayDouble::SearchLast(double)  
#include <Arrays\ArrayDouble.mqh>  
//---  
void OnStart()  
{  
    CArrayDouble *array=new CArrayDouble;  
    //---  
    if(array==NULL)  
    {  
        printf("Object create error");  
        return;  
    }  
    //--- add arrays elements  
    //--- . . .  
    //--- sort array  
    array.Sort();  
    //--- search element  
    if(array.SearchLast(100.0)!=-1) printf("Element found");  
    else                          printf("Element not found");  
    //--- delete array  
    delete array;  
}
```

Save

Saves data array in the file.

```
virtual bool Save(  
    int file_handle    // File handle  
)
```

Parameters

file_handle

[in] Handle to open earlier, with the function FileOpen (...) , binary file.

Return Value

true - if successfully completed, false - if an error.

Example:

```
//--- example for CArrayDouble::Save(int)  
#include <Arrays\ArrayDouble.mqh>  
//---  
void OnStart()  
{  
    int file_handle;  
    CArrayDouble *array=new CArrayDouble;  
    //---  
    if(array!=NULL)  
    {  
        printf("Object create error");  
        return;  
    }  
    //--- add 100 arrays elements  
    for(int i=0;i<100;i++)  
    {  
        array.Add(i);  
    }  
    //--- open file  
    file_handle=FileOpen("MyFile.bin",FILE_WRITE|FILE_BIN|FILE_ANSI);  
    if(file_handle>=0)  
    {  
        if(!array.Save(file_handle))  
        {  
            //--- file save error  
            printf("File save: Error %d!",GetLastError());  
            delete array;  
            FileClose(file_handle);  
            //---  
            return;  
        }  
        FileClose(file_handle);  
    }
```

```
    }  
    ///--- delete array  
    delete array;  
}
```

Load

Loads data array from a file.

```
virtual bool Load(
    int file_handle    // File handle
)
```

Parameters

file_handle

[in] Handle to open earlier, with the function FileOpen (...) , binary file.

Return Value

true - if successfully completed, false - if an error.

Example:

```
//--- example for CArrayDouble::Load(int)
#include <Arrays\ArrayDouble.mqh>
//---
void OnStart()
{
    int file_handle;
    CArrayDouble *array=new CArrayDouble;
    //---
    if(array!=NULL)
    {
        printf("Object create error");
        return;
    }
    //--- open file
    file_handle=FileOpen("MyFile.bin",FILE_READ|FILE_BIN|FILE_ANSI);
    if(file_handle>=0)
    {
        if(!array.Load(file_handle))
        {
            //--- file load error
            printf("File load: Error %d!",GetLastError());
            delete array;
            FileClose(file_handle);
            //---
            return;
        }
        FileClose(file_handle);
    }
    //--- use arrays elements
    for(int i=0;i<array.Total();i++)
    {
        printf("Element[%d] = %f",i,array.At(i));
    }
}
```

```
    }  
    ///--- delete array  
    delete array;  
}
```

Type

Gets the type identifier of the array.

```
virtual int Type() const
```

Return Value

ID type of the array (for CArrayDouble - 87) .

Example:

```
//--- example for CArrayDouble::Type()
#include <Arrays\ArrayDouble.mqh>
//---
void OnStart()
{
    CArrayDouble *array=new CArrayDouble;
    //---
    if(array==NULL)
    {
        printf("Object create error");
        return;
    }
    //--- get array type
    int type=array.Type();
    //--- delete array
    delete array;
}
```

CArrayString

CArrayString class is a class of dynamic array of variables of type string.

Description

Class CArrayString provides the ability to work with a dynamic array of variables of type string. In the class implemented the ability to add / insert / delete elements in an array, sort array, searching in sorted array. In addition, the implemented methods of work with the file.

Declaration

```
class CArrayString : public CArray
```

Title

```
#include <Arrays\ArrayString.mqh>
```

Class Methods

Memory control	
Reserve	Allocates memory to increase the size of the array
Resize	Sets a new (smaller) size of the array
Shutdown	Sets a new (smaller) size of the array
Add methods	
Add	Adds an element to the end of the array
AddArray	Adds to the end of the array elements from another array
AddArray	Adds to the end of the array elements from another array
Insert	Inserts an element in the array to the specified position
InsertArray	Inserts an array of elements from another array with the specified position
InsertArray	Inserts an array of elements from another array with the specified position
AssignArray	Copies the array elements from another array
AssignArray	Copies the array elements from another array
Update methods	
Update	Changes the element at the specified position array
Shift	Moves an item from a given position in the array to the specified offset
Delete methods	
Delete	Removes the element from the specified position array

DeleteRange	Deletes a group of elements from the specified position array
Access methods	
At	Gets the element from the specified position array
Compare methods	
CompareArray	Compares array with another array
CompareArray	Compares array with another array
Sorted array operations	
InsertSort	Inserts element in a sorted array
Search	Searches for an element equal to the model in sorted array
SearchGreat	Searches for an element of more samples in sorted array
SearchLess	Searches for an element less than the sample in the sorted array
SearchGreatOrEqual	Searches for an element greater than or equal to the model in sorted array
SearchLessOrEqual	Searches for an element less than or equal to the model in sorted array
SearchFirst	Finds the first element equal to the model in sorted array
SearchLast	Finds the last element equal to the model in sorted array
Input/output	
virtual Save	Saves data array in the file
virtual Load	Loads data array from a file
virtual Type	Gets the type identifier of the array

Reserve

Allocates memory to increase the size of the array.

```
bool Reserve(  
    int size      // Number  
)
```

Parameters

size

[in] The number of additional elements of the array.

Return Value

true if successful, false - if there was an attempt to seek the amount is less than or equal to zero, or if the array did not increase.

Note

To reduce memory fragmentation, increase the size of the array is made with a step previously given through the method of Step (int) , or 16 (default) .

Example:

```
//--- example for CArrayString::Reserve(int)  
#include <Arrays\ArrayString.mqh>  
//---  
void OnStart()  
{  
    CArrayString *array=new CArrayString;  
    //---  
    if(array==NULL)  
    {  
        printf("Object create error");  
        return;  
    }  
    //--- reserve memory  
    if(!array.Reserve(1024))  
    {  
        printf("Reserve error");  
        delete array;  
        return;  
    }  
    //--- use array  
    //--- . . .  
    //--- delete array  
    delete array;  
}
```

Resize

Sets a new (smaller) size of the array.

```
bool Resize(  
    int size      // Size  
)
```

Parameters

size

[in] New size of the array.

Return Value

true if successful, false - if there was an attempt to set the size of less than or equal to zero.

Note

Changing the size of the array allows optimal use of memory. Superfluous elements on the right lost. To reduce fragmentation of memory, change the size of the array is made with a step previously given through the method of Step (int) , or 16 (default) .

Example:

```
//--- example for CArrayString::Resize(int)  
#include <Arrays\ArrayString.mqh>  
//---  
void OnStart()  
{  
    CArrayString *array=new CArrayString;  
    //---  
    if(array==NULL)  
    {  
        printf("Object create error");  
        return;  
    }  
    //--- add arrays elements  
    //--- . . .  
    //--- resize array  
    if(!array.Resize(10))  
    {  
        printf("Resize error");  
        delete array;  
        return;  
    }  
    //--- delete array  
    delete array;  
}
```

Shutdown

Clears the array with a full memory release.

```
bool Shutdown()
```

Return Value

true if successful, false - if an error occurred.

Example:

```
//--- example for CArrayString::Shutdown()
#include <Arrays\ArrayString.mqh>
//---
void OnStart()
{
    CArrayString *array=new CArrayString;
    //---
    if(array==NULL)
    {
        printf("Object create error");
        return;
    }
    //--- add arrays elements
    //--- . . .
    //--- shutdown array
    if(!array.Shutdown())
    {
        printf("Shutdown error");
        delete array;
        return;
    }
    //--- delete array
    delete array;
}
```

Add

Adds an element to the end of the array.

```
bool Add(  
    string element    // Element to add  
)
```

Parameters

element

[in] value of the element to add to the array.

Return Value

true if successful, false - if you can not add an element.

Example:

```
//--- example for CArrayString::Add(string)  
#include <Arrays\ArrayString.mqh>  
//---  
void OnStart()  
{  
    CArrayString *array=new CArrayString;  
    //---  
    if(array==NULL)  
    {  
        printf("Object create error");  
        return;  
    }  
    //--- add arrays elements  
    for(int i=0;i<100;i++)  
    {  
        if(!array.Add(IntegerToString(i)))  
        {  
            printf("Element addition error");  
            delete array;  
            return;  
        }  
    }  
    //--- use array  
    //--- . . .  
    //--- delete array  
    delete array;  
}
```

AddArray

Adds to the end of the array elements from another array.

```
bool AddArray(  
    const string& src[]      // Source array  
)
```

Parameters

src[]

[in] Reference to an array of source elements to add.

Return Value

true if successful, false - if you can not add items.

Example:

```
///--- example for CArrayString::AddArray(const string &[])  
#include <Arrays\ArrayString.mqh>  
///---  
string src[];  
///---  
void OnStart()  
{  
    CArrayString *array=new CArrayString;  
    ///---  
    if(array==NULL)  
    {  
        printf("Object create error");  
        return;  
    }  
    ///--- add another array  
    if(!array.AddArray(src))  
    {  
        printf("Array addition error");  
        delete array;  
        return;  
    }  
    ///--- use array  
    ///--- . . .  
    ///--- delete array  
    delete array;  
}
```

AddArray

Adds to the end of the array elements from another array.

```
bool AddArray(  
    const CArrayString* src      // Pointer to the source  
)
```

Parameters

src

[in] Pointer to an instance of class CArrayString - the source of elements to add.

Return Value

true if successful, false - if you can not add items.

Example:

```
///--- example for CArrayString::AddArray(const CArrayString*)  
#include <Arrays\ArrayString.mqh>  
///---  
void OnStart()  
{  
    CArrayString *array=new CArrayString;  
    ///---  
    if(array==NULL)  
    {  
        printf("Object create error");  
        return;  
    }  
    ///--- create source array  
    CArrayString *src=new CArrayString;  
    if(src==NULL)  
    {  
        printf("Object create error");  
        delete array;  
        return;  
    }  
    ///--- add source arrays elements  
    ///--- . . .  
    ///--- add another array  
    if(!array.AddArray(src))  
    {  
        printf("Array addition error");  
        delete src;  
        delete array;  
        return;  
    }  
    ///--- delete source array  
    delete src;
```

```
//--- use array  
//--- . . .  
//--- delete array  
delete array;  
}
```

Insert

Inserts an element in the array to the specified position.

```
bool Insert(  
    string element,    // Element to insert  
    int pos            // Position  
)
```

Parameters

element

[in] Value of the element to be inserted into an array

pos

[in] Position in the array to insert

Return Value

true if successful, false - if you can not insert the element.

Example:

```
//--- example for CArrayString::Insert(string,int)  
#include <Arrays\ArrayString.mqh>  
//---  
void OnStart()  
{  
    CArrayString *array=new CArrayString;  
    //---  
    if(array==NULL)  
    {  
        printf("Object create error");  
        return;  
    }  
    //--- insert elements  
    for(int i=0;i<100;i++)  
    {  
        if(!array.Insert(IntegerToString(i),0))  
        {  
            printf("Insert error");  
            delete array;  
            return;  
        }  
    }  
    //--- use array  
    //--- . . .  
    //--- delete array  
    delete array;  
}
```


InsertArray

Inserts an array of elements from another array specified position.

```
bool  InsertArray(  
    const string&  src[],      // Source array  
    int            pos         // Position  
)
```

Parameters

src[]

[in] Reference to an array of source elements to insert

pos

[in] Position in the array to insert

Return Value

true if successful, false - if you can not paste items.

Example:

```
//--- example for CArrayString::InsertArray(const string &[],int)  
#include <Arrays\ArrayString.mqh>  
//---  
string src[];  
//---  
void OnStart()  
{  
    CArrayString *array=new CArrayString;  
    //---  
    if(array==NULL)  
    {  
        printf("Object create error");  
        return;  
    }  
    //--- insert another array  
    if(!array.InsertArray(src,0))  
    {  
        printf("Array inserting error");  
        delete array;  
        return;  
    }  
    //--- use array  
    //--- . . .  
    //--- delete array  
    delete array;  
}
```

InsertArray

Inserts an array of elements from another array specified position.

```
bool  InsertArray(  
    CArrayString*  src,      // Pointer to the source  
    int            pos       // Position  
)
```

Parameters

src

[in] Pointer to an instance of class CArrayString - the source of elements to insert.

pos

[in] Position in the array to insert

Return Value

true if successful, false - if you can not paste items.

Example:

```
//--- example for CArrayString::InsertArray(const CArrayString*,int)  
#include <Arrays\ArrayString.mqh>  
//---  
void OnStart()  
{  
    CArrayString *array=new CArrayString;  
    //---  
    if(array==NULL)  
    {  
        printf("Object create error");  
        return;  
    }  
    //--- create source array  
    CArrayString *src=new CArrayString;  
    if(src==NULL)  
    {  
        printf("Object create error");  
        delete array;  
        return;  
    }  
    //--- add source arrays elements  
    //--- . . .  
    //--- insert another array  
    if(!array.InsertArray(src,0))  
    {  
        printf("Array inserting error");  
        delete src;  
        delete array;  
    }  
}
```

```
        return;  
    }  
    //--- delete source array  
    delete src;  
    //--- use array  
    //--- . . .  
    //--- delete array  
    delete array;  
}
```

AssignArray

Copies the array elements from another array.

```
bool AssignArray(  
    const string& src[]      // Source array  
)
```

Parameters

src[]

[in] Reference to an array of source elements to copy.

Return Value

true if successful, false - if you can not copy the items.

Example:

```
//--- example for CArrayString::AssignArray(const string &[])  
#include <Arrays\ArrayString.mqh>  
//---  
string src[];  
//---  
void OnStart()  
{  
    CArrayString *array=new CArrayString;  
    //---  
    if(array==NULL)  
    {  
        printf("Object create error");  
        return;  
    }  
    //--- assign another array  
    if(!array.AssignArray(src))  
    {  
        printf("Array assigned error");  
        delete array;  
        return;  
    }  
    //--- use array  
    //--- . . .  
    //--- delete array  
    delete array;  
}
```

AssignArray

Copies the array elements from another array.

```
bool AssignArray(  
    const CArrayString* src      // Pointer to the source  
)
```

Parameters

src

[in] Pointer to an instance of class CArrayString - source of elements to copy.

Return Value

true if successful, false - if you can not copy the items.

Example:

```
//--- example for CArrayString::AssignArray(const CArrayString*)  
#include <Arrays\ArrayString.mqh>  
//---  
void OnStart()  
{  
    CArrayString *array=new CArrayString;  
    //---  
    if(array==NULL)  
    {  
        printf("Object create error");  
        return;  
    }  
    //--- create source array  
    CArrayString *src =new CArrayString;  
    if(src==NULL)  
    {  
        printf("Object create error");  
        delete array;  
        return;  
    }  
    //--- add source arrays elements  
    //--- . . .  
    //--- assign another array  
    if(!array.AssignArray(src))  
    {  
        printf("Array assigned error");  
        delete src;  
        delete array;  
        return;  
    }  
    //--- arrays is identical  
    //--- delete source array
```

```
delete src;  
//--- use array  
//--- . . .  
//--- delete array  
delete array;  
}
```

Update

Changes the element at the specified position array.

```
bool Update(  
    int    pos,           // Position  
    string element       // Value  
)
```

Parameters

pos

[in] Position of the element in the array to change

element

[in] New value element

Return Value

true if successful, false - if you can not change the element.

Example:

```
//--- example for CArrayString::Update(int, string)  
#include <Arrays\ArrayString.mqh>  
//---  
void OnStart()  
{  
    CArrayString *array=new CArrayString;  
    //---  
    if(array==NULL)  
    {  
        printf("Object create error");  
        return;  
    }  
    //--- add arrays elements  
    //--- . . .  
    //--- update element  
    if(!array.Update(0,"ABC"))  
    {  
        printf("Update error");  
        delete array;  
        return;  
    }  
    //--- delete array  
    delete array;  
}
```

Shift

Moves an item from a given position in the array to the specified offset.

```
bool Shift(  
    int pos,          // Position  
    int shift         // Shift  
)
```

Parameters

pos

[in] Position of the moved element array

shift

[in] The value of displacement (both positive and negative) .

Return Value

true if successful, false - if you can not move the item.

Example:

```
//--- example for CArrayString::Shift(int,int)  
#include <Arrays\ArrayString.mqh>  
//---  
void OnStart()  
{  
    CArrayString *array=new CArrayString;  
    //---  
    if(array==NULL)  
    {  
        printf("Object create error");  
        return;  
    }  
    //--- add arrays elements  
    //--- . . .  
    //--- shift element  
    if(!array.Shift(10,-5))  
    {  
        printf("Shift error");  
        delete array;  
        return;  
    }  
    //--- delete array  
    delete array;  
}
```


Delete

Removes the element from the given position in the array.

```
bool Delete(  
    int pos    // Position  
)
```

Parameters

pos

[in] position removes the element in the array.

Return Value

true if successful, false - if you can not remove the element.

Example:

```
///--- example for CArrayString::Delete(int)  
#include <Arrays\ArrayString.mqh>  
///---  
void OnStart()  
{  
    CArrayString *array=new CArrayString;  
    ///---  
    if(array==NULL)  
    {  
        printf("Object create error");  
        return;  
    }  
    ///--- add arrays elements  
    ///--- . . .  
    ///--- delete element  
    if(!array.Delete(0))  
    {  
        printf("Delete error");  
        delete array;  
        return;  
    }  
    ///--- delete array  
    delete array;  
}
```

DeleteRange

Deletes a group of elements from a given position in the array.

```
bool DeleteRange(  
    int from,      // Position of the first element  
    int to         // Position of last element  
)
```

Parameters

from

[in] Position of the first removes the element in the array.

to

[in] Position of the last deleted element in the array.

Return Value

true if successful, false - if you can not remove elements.

Example:

```
//--- example for CArrayString::DeleteRange(int,int)  
#include <Arrays\ArrayString.mqh>  
//---  
void OnStart()  
{  
    CArrayString *array=new CArrayString;  
    //---  
    if(array==NULL)  
    {  
        printf("Object create error");  
        return;  
    }  
    //--- add arrays elements  
    //--- . . .  
    //--- delete elements  
    if(!array.DeleteRange(0,10))  
    {  
        printf("Delete error");  
        delete array;  
        return;  
    }  
    //--- delete array  
    delete array;  
}
```

At

Gets the element from the given position in the array.

```
string At(  
    int pos      // Position  
) const
```

Parameters

pos

[in] Position of the desired element in the array.

Return Value

The value of the element in case of success, "-" if there was an attempt to get an element of not existing positions (the last error `ERR_OUT_OF_RANGE`) .

Note

Of course, "" may be a valid value of an array element, so having a value, always check the last error code.

Example:

```
//--- example for CArrayString::At(int)  
#include <Arrays\ArrayString.mqh>  
//---  
void OnStart()  
{  
    CArrayString *array=new CArrayString;  
    //---  
    if(array==NULL)  
    {  
        printf("Object create error");  
        return;  
    }  
    //--- add arrays elements  
    //--- . . .  
    for(int i=0;i<array.Total();i++)  
    {  
        string result=array.At(i);  
        if(result==" " && GetLastError()==ERR_OUT_OF_RANGE)  
        {  
            //--- Error reading from array  
            printf("Get element error");  
            delete array;  
            return;  
        }  
        //--- use element  
        //--- . . .  
    }  
}
```

```
//--- delete array  
delete array;  
}
```

CompareArray

Compares array with another array.

```
bool CompareArray(  
    const string& src[]    // Source array  
) const
```

Parameters

src[]

[in] Reference to an array of source elements for comparison.

Return Value

true if arrays are equal, false - if not.

Example:

```
///--- example for CArrayString::CompareArray(const string &[])  
#include <Arrays\ArrayString.mqh>  
///---  
string src[];  
///---  
void OnStart()  
{  
    CArrayString *array=new CArrayString;  
    ///---  
    if(array==NULL)  
    {  
        printf("Object create error");  
        return;  
    }  
    ///--- compare with another array  
    int result=array.CompareArray(src);  
    ///--- delete array  
    delete array;  
}
```

CompareArray

Compares array with another array.

```
bool CompareArrays(  
    const CArrayString* src      // Pointer to the source  
    ) const
```

Parameters

src

[in] Pointer to an instance of class CArrayString - the source of elements for comparison.

Return Value

true if arrays are equal, false - if not.

Example:

```
//--- example for CArrayString::CompareArray(const CArrayString*)  
#include <Arrays\ArrayString.mqh>  
//---  
void OnStart()  
{  
    CArrayString *array=new CArrayString;  
    //---  
    if(array==NULL)  
    {  
        printf("Object create error");  
        return;  
    }  
    //--- create source array  
    CArrayString *src=new CArrayString;  
    if(src==NULL)  
    {  
        printf("Object create error");  
        delete array;  
        return;  
    }  
    //--- add source arrays elements  
    //--- . . .  
    //--- compare with another array  
    int result=array.CompareArray(src);  
    //--- delete arrays  
    delete src;  
    delete array;  
}
```

InsertSort

Inserts element in a sorted array.

```
bool  InsertSort(  
    string  element      // Element to insert  
)
```

Parameters

element

[in] value of the element to be inserted into a sorted array

Return Value

true if successful, false - if you can not insert the element.

Example:

```
//--- example for CArrayString::InsertSort(string)  
#include <Arrays\ArrayString.mqh>  
//---  
void OnStart()  
{  
    CArrayString *array=new CArrayString;  
    //---  
    if(array==NULL)  
    {  
        printf("Object create error");  
        return;  
    }  
    //--- add arrays elements  
    //--- . . .  
    //--- sort array  
    array.Sort();  
    //--- insert element  
    if(!array.InsertSort("ABC"))  
    {  
        printf("Insert error");  
        delete array;  
        return;  
    }  
    //--- delete array  
    delete array;  
}
```

Search

Searches for an element equal to the sample in the sorted array.

```
int Search(  
    string element    // Sample  
    ) const
```

Parameters

element

[in] The sample element to search in the array.

Return Value

The position of the found element, if successful, -1 - if the item was not found.

Example:

```
//--- example for CArrayString::Search(string)  
#include <Arrays\ArrayString.mqh>  
//---  
void OnStart()  
{  
    CArrayString *array=new CArrayString;  
    //---  
    if(array==NULL)  
    {  
        printf("Object create error");  
        return;  
    }  
    //--- add arrays elements  
    //--- . . .  
    //--- sort array  
    array.Sort();  
    //--- search element  
    if(array.Search("ABC")!= -1) printf("Element found");  
    else                        printf("Element not found");  
    //--- delete array  
    delete array;  
}
```


SearchGreat

Searches for an element of more samples in sorted array.

```
int SearchGreat(  
    string element    // Sample  
    ) const
```

Parameters

element

[in] The sample element to search in the array.

Return Value

The position of the found element, if successful, -1 - if the item was not found.

Example:

```
//--- example for CArrayString::SearchGreat(string)  
#include <Arrays\ArrayString.mqh>  
//---  
void OnStart()  
{  
    CArrayString *array=new CArrayString;  
    //---  
    if(array==NULL)  
    {  
        printf("Object create error");  
        return;  
    }  
    //--- add arrays elements  
    //--- . . .  
    //--- sort array  
    array.Sort();  
    //--- search element  
    if(array.SearchGreat("ABC")!= -1) printf("Element found");  
    else                             printf("Element not found");  
    //--- delete array  
    delete array;  
}
```

SearchLess

Searches for an element less than the sample in the sorted array.

```
int SearchLess(  
    string element    // Sample  
    ) const
```

Parameters

element

[in] The sample element to search in the array.

Return Value

The position of the found element, if successful, -1 - if the item was not found.

Example:

```
//--- example for CArrayString:: SearchLess(string)  
#include <Arrays\ArrayString.mqh>  
//---  
void OnStart()  
{  
    CArrayString *array=new CArrayString;  
    //---  
    if(array==NULL)  
    {  
        printf("Object create error");  
        return;  
    }  
    //--- add arrays elements  
    //--- . . .  
    //--- sort array  
    array.Sort();  
    //--- search element  
    if(array.SearchLess("ABC")!=-1) printf("Element found");  
    else                          printf("Element not found");  
    //--- delete array  
    delete array;  
}
```

SearchGreatOrEqual

Searches for an element greater than or equal to the sample in the sorted array.

```
int SearchGreatOrEqual(  
    string element    // Sample  
) const
```

Parameters

element

[in] The sample element to search in the array.

Return Value

The position of the found element, if successful, -1 - if the item was not found.

Example:

```
//--- example for CArrayString:: SearchGreatOrEqual(string)  
#include <Arrays\ArrayString.mqh>  
//---  
void OnStart()  
{  
    CArrayString *array=new CArrayString;  
    //---  
    if(array==NULL)  
    {  
        printf("Object create error");  
        return;  
    }  
    //--- add arrays elements  
    //--- . . .  
    //--- sort array  
    array.Sort();  
    //--- search element  
    if(array.SearchGreatOrEqual("ABC")!= -1) printf("Element found");  
    else                                     printf("Element not found");  
    //--- delete array  
    delete array;  
}
```

SearchLessOrEqual

Searches for an element less than or equal to the sample in the sorted array.

```
int SearchLessOrEqual(  
    string element    // Sample  
    ) const
```

Parameters

element

[in] The sample element to search in the array.

Return Value

The position of the found element, if successful, -1 - if the item was not found.

Example:

```
///--- example for CArrayString:: SearchLessOrEqual(string)  
#include <Arrays\ArrayString.mqh>  
///---  
void OnStart()  
{  
    CArrayString *array=new CArrayString;  
    ///---  
    if(array==NULL)  
    {  
        printf("Object create error");  
        return;  
    }  
    ///--- add arrays elements  
    ///--- . . .  
    ///--- sort array  
    array.Sort();  
    ///--- search element  
    if(array.SearchLessOrEqual("ABC")!= -1) printf("Element found");  
    else printf("Element not found");  
    ///--- delete array  
    delete array;  
}
```

SearchFirst

Finds the first element equal to the sample in the sorted array.

```
int SearchFirst(  
    string element    // Sample  
    ) const
```

Parameters

element

[in] The sample element to search in the array.

Return Value

The position of the found element, if successful, -1 - if the item was not found.

Example:

```
//--- example for CArrayString:: SearchFirst(string)  
#include <Arrays\ArrayString.mqh>  
//---  
void OnStart()  
{  
    CArrayString *array=new CArrayString;  
    //---  
    if(array==NULL)  
    {  
        printf("Object create error");  
        return;  
    }  
    //--- add arrays elements  
    //--- . . .  
    //--- sort array  
    array.Sort();  
    //--- search element  
    if(array.SearchFirst("ABC")!= -1) printf("Element found");  
    else                             printf("Element not found");  
    //--- delete array  
    delete array;  
}
```

SearchLast

Finds the last element equal to the model in sorted array.

```
int SearchLast(  
    string element    // Sample  
    ) const
```

Parameters

element

[in] The sample element to search in the array.

Return Value

The position of the found element, if successful, -1 - if the item was not found.

Example:

```
//--- example for CArrayString:: SearchLast(string)  
#include <Arrays\ArrayString.mqh>  
//---  
void OnStart()  
{  
    CArrayString *array=new CArrayString;  
    //---  
    if(array==NULL)  
    {  
        printf("Object create error");  
        return;  
    }  
    //--- add arrays elements  
    //--- . . .  
    //--- sort array  
    array.Sort();  
    //--- search element  
    if(array.SearchLast("ABC")!= -1) printf("Element found");  
    else                             printf("Element not found");  
    //--- delete array  
    delete array;  
}
```

Save

Saves data array in the file.

```
virtual bool Save(  
    int file_handle    // File handle  
)
```

Parameters

file_handle

[in] Handle to open earlier, with the function FileOpen (...) , binary file.

Return Value

true - if successfully completed, false - if an error.

Example:

```
//--- example for CArrayString::Save(int)  
#include <Arrays\ArrayString.mqh>  
//---  
void OnStart()  
{  
    int file_handle;  
    CArrayString *array=new CArrayString;  
    //---  
    if(array!=NULL)  
    {  
        printf("Object create error");  
        return;  
    }  
    //--- add 100 arrays elements  
    for(int i=0;i<100;i++)  
    {  
        array.Add(IntegerToString(i));  
    }  
    //--- open file  
    file_handle=FileOpen("MyFile.bin",FILE_WRITE|FILE_BIN|FILE_ANSI);  
    if(file_handle>=0)  
    {  
        if(!array.Save(file_handle))  
        {  
            //--- file save error  
            printf("File save: Error %d!",GetLastError());  
            delete array;  
            FileClose(file_handle);  
            //---  
            return;  
        }  
        FileClose(file_handle);  
    }
```

```
    }  
    delete array;  
}
```


Load

Loads data array from a file.

```
virtual bool Load(
    int file_handle    // File handle
)
```

Parameters

file_handle

[in] Handle to open earlier, with the function FileOpen (...) , binary file.

Return Value

true - if successfully completed, false - if an error.

Example:

```
//--- example for CArrayString::Load(int)
#include <Arrays\ArrayString.mqh>
//---
void OnStart()
{
    int file_handle;
    CArrayString *array=new CArrayString;
    //---
    if(array!=NULL)
    {
        printf("Object create error");
        return;
    }
    //--- open file
    file_handle=FileOpen("MyFile.bin",FILE_READ|FILE_BIN|FILE_ANSI);
    if(file_handle>=0)
    {
        if(!array.Load(file_handle))
        {
            //--- file load error
            printf("File load: Error %d!",GetLastError());
            delete array;
            FileClose(file_handle);
            //---
            return;
        }
        FileClose(file_handle);
    }
    //--- use arrays elements
    for(int i=0;i<array.Total();i++)
    {
        printf("Element[%d] = '%s'",i,array.At(i));
```

```
    }  
    delete array;  
}
```

Type

Gets the type identifier of the array.

```
virtual int Type() const
```

Return Value

ID type of the array (for CArrayString - 89) .

Example:

```
//--- example for CArrayString::Type()
#include <Arrays\ArrayString.mqh>
//---
void OnStart()
{
    CArrayString *array=new CArrayString;
    //---
    if(array==NULL)
    {
        printf("Object create error");
        return;
    }
    //--- get array type
    int type=array.Type();
    //--- delete array
    delete array;
}
```

CArrayObj

CArrayObj class is a class of dynamic array of pointers to instances of CObject and his heirs.

Description

Class CArrayObj provides the ability to work with a dynamic array of pointers to instances of [CObject](#) and his heirs. This gives the possibility to work as a multidimensional dynamic arrays of primitive data types, and the more difficult for organized data structures.

In the class implemented the ability to add / insert / delete elements in an array, sort array, searching in sorted array. In addition, the implemented methods of work with the file.

There are certain [subtleties](#) of the class CArrayObj.

Declaration

```
class CArrayObj : public CArray
```

Title

```
#include <Arrays\ArrayObj.mqh>
```

Class Method

Attributes	
FreeMode	Gets the flag memory management
FreeMode	Sets the flag memory management
Memory control	
Reserve	Allocates memory to increase the size of the array
Resize	Sets a new (smaller) size of the array
Shutdown	Clears the array with a total exemption memory array (not element) .
Add methods	
Add	Adds an element to the end of the array
AddArray	Adds an element to the end of the array
Insert	Inserts an element in the array to the specified position
InsertArray	Inserts an array of elements from another array with the specified position
AssignArray	Copies the array elements from another array
Update methods	
Update	Changes the element at the specified position array
Shift	Moves an item from a given position in the array to the specified offset

Delete methods	
Detach	Gets the element from the specified position and removing it from the array
Delete	Removes the element from the specified position array
DeleteRange	Deletes a group of elements from the specified position array
Clear	Removes all elements of the array without the release of the memory array
Access methods	
At	Gets the element from the specified position array
Compare methods	
CompareArray	Compares array with another array
Sorted array operations	
InsertSort	Inserts element in a sorted array
Search	Searches for an element equal to the model in sorted array
SearchGreat	Searches for an element of more samples in sorted array
SearchLess	Searches for an element less than the sample in the sorted array
SearchGreatOrEqual	Searches for an element greater than or equal to the model in sorted array
SearchLessOrEqual	Searches for an element less than or equal to the model in sorted array
SearchFirst	Finds the first element equal to the model in sorted array
SearchLast	Finds the last element equal to the model in sorted array
Input/output	
Save	Saves data array in the file
Load	Loads data array from a file
Type	Gets the type identifier of the array

Derived classes:

- [CIndicator](#)
- [CIndicators](#)

Practical application of arrays are descendants of class CObject (including all classes of the standard library) .

For example, consider the options for two-dimensional array:

```
#include <Arrays\ArrayDouble.mqh>
```

```
#include <Arrays\ArrayObj.mqh>
//---
void OnStart()
{
    int i,j;
    int first_size=10;
    int second_size=100;
    //--- create array
    CArrayObj *array=new CArrayObj;
    CArrayDouble *sub_array;
    //---
    if(array==NULL)
    {
        printf("Object create error");
        return;
    }
    //--- create subarrays
    for(i=0;i<first_size;i++)
    {
        sub_array=new CArrayDouble;
        if(sub_array==NULL)
        {
            delete array;
            printf("Object create error");
            return;
        }
        //--- fill array
        for(j=0;j<second_size;j++)
        {
            sub_array.Add(i*j);
        }
        array.Add(sub_array);
    }
    //--- create array OK
    for(i=0;i<first_size;i++)
    {
        sub_array=array.At(i);
        for(j=0;j<second_size;j++)
        {
            double element=sub_array.At(j);
            //--- use array element
        }
    }
    delete array;
}
```

Subtleties

The class has a mechanism to control dynamic memory, so be careful when working with elements of the array.

Mechanism of memory management can be switched on / off using the method `FreeMode (bool)`. By default, the mechanism is enabled.

Accordingly, there are two options for dealing with the class `CArrayObj`:

1. Mechanism of memory management is enabled. (default)

In this case, `CArrayObj` take responsibility for freeing the memory elements after their removal from the array. In this program the user should not free the array elements.

Example:

```
int i;
//--- Create an array
CArrayObj *array=new CArrayObj;
//--- Fill array elements
for(i=0;i<10;i++) array.Add(new CObject);
//--- Do something
for(i=0;i<array.Total();i++)
{
    CObject *object=array.At(i);
    //--- Action with an element
    . . .
}
//--- Remove the array with the elements
delete array;
```

2. Mechanism of memory management is turned off.

In this case, `CArrayObj` not otvetstvechaet for freeing the memory elements after their removal from the array. In this program the user must free the array elements.

Example:

```
int i;
//--- Create an array
CArrayObj *array=new CArrayObj;
//--- Disable the mechanism of memory management
array.FreeMode(false);
//--- Fill array elements
for(i=0;i<10;i++) array.Add(new CObject);
//--- Do something
for(i=0;i<array.Total();i++)
{
    CObject *object=array.At(i);
    //--- Action with an element
    . . .
}
```

```
    }  
    ///--- Remove array elements  
    while(array.Total()) delete array.Detach();  
    ///--- Remove empty array  
    delete array;
```


FreeMode

Gets the flag memory management.

```
bool FreeMode() const
```

Return Value

Flag of memory management.

Example:

```
//--- example for CArrayObj::FreeMode()
#include <Arrays\ArrayObj.mqh>
//---
void OnStart()
{
    CArrayObj *array=new CArrayObj;
    //---
    if(array==NULL)
    {
        printf("Object create error");
        return;
    }
    //--- get free mode flag
    bool array_free_mode=array.FreeMode();
    //--- delete array
    delete array;
}
```

FreeMode

Sets the flag memory management.

```
void FreeMode(  
    bool mode    // New flag  
)
```

Parameters

mode

[in] New value of the flag memory management.

Return Value

None.

Note

Setting the flag memory management - an important point in the use of class CArrayObj. Since the array elements are pointers to dynamic objects, it is important to determine what to do with them when removing from the array.

If the flag is set, removing an element from the array, the element is automatically deleted by the operator delete. If the flag is not set, it is assumed that a pointer to the deleted object is still somewhere in the user program and will be relieved of it (the program) then.

If the user resets the flag memory management, the user must understand their responsibility for the removal of the array before the completion of the program, because otherwise, is not freed memory occupied by the elements when they create new operator.

When large amounts of data, it could lead, eventually, even to break your terminal. If the user does not reset the flag memory management, there is another "reef".

Using pointers, array, stored somewhere in the local variables, after removing the array will lead to a critical error and crashes the program user. By default, the memory management flag is set, ie the class of the array is responsible for freeing the memory elements.

Example:

```
//--- example for CArrayObj::FreeMode(bool)  
#include <Arrays\ArrayObj.mqh>  
//---  
void OnStart()  
{  
    CArrayObj *array=new CArrayObj;  
    //---  
    if(array==NULL)  
    {  
        printf("Object create error");  
        return;  
    }  
    //--- reset free mode flag  
    array.FreeMode(false);
```

```
//--- use array  
//--- . . .  
//--- delete array  
delete array;  
}
```

Reserve

Allocates memory to increase the size of the array.

```
bool Reserve(  
    int size      // Number  
)
```

Parameters

size

[in] The number of additional elements of the array.

Return Value

true if successful, false - if there was an attempt to seek the amount is less than or equal to zero, or if the array did not increase.

Note

To reduce memory fragmentation, increase the size of the array is made with a step previously given through the method of Step (int) , or 16 (default) .

Example:

```
//--- example for CArrayObj::Reserve(int)  
#include <Arrays\ArrayObj.mqh>  
//---  
void OnStart()  
{  
    CArrayObj *array=new CArrayObj;  
    //---  
    if(array==NULL)  
    {  
        printf("Object create error");  
        return;  
    }  
    if(!array.Reserve(1024))  
    {  
        printf("Reserve error");  
        delete array;  
        return;  
    }  
    //--- use array  
    //--- . . .  
    //--- delete array  
    delete array;  
}
```

Resize

Sets a new (smaller) size of the array.

```
bool Resize(  
    int size      // Size  
)
```

Parameters

size

[in] New size of the array.

Return Value

To reduce fragmentation of memory, change the size of the array is made with a step previously given through the method of Step (int) , or 16 (default) .

Note

Changing the size of array allows to use memory in the optimal way. Excess element located to the right are lost. The memory for lost elements is released or not depending on the mode of memory management.

To reduce fragmentation of memory, change the size of the array is made with a step previously given through the method of Step (int) , or 16 (default) .

Example:

```
//--- example for CArrayObj::Resize(int)  
#include <Arrays\ArrayObj.mqh>  
//---  
void OnStart()  
{  
    CArrayObj *array=new CArrayObj;  
    //---  
    if(array==NULL)  
    {  
        printf("Object create error");  
        return;  
    }  
    //--- add arrays elements  
    //--- . . .  
    //--- resize array  
    if(!array.Resize(10))  
    {  
        printf("Resize error");  
        delete array;  
        return;  
    }  
    //--- delete array  
    delete array;
```

```
}
```

Clear

Removes all elements of the array without the release of the memory array.

```
void Clear()
```

Return Value

No.

Note

If enabled memory management, memory, deleted items are exempt.

Example:

```
//--- example for CArrayObj::Clear()
#include <Arrays\ArrayObj.mqh>
//---
void OnStart()
{
    CArrayObj *array=new CArrayObj;
    //---
    if(array==NULL)
    {
        printf("Object create error");
        return;
    }
    //--- add arrays elements
    //--- . . .
    //--- clear array
    array.Clear();
    //--- delete array
    delete array;
}
```

Shutdown

Clears the array with a total exemption memory array (not element) .

```
bool Shutdown()
```

Return Value

true if successful, false - if an error occurred.

Note

If enabled memory management, memory, deleted items are exempt.

Example:

```
//--- example for CArrayObj::Shutdown()
#include <Arrays\ArrayObj.mqh>
//---
void OnStart()
{
    CArrayObj *array=new CArrayObj;
    //---
    if(array==NULL)
    {
        printf("Object create error");
        return;
    }
    //--- add arrays elements
    //--- . . .
    //--- shutdown array
    if(!array.Shutdown())
    {
        printf("Shutdown error");
        delete array;
        return;
    }
    //--- delete array
    delete array;
}
```


CreateElement

Creates a new array element at the specified position.

```
bool CreateElement(  
    int index    // Position  
)
```

Parameters

index

[in] position in which you want to create a new element.

Return Value

true if successful, false - if you can not create element.

Note

Method CreateElement (int) in class CArrayObj always returns false and does not perform any action. If necessary, in a derived class, method CreateElement (int) should be implemented.

Example:

```
///--- example for CArrayObj::CreateElement(int)  
#include <Arrays\ArrayObj.mqh>  
///---  
void OnStart()  
{  
    int size=100;  
    CArrayObj *array=new CArrayObj;  
    ///---  
    if(array==NULL)  
    {  
        printf("Object create error");  
        return;  
    }  
    ///--- fill array  
    array.Reserve(size);  
    for(int i=0;i<size;i++)  
    {  
        if(!array.CreateElement(i))  
        {  
            printf("Element create error");  
            delete array;  
            return;  
        }  
    }  
    ///--- use array  
    ///--- . . .  
    ///--- delete array
```

```
delete array;  
}
```

Add

Adds an element to the end of the array.

```
bool Add(  
    CObject* element    // Element to add  
)
```

Parameters

element

[in] value of the element to add to the array.

Return Value

true if successful, false - if you can not add an element.

Note

Element is not added to the array if the value for transmit invalid pointer (such as NULL) .

Example:

```
//--- example for CArrayObj::Add(CObject*)  
#include <Arrays\ArrayObj.mqh>  
//---  
void OnStart()  
{  
    CArrayObj *array=new CArrayObj;  
    //---  
    if(array==NULL)  
    {  
        printf("Object create error");  
        return;  
    }  
    //--- add 100 arrays elements  
    for(int i=0;i<100;i++)  
    {  
        if(!array.Add(new CObject))  
        {  
            printf("Element addition error");  
            delete array;  
            return;  
        }  
    }  
    //--- use array  
    //--- . . .  
    //--- delete array  
    delete array;  
}
```

AddArray

Adds to the end of the array elements from another array.

```
bool AddArray(  
    const CArrayObj * src      // Pointer to the source  
)
```

Parameters

src

[in] Pointer to an instance of class [CArrayDouble](#) - source of elements to add.

Return Value

true if successful, false - if you can not add items.

Note

Adding elements of the array in the array is actually copying the pointers. Therefore, when calling the method, there is a "reef" - that there may be a pointer to a dynamic object in more than one variable.

```
//--- example  
extern bool      make_error;  
extern int       error;  
extern CArrayObj *src;  
//--- Create a new instance CArrayObj  
//--- Default memory management is turned on  
CArrayObj *array=new CArrayObj;  
//--- Add (copy) the elements of the array-istochika  
if(array!=NULL)  
    bool result=array.AddArray(src);  
if(make_error)  
{  
    //--- Commit wrongdoing  
    switch(error)  
    {  
        case 0:  
            //--- Remove the array-source, without checking its flag memory management  
            delete src;  
            //--- Result:  
            //--- May appeal  
            //--- For "bitomu" pointer in the array receiver  
            break;  
        case 1:  
            //--- Disable the mechanism of memory management in an array of source  
            if(src.FreeMode()) src.FreeMode(false);  
            //--- But do not remove the array-source  
            //--- Result:  
            //--- After removal of the array, the receiver may appeal  
            //--- For "bitomu" pointer in the array-source
```

```

        break;
    case 2:
        //--- Disable the mechanism of memory management in an array of source
        src.FreeMode(false);
        //--- Disable the mechanism of memory management in an array receiver
        array.FreeMode(false);
        //--- Result:
        //--- After the completion of the program, obtain a "memory leak"
        break;
    }
}
else
{
    //--- Disable the mechanism of memory management in an array of source
    if(src.FreeMode()) src.FreeMode(false);
    //--- Remove the array-source
    delete src;
    //--- Result:
    //--- Treatment for an array-receipient be correct
    //--- Delete the array, the receiver will delete its elements
}

```

Example:

```

//--- example for CArrayObj::AddArray(const CArrayObj*)
#include <Arrays\ArrayObj.mqh>
//---
void OnStart()
{
    CArrayObj *array=new CArrayObj;
    //---
    if(array==NULL)
    {
        printf("Object create error");
        return;
    }
    //--- create source array
    CArrayObj *src=new CArrayObj;
    if(src==NULL)
    {
        printf("Object create error");
        delete array;
        return;
    }
    //--- reset free mode flag
    src.FreeMode(false);
    //--- fill source array
    //--- . . .

```

```
//--- add another array
if(!array.AddArray(src))
{
    printf("Array addition error");
    delete src;
    delete array;
    return;
}
//--- delete source array without elements
delete src;
//--- use array
//--- . . .
//--- delete array
delete array;
}
```

Insert

Inserts an element in the array to the specified position.

```
bool Insert(  
    CObject* element,    // Element to insert  
    int      pos         // Position  
)
```

Parameters

element

[in] value of the element to be inserted into an array

pos

[in] Position in the array to insert

Return Value

true if successful, false - if you can not insert the element.

Note

Element is not added to the array if the value for transmit invalid pointer (such as NULL) .

Example:

```
//--- example for CArrayObj::Insert(CObject*,int)  
#include <Arrays\ArrayObj.mqh>  
//---  
void OnStart()  
{  
    CArrayObj *array=new CArrayObj;  
    //---  
    if(array==NULL)  
    {  
        printf("Object create error");  
        return;  
    }  
    //--- insert elements  
    for(int i=0;i<100;i++)  
    {  
        if(!array.Insert(new CObject,0))  
        {  
            printf("Insert error");  
            delete array;  
            return;  
        }  
    }  
    //--- use array  
    //--- . . .  
    //--- delete array
```

```
delete array;  
}
```


InsertArray

Inserts an array of elements from another array specified position.

```
bool  InsertArray(  
    const CArrayObj*  src,      // Pointer to the source  
    int               pos      // Position  
)
```

Parameters

src

[in] Pointer to an instance of class CArrayObj-source elements to insert.

pos

[in] Position in the array to insert

Return Value

true if successful, false - if you can not paste items.

Note

See: [CArrayObj::AddArray\(const CArrayObj* \)](#).

Example:

```
//--- example for CArrayObj::InsertArray(const CArrayObj*,int)  
#include <Arrays\ArrayObj.mqh>  
//---  
void OnStart()  
{  
    CArrayObj *array=new CArrayObj;  
    //---  
    if(array==NULL)  
    {  
        printf("Object create error");  
        return;  
    }  
    //--- create source array  
    CArrayObj *src=new CArrayObj;  
    if(src==NULL)  
    {  
        printf("Object create error");  
        delete array;  
        return;  
    }  
    //--- reset free mode flag  
    src.FreeMode(false);  
    //--- fill source array  
    //--- . . .  
    //--- insert another array
```

```
if(!array.InsertArray(src,0))
{
    printf("Array inserting error");
    delete src;
    delete array;
    return;
}
//--- delete source array without elements
delete src;
//--- use array
//--- . . .
//--- delete array
delete array;
}
```

AssignArray

Copies the array elements from another array.

```
bool AssignArray(  
    const CArrayObj* src      // Pointer to the source  
)
```

Parameters

src

[in] Pointer to an instance of class CArrayObj - source of elements to copy.

Return Value

true if successful, false - if you can not copy the items.

Note

If the challenge AssignArray array receiver is not empty, all its elements will be removed from the array and, if the flag memory management, memory, deleted items will be released. Array-receiver is an exact copy of the array source. Additionally see [CArrayObj::AddArray\(const CArrayObj*\)](#).

Example:

```
//--- example for CArrayObj::AssignArray(const CArrayObj*)  
#include <Arrays\ArrayObj.mqh>  
//---  
void OnStart()  
{  
    CArrayObj *array=new CArrayObj;  
    //---  
    if(array==NULL)  
    {  
        printf("Object create error");  
        return;  
    }  
    //--- create source array  
    CArrayObj *src=new CArrayObj;  
    if(src==NULL)  
    {  
        printf("Object create error");  
        delete array;  
        return;  
    }  
    //--- reset free mode flag  
    src.FreeMode(false);  
    //--- fill source array  
    //--- . . .  
    //--- assign another array  
    if(!array.AssignArray(src))  
    {
```

```
    printf("Array assigned error");
    delete src;
    delete array;
    return;
}
//--- arrays is identical
//--- delete source array without elements
delete src;
//--- use array
//--- . . .
//--- delete array
delete array;
}
```

Update

Changes the element at the specified position array.

```
bool Update(  
    int      pos,          // Position  
    CObject* element       // Value  
)
```

Parameters

pos

[in] Position of the element in the array to change

element

[in] New value element

Return Value

true if successful, false - if you can not change the element.

Note

The element does not change if we as a parameter to pass an invalid pointer (ie NULL) . If enabled memory management, memory placeholder released.

Example:

```
//--- example for CArrayObj::Update(int,CObject*)  
#include <Arrays\ArrayObj.mqh>  
//---  
void OnStart()  
{  
    CArrayObj *array=new CArrayObj;  
    //---  
    if(array==NULL)  
    {  
        printf("Object create error");  
        return;  
    }  
    //--- add arrays elements  
    //--- . . .  
    //--- update element  
    if(!array.Update(0,new CObject))  
    {  
        printf("Update error");  
        delete array;  
        return;  
    }  
    //--- delete array  
    delete array;  
}
```

Shift

Moves an item from a given position in the array to the specified offset.

```
bool Shift(  
    int pos,          // Position  
    int shift         // Shift  
)
```

Parameters

pos

[in] Position of the moved element array

shift

[in] The value of displacement (both positive and negative) .

Return Value

true if successful, false - if you can not move the item.

Example:

```
//--- example for CArrayObj::Shift(int,int)  
#include <Arrays\ArrayObj.mqh>  
//---  
void OnStart()  
{  
    CArrayObj *array=new CArrayObj;  
    //---  
    if(array==NULL)  
    {  
        printf("Object create error");  
        return;  
    }  
    //--- add arrays elements  
    //--- . . .  
    //--- shift element  
    if(!array.Shift(10,-5))  
    {  
        printf("Shift error");  
        delete array;  
        return;  
    }  
    //--- delete array  
    delete array;  
}
```

Detach

Remove an item from a given position in the array.

```
CObject* Detach(  
    int pos    // Position  
)
```

Parameters

pos

[in] Position of the seized item in the array.

Return Value

Pointer to the removal of elements in case of success, NULL - if you can not remove the element.

Note

When removed from the array element is not removed in any state of the flag memory management.
Pointer to the array element withdrawn from the ingredients of the release after use.

Example:

```
///--- example for CArrayObj::Detach(int)  
#include <Arrays\ArrayObj.mqh>  
///---  
void OnStart()  
{  
    CArrayObj *array=new CArrayObj;  
    ///---  
    if(array==NULL)  
    {  
        printf("Object create error");  
        return;  
    }  
    ///--- add arrays elements  
    ///--- . . .  
    CObject *object=array.Detach(0);  
    if(object==NULL)  
    {  
        printf("Detach error");  
        delete array;  
        return;  
    }  
    ///--- use element  
    ///--- . . .  
    ///--- delete element  
    delete object;  
    ///--- delete array  
    delete array;
```

```
}
```


Delete

Removes the element from the given position in the array.

```
bool Delete(  
    int pos    // Position  
)
```

Parameters

pos

[in] position removes the element in the array.

Return Value

true if successful, false - if you can not remove the element.

Note

If enabled memory management, memory, deleted items are exempt.

Example:

```
///--- example for CArrayObj::Delete(int)  
#include <Arrays\ArrayObj.mqh>  
///---  
void OnStart()  
{  
    CArrayObj *array=new CArrayObj;  
    ///---  
    if(array==NULL)  
    {  
        printf("Object create error");  
        return;  
    }  
    ///--- add arrays elements  
    ///--- . . .  
    if(!array.Delete(0))  
    {  
        printf("Delete error");  
        delete array;  
        return;  
    }  
    ///--- delete array  
    delete array;  
}
```

DeleteRange

Deletes a group of elements from a given position in the array.

```
bool DeleteRange(  
    int from,      // Position of the first element  
    int to         // Position of last element  
)
```

Parameters

from

[in] Position of the first removes the element in the array.

to

[in] Position of the last deleted element in the array.

Return Value

true if successful, false - if you can not remove elements.

Note

If enabled memory management, memory, deleted items are exempt.

Example:

```
//--- example for CArrayObj::DeleteRange(int,int)  
#include <Arrays\ArrayObj.mqh>  
//---  
void OnStart()  
{  
    CArrayObj *array=new CArrayObj;  
    //---  
    if(array==NULL)  
    {  
        printf("Object create error");  
        return;  
    }  
    //--- add arrays elements  
    //--- . . .  
    //--- delete elements  
    if(!array.DeleteRange(0,10))  
    {  
        printf("Delete error");  
        delete array;  
        return;  
    }  
    //--- delete array  
    delete array;  
}
```

At

Gets the element from the given position in the array.

```
CObject* At(  
    int pos    // Position  
)
```

Parameters

pos

[in] Position of the desired element in the array.

Return Value

The value of the element, if successful, NULL-if there was an attempt to get an element of non-existent position.

Example:

```
//--- example for CArrayObj::At(int)  
#include <Arrays\ArrayObj.mqh>  
//---  
void OnStart()  
{  
    CArrayObj *array=new CArrayObj;  
    //---  
    if(array==NULL)  
    {  
        printf("Object create error");  
        return;  
    }  
    //--- add elements  
    //--- . . .  
    for(int i=0;i<array.Total();i++)  
    {  
        CObject *result=array.At(i);  
        if(result==NULL)  
        {  
            //--- Error reading from array  
            printf("Get element error");  
            delete array;  
            return;  
        }  
        //--- use element  
        //--- . . .  
    }  
    delete array;  
}
```

CompareArray

Compares array with another array.

```
bool CompareArray(  
    const CArrayObj* src      // Pointer to the source  
    ) const
```

Parameters

src

[in] Pointer to an instance of class CArrayObj - the source of elements for comparison.

Return Value

true if arrays are equal, false - if not.

Example:

```
//--- example for CArrayObj::CompareArray(const CArrayObj*)  
#include <Arrays\ArrayObj.mqh>  
//---  
void OnStart()  
{  
    CArrayObj *array=new CArrayObj;  
    //---  
    if(array==NULL)  
    {  
        printf("Object create error");  
        return;  
    }  
    //--- create source array  
    CArrayObj *src=new CArrayObj;  
    if(src==NULL)  
    {  
        printf("Object create error");  
        delete array;  
        return;  
    }  
    //--- fill source array  
    //--- . . .  
    //--- compare with another array  
    int result=array.CompareArray(src);  
    //--- delete arrays  
    delete src;  
    delete array;  
}
```

InsertSort

Inserts element in a sorted array.

```
bool  InsertSort(  
    CObject*  element    // Element to insert  
)
```

Parameters

element

[in] Value of the element to be inserted into a sorted array

Return Value

true if successful, false - if you can not insert the element.

Note

Element is not added to the array if the value for transmit invalid pointer (such as NULL) .

Example:

```
///--- example for CArrayObj::InsertSort(CObject*)  
#include <Arrays\ArrayObj.mqh>  
///---  
void OnStart()  
{  
    CArrayObj *array=new CArrayObj;  
    ///---  
    if(array==NULL)  
    {  
        printf("Object create error");  
        return;  
    }  
    ///--- add arrays elements  
    ///--- . . .  
    ///--- sort array  
    array.Sort();  
    ///--- insert element  
    if(!array.InsertSort(new CObject))  
    {  
        printf("Insert error");  
        delete array;  
        return;  
    }  
    ///--- delete array  
    delete array;  
}
```

Search

Searches for an element equal to the sample in the sorted array.

```
int Search(  
    CObject* element    // Sample  
    ) const
```

Parameters

element

[in] The sample element to search in the array.

Return Value

The position of the found element, if successful, -1 - if the item was not found.

Example:

```
//--- example for CArrayObj::Search(CObject*)  
#include <Arrays\ArrayObj.mqh>  
//---  
void OnStart()  
{  
    CArrayObj *array=new CArrayObj;  
    //---  
    if(array==NULL)  
    {  
        printf("Object create error");  
        return;  
    }  
    //--- add arrays elements  
    //--- . . .  
    //--- sort array  
    array.Sort();  
    //--- create sample  
    CObject *sample=new CObject;  
    if(sample==NULL)  
    {  
        printf("Sample create error");  
        delete array;  
        return;  
    }  
    //--- set sample attributes  
    //--- . . .  
    //--- search element  
    if(array.Search(sample)!=-1) printf("Element found");  
    else                        printf("Element not found");  
    //--- delete array  
    delete array;  
}
```

SearchGreat

Searches for an element of more samples in sorted array.

```
int SearchGreat(  
    CObject* element    // Sample  
    ) const
```

Parameters

element

[in] The sample element to search in the array.

Return Value

The position of the found element, if successful, -1 - if the item was not found.

Example:

```
//--- example for CArrayObj::SearchGreat(CObject*)  
#include <Arrays\ArrayObj.mqh>  
//---  
void OnStart()  
{  
    CArrayObj *array=new CArrayObj;  
    //---  
    if(array==NULL)  
    {  
        printf("Object create error");  
        return;  
    }  
    //--- add arrays elements  
    //--- . . .  
    //--- sort array  
    array.Sort();  
    //--- create sample  
    CObject *sample=new CObject;  
    if(sample==NULL)  
    {  
        printf("Sample create error");  
        delete array;  
        return;  
    }  
    //--- set sample attributes  
    //--- . . .  
    //--- search element  
    if(array.SearchGreat(sample)!=-1) printf("Element found");  
    else                             printf("Element not found");  
    //--- delete array  
    delete array;  
}
```

SearchLess

Searches for an element less than the sample in the sorted array.

```
int SearchLess(  
    CObject* element    // Sample  
    ) const
```

Parameters

element

[in] The sample element to search in the array.

Return Value

The position of the found element, if successful, -1 - if the item was not found.

Example:

```
//--- example for CArrayObj:: SearchLess(CObject*)  
#include <Arrays\ArrayObj.mqh>  
//---  
void OnStart()  
{  
    CArrayObj *array=new CArrayObj;  
    //---  
    if(array==NULL)  
    {  
        printf("Object create error");  
        return;  
    }  
    //--- add arrays elements  
    //--- . . .  
    //--- sort array  
    array.Sort();  
    //--- create sample  
    CObject *sample=new CObject;  
    if(sample==NULL)  
    {  
        printf("Sample create error");  
        delete array;  
        return;  
    }  
    //--- set sample attributes  
    //--- . . .  
    //--- search element  
    if(array.SearchLess(sample)!=-1) printf("Element found");  
    else                             printf("Element not found");  
    //--- delete array  
    delete array;  
}
```


SearchGreatOrEqual

Searches for an element greater than or equal to the sample in the sorted array.

```
int SearchGreatOrEqual(  
    CObject* element // Sample  
    ) const
```

Parameters

element

[in] The sample element to search in the array.

Return Value

The position of the found element, if successful, -1 - if the item was not found.

Example:

```
//--- example for CArrayObj::SearchGreatOrEqual(CObject*)  
#include <Arrays\ArrayObj.mqh>  
//---  
void OnStart()  
{  
    CArrayObj *array=new CArrayObj;  
    //---  
    if(array==NULL)  
    {  
        printf("Object create error");  
        return;  
    }  
    //--- add arrays elements  
    //--- . . .  
    //--- sort array  
    array.Sort();  
    //--- create sample  
    CObject *sample=new CObject;  
    if(sample==NULL)  
    {  
        printf("Sample create error");  
        delete array;  
        return;  
    }  
    //--- set sample attributes  
    //--- . . .  
    //--- search element  
    if(array.SearchGreatOrEqual(sample)!=-1) printf("Element found");  
    else                                     printf("Element not found");  
    //--- delete array  
    delete array;  
}
```

SearchLessOrEqual

Searches for an element less than or equal to the sample in the sorted array.

```
int SearchLessOrEqual(  
    CObject* element    // Sample  
    ) const
```

Parameters

element

[in] The sample element to search in the array.

Return Value

The position of the found element, if successful, -1 - if the item was not found.

Example:

```
//--- example for CArrayObj:: SearchLessOrEqual(CObject*)  
#include <Arrays\ArrayObj.mqh>  
//---  
void OnStart()  
{  
    CArrayObj *array=new CArrayObj;  
    //---  
    if(array==NULL)  
    {  
        printf("Object create error");  
        return;  
    }  
    //--- add arrays elements  
    //--- . . .  
    //--- sort array  
    array.Sort();  
    //--- create sample  
    CObject *sample=new CObject;  
    if(sample==NULL)  
    {  
        printf("Sample create error");  
        delete array;  
        return;  
    }  
    //--- set sample attributes  
    //--- . . .  
    //--- search element  
    if(array.SearchLessOrEqual(sample)!=-1) printf("Element found");  
    else                                  printf("Element not found");  
    //--- delete array  
    delete array;  
}
```

SearchFirst

Finds the first element equal to the sample in the sorted array.

```
int SearchFirst(  
    CObject* element    // Sample  
    ) const
```

Parameters

element

[in] The sample element to search in the array.

Return Value

The position of the found element, if successful, -1 - if the item was not found.

Example:

```
//--- example for CArrayObj::SearchFirst(CObject*)  
#include <Arrays\ArrayObj.mqh>  
//---  
void OnStart()  
{  
    CArrayObj *array=new CArrayObj;  
    //---  
    if(array==NULL)  
    {  
        printf("Object create error");  
        return;  
    }  
    //--- add arrays elements  
    //--- . . .  
    //--- sort array  
    array.Sort();  
    //--- create sample  
    CObject *sample=new CObject;  
    if(sample==NULL)  
    {  
        printf("Sample create error");  
        delete array;  
        return;  
    }  
    //--- set sample attributes  
    //--- . . .  
    //--- search element  
    if(array.SearchFirst(sample)!=-1) printf("Element found");  
    else                             printf("Element not found");  
    //--- delete array  
    delete array;  
}
```

SearchLast

Finds the last element equal to the model in sorted array.

```
int SearchLast(  
    CObject* element    // Sample  
    ) const
```

Parameters

element

[in] The sample element to search in the array.

Return Value

The position of the found element, if successful, -1 - if the item was not found.

Example:

```
//--- example for CArrayObj:: SearchLast(CObject*)  
#include <Arrays\ArrayObj.mqh>  
//---  
void OnStart()  
{  
    CArrayObj *array=new CArrayObj;  
    //---  
    if(array==NULL)  
    {  
        printf("Object create error");  
        return;  
    }  
    //--- add arrays elements  
    //--- . . .  
    //--- sort array  
    array.Sort();  
    //--- create sample  
    CObject *sample=new CObject;  
    if(sample==NULL)  
    {  
        printf("Sample create error");  
        delete array;  
        return;  
    }  
    //--- set sample attributes  
    //--- . . .  
    //--- search element  
    if(array.SearchLast(sample)!=-1) printf("Element found");  
    else                             printf("Element not found");  
    //--- delete array  
    delete array;  
}
```

Save

Saves data array in the file.

```
virtual bool Save(  
    int file_handle    // File handle  
)
```

Parameters

file_handle

[in] Handle to previously opened by FileOpen (...) function binary file.

Return Value

true - if successfully completed, false - if an error.

Example:

```
//--- example for CArrayObj::Save(int)  
#include <Arrays\ArrayObj.mqh>  
//---  
void OnStart()  
{  
    int file_handle;  
    CArrayObj *array=new CArrayObj;  
    //---  
    if(array!=NULL)  
    {  
        printf("Object create error");  
        return;  
    }  
    //--- add arrays elements  
    //--- . . .  
    //--- open file  
    file_handle=FileOpen("MyFile.bin",FILE_WRITE|FILE_BIN|FILE_ANSI);  
    if(file_handle>=0)  
    {  
        if(!array.Save(file_handle))  
        {  
            //--- file save error  
            printf("File save: Error %d!",GetLastError());  
            delete array;  
            FileClose(file_handle);  
            //---  
            return;  
        }  
        FileClose(file_handle);  
    }  
    delete array;  
}
```

Load

Loads data array from a file.s

```
virtual bool Load(  
    int file_handle    // File handle  
)
```

Parameters

file_handle

[in] Handle to open earlier, with the function `FileOpen (...)` , binary file.

Return Value

true - if successfully completed, false - if an error.

Note

When reading from the file array to create each element of the method is called [CArrayObj::CreateElement\(int\)](#) .

Example:

```
//--- example for CArrayObj::Load(int)  
#include <Arrays\ArrayObj.mqh>  
//---  
void OnStart()  
{  
    int file_handle;  
    CArrayObj *array=new CArrayObj;  
    //---  
    if(array!=NULL)  
    {  
        printf("Object create error");  
        return;  
    }  
    //--- open file  
    file_handle=FileOpen("MyFile.bin",FILE_READ|FILE_BIN|FILE_ANSI);  
    if(file_handle>=0)  
    {  
        if(!array.Load(file_handle))  
        {  
            //--- file load error  
            printf("File load: Error %d!",GetLastError());  
            delete array;  
            FileClose(file_handle);  
            //---  
            return;  
        }  
        FileClose(file_handle);
```

```
    }  
    ///--- use arrays elements  
    ///--- . . .  
    ///--- delete array  
    delete array;  
}
```

Type

Gets the type identifier of the array.

```
virtual int Type() const
```

Return Value

ID type of the array (for CArrayObj - 7778) .

Example:

```
//--- example for CArrayObj::Type()
#include <Arrays\ArrayObj.mqh>
//---
void OnStart()
{
    CArrayObj *array=new CArrayObj;
    //---
    if(array==NULL)
    {
        printf("Object create error");
        return;
    }
    //--- get array type
    int type=array.Type();
    //--- delete array
    delete array;
}
```


CList

CList Class is a class of dynamic list of instances of the class CObject and his heirs.

Description

Class CList provides the ability to work with a list of instances of [CObject](#) and his heirs. In the class implemented the ability to add / insert / delete items in the list, sort the list, search in sorted list. In addition, the implemented methods of work with the file.

There are some subtleties of working with the class CList. The class has a mechanism to control dynamic memory, so be careful when working with elements of the list.

[Subtleties](#) of the mechanism of memory management similar to those described in CArrayObj.

Declaration

```
class CList : public CObject
```

Title

```
#include <Arrays\List.mqh>
```

Class Methods

Attributes	
FreeMode	Gets the flag memory management when deleting list items.
FreeMode	Sets the flag memory management when deleting items in the list
Total	Gets the number of elements in the list
IsSorted	Gets flag sorted list
SortMode	Gets the version of the sorting
Create methods	
CreateElement	Creates a new item to the list
Add methods	
Add	Adds element to the end of the list
Insert	Inserts element in the list in the specified position
Delete methods	
DetachCurrent	Remove an item from the current position of the list without deleting it "physically"
DeleteCurrent	Removes the element from the current position in the list
Delete	Removes the element from the specified position in the list
Clear	Removes all list items

Navigation	
IndexOf	Gets the index of the list item
GetNodeAtIndex	Gets an item with the specified index of the list
GetFirstNode	Gets the first element of the list
GetPrevNode	Gets the previous element list
GetCurrentNode	Gets the current list item
GetNextNode	Gets the next item in the list
GetLastNode	Gets the last item
Ordering methods	
Sort	Sort list
MoveToIndex	Moves the current item list to the specified position
Exchange	Changes elements of the list seats
Compare methods	
CompareList	Compares the list with another list
Search methods	
Search	Searches for an element equal to the model in sorted list
Input/output	
virtual Save	Saves data in the file list
virtual Load	Loads data from file list
virtual Type	Gets the type identifier list

FreeMode

Gets the flag memory management when deleting list items.

```
bool FreeMode() const
```

Return Value

Flag of memory management.

Example:

```
//--- example for CList::FreeMode()
#include <Arrays\List.mqh>
//---
void OnStart()
{
    CList *list=new CList;
    //---
    if(list==NULL)
    {
        printf("Object create error");
        return;
    }
    //--- get free mode flag
    bool list_free_mode=list.FreeMode();
    //--- delete list
    delete list;
}
```

FreeMode

Sets the flag memory management when deleting list items.

```
void FreeMode(  
    bool mode    // New value  
)
```

Parameters

mode

[in] New value of the flag memory management.

Note

Setting the flag memory management - an important point in the use of class CList. Since the elements of the list are pointers to dynamic objects, it is important to determine what to do with them when you delete from the list. If the flag is set, then when you delete an item from the list, the item is automatically deleted by the operator delete. If the flag is not set, it is assumed that a pointer to the deleted object is still somewhere in the user program and will be relieved of it (the program) then.

If the user resets the flag memory management, the user must understand their responsibility for the removal of items in the list before completing the program, because otherwise, is not freed memory occupied by the elements when they create new operator. When large amounts of data, it could lead, eventually, even to break your terminal.

If the user does not reset the flag memory management, there is another "reef". Using pointers-list items that are stored somewhere in the local variables, after removing the list, will lead to a critical error and crashes the program user. By default, the memory management flag is set, ie the class list, is responsible for freeing the memory elements.

Example:

```
//--- example for CList::FreeMode(bool)  
#include <Arrays\List.mqh>  
//---  
void OnStart()  
{  
    CList *list=new CList;  
    //---  
    if(list==NULL)  
    {  
        printf("Object create error");  
        return;  
    }  
    //--- reset free mode flag  
    list.FreeMode(false);  
    //--- use list  
    //--- . . .  
    //--- delete list  
    delete list;  
}
```

Total

Gets the number of elements in the list.

```
int Total() const
```

Return Value

Number of elements in the list.

Example:

```
//--- example for CList::Total()
#include <Arrays\List.mqh>
//---
void OnStart()
{
    CList *list=new CList;
    //---
    if(list==NULL)
    {
        printf("Object create error");
        return;
    }
    //--- check total
    int total=list.Total();
    //--- use list
    //--- ...
    //--- delete list
    delete list;
}
```

IsSorted

Gets the flag sorted list.

```
bool IsSorted(  
    int mode=0      // Sorting mode  
) const
```

Parameters

mode=0

[in] Tested version sorting

Return Value

Flag of the sorted list. If the list is sorted by the specified option? true, otherwise? false.

Note

Flag of the sorted list can not be changed directly. Flag set by Sort (int) and resets any methods to add / insert.

Example:

```
//--- example for CList::IsSorted()  
#include <Arrays\List.mqh>  
//---  
void OnStart()  
{  
    CList *list=new CList;  
    //---  
    if(list==NULL)  
    {  
        printf("Object create error");  
        return;  
    }  
    //--- check sorted  
    if(list.IsSorted(0))  
    {  
        //--- use methods for sorted list  
        //--- ...  
    }  
    //--- delete list  
    delete list;  
}
```

SortMode

Gets the version of the sort.

```
int SortMode() const
```

Return Value

Option sorting, or -1 if the list is not sorted.

Example:

```
//--- example for CList::SortMode()
#include <Arrays\List.mqh>
//---
void OnStart()
{
    CList *list=new CList;
    //---
    if(list==NULL)
    {
        printf("Object create error");
        return;
    }
    //--- check sort mode
    int sort_mode=list.SortMode();
    //--- use list
    //--- ...
    //--- delete list
    delete list;
}
```

CreateElement

Creates a new item to the list.

```
CObject* CreateElement()
```

Return Value

Pointer to the newly created element, if successful, NULL - if you can not create element.

Note

Method CreateElement () in class CList always returns NULL and does not perform any actions. If necessary, in a derived class, method CreateElement () should be implemented.

Example:

```
//--- example for CList::CreateElement(int)
#include <Arrays\List.mqh>
//---
void OnStart()
{
    int    size=100;
    CList *list=new CList;
    //---
    if(list==NULL)
    {
        printf("Object create error");
        return;
    }
    //--- fill list
    for(int i=0;i<size;i++)
    {
        CObject *object=list.CreateElement();
        if(object==NULL)
        {
            printf("Element create error");
            delete list;
            return;
        }
        list.Add(object);
    }
    //--- use list
    //--- . . .
    //--- delete list
    delete list;
}
```


Add

Adds an element to the end of the list.

```
int Add(  
    CObject* element    // Element to add  
)
```

Parameters

element

[in] Value of the element to add to the list.

Return Value

If successful, it returns the index of added element, or -1 in the case of error.

Note

Element is not added to the list, if the parameter does not pass valid pointer (ie NULL) .

Example:

```
//--- example for CList::Add(CObject*)  
#include <Arrays\List.mqh>  
//---  
void OnStart()  
{  
    CList *list=new CList;  
    //---  
    if(list==NULL)  
    {  
        printf("Object create error");  
        return;  
    }  
    //--- add 100 elements  
    for(int i=0;i<100;i++)  
    {  
        if(list.Add(new CObject)==-1)  
        {  
            printf("Element addition error");  
            delete list;  
            return;  
        }  
    }  
    //--- use list  
    //--- . . .  
    //--- delete list  
    delete list;  
}
```

Insert

Inserts element in the list in the specified position.

```
int Insert(  
    CObject* element,    // Element to insert  
    int      pos         // Position  
)
```

Parameters

element

[in] value of the element to insert in the list

pos

[in] Position in the list to insert

Return Value

If successful, it returns the index of inserted element, or -1 in the case of error.

Note

Element is not added to the list, if the parameter does not pass valid pointer (ie NULL) .

Example:

```
//--- example for CList::Insert(CObject*,int)  
#include <Arrays\List.mqh>  
//---  
void OnStart()  
{  
    CList *list=new CList;  
    //---  
    if(list==NULL)  
    {  
        printf("Object create error");  
        return;  
    }  
    //--- insert 100 elements  
    for(int i=0;i<100;i++)  
    {  
        if(list.Insert(new CObject,0)==-1)  
        {  
            printf("Element insert error");  
            delete list;  
            return;  
        }  
    }  
    //--- use list  
    //--- . . .  
    //--- delete list
```

```
delete list;  
}
```

DetachCurrent

Extracts an element from the current position without its "physical" deletion.

```
CObject* DetachCurrent()
```

Return Value

Pointer to the removal of elements in case of success, NULL - if you can not remove the element.

Note

When removed from the list, the item is not removed in any state of the flag memory management.
Pointer to withdraw from the list of ingredients of the release element after use.

Example:

```
//--- example for CList::DetachCurrent()
#include <Arrays\List.mqh>
//---
void OnStart()
{
    CList *list=new CList;
    //---
    if(list==NULL)
    {
        printf("Object create error");
        return;
    }
    //--- add list elements
    //--- . . .
    CObject *object=list.DetachCurrent();
    if(object==NULL)
    {
        printf("Detach error");
        delete list;
        return;
    }
    //--- use element
    //--- . . .
    //--- delete element
    delete object;
    //--- delete list
    delete list;
}
```

DeleteCurrent

Removes the element from the current position in the list.

```
bool DeleteCurrent()
```

Return Value

true if successful, false - if you can not remove the element.

Note

If enabled memory management, memory, removes the element is released.

Example:

```
//--- example for CList::DeleteCurrent()
#include <Arrays\List.mqh>
//---
void OnStart()
{
    CList *list=new CList;
    //---
    if(list==NULL)
    {
        printf("Object create error");
        return;
    }
    //--- add list elements
    //--- . . .
    if(!list.DeleteCurrent())
    {
        printf("Delete error");
        delete list;
        return;
    }
    //--- delete list
    delete list;
}
```

Delete

Removes the element from the given position in the list.

```
bool Delete(  
    int pos    // Position  
)
```

Parameters

pos

[in] position removes the element in the list.

Return Value

true if successful, false - if you can not remove the element.

Note

If enabled memory management, memory, removes the element is released.

Example:

```
///--- example for CList::Delete(int)  
#include <Arrays\List.mqh>  
///---  
void OnStart()  
{  
    CList *list=new CList;  
    ///---  
    if(list==NULL)  
    {  
        printf("Object create error");  
        return;  
    }  
    ///--- add list elements  
    ///--- . . .  
    if(!list.Delete(0))  
    {  
        printf("Delete error");  
        delete list;  
        return;  
    }  
    ///--- delete list  
    delete list;  
}
```

Clear

Removes all elements of the list.

```
void Clear()
```

Note

If enabled memory management, memory, deleted items are exempt.

Example:

```
//--- example for CList::Clear()
#include <Arrays\List.mqh>
//---
void OnStart()
{
    CList *list=new CList;
    //---
    if(list==NULL)
    {
        printf("Object create error");
        return;
    }
    //--- add list elements
    //--- . . .
    //--- clear list
    list.Clear();
    //--- delete list
    delete list;
}
```

IndexOf

Gets the index of the list item.

```
int IndexOf(  
    CObject* element    // Pointer to the element  
)
```

Parameters

element

[in] Pointer to the list item.

Return Value

Index item in the list, or -1.

Example:

```
//--- example for CList::IndexOf(CObject*)  
#include <Arrays\List.mqh>  
//---  
void OnStart()  
{  
    CList *list=new CList;  
    //---  
    if(list==NULL)  
    {  
        printf("Object create error");  
        return;  
    }  
    CObject *object=new CObject;  
    if(object==NULL)  
    {  
        printf("Element create error");  
        delete list;  
        return;  
    }  
    if(list.Add(object))  
    {  
        int pos=list.IndexOf(object);  
    }  
    //--- delete list  
    delete list;  
}
```


GetNodeAtIndex

Gets an item with the specified index of the list.

```
CObject* GetNodeAtIndex(  
    int pos      // position  
)
```

Parameters

pos

[in] item index in the list.

Returned value

Pointer to the item in case of success, NULL - if you can not get a pointer.

Example:

```
//--- example for CList::GetNodeAtIndex(int)  
#include <Arrays\List.mqh>  
//---  
void OnStart()  
{  
    CList *list=new CList;  
    //---  
    if(list==NULL)  
    {  
        printf("Object create error");  
        return;  
    }  
    //--- add list elements  
    //--- . . .  
    CObject *object=list.GetNodeAtIndex(10);  
    if(object==NULL)  
    {  
        printf("Get node error");  
        delete list;  
        return;  
    }  
    //--- use element  
    //--- . . .  
    //--- do not delete element  
    //--- delete list  
    delete list;  
}
```

GetFirstNode

Gets the first element of the list.

```
CObject* GetFirstNode()
```

Return Value

Pointer to the first item in case of success, NULL - if you can not get a pointer.

Example:

```
//--- example for CList::GetFirstNode()
#include <Arrays\List.mqh>
//---
void OnStart()
{
    CList *list=new CList;
    //---
    if(list==NULL)
    {
        printf("Object create error");
        return;
    }
    //--- add list elements
    //--- . . .
    CObject *object=list.GetFirstNode();
    if(object==NULL)
    {
        printf("Get node error");
        delete list;
        return;
    }
    //--- use element
    //--- . . .
    //--- do not delete element
    //--- delete list
    delete list;
}
```

GetPrevNode

Gets the previous element of the list.

```
CObject* GetPrevNode()
```

Return Value

Pointer to the previous element, if successful, NULL - if you can not get a pointer.

Example:

```
//--- example for CList::GetPrevNode()
#include <Arrays\List.mqh>
//---
void OnStart()
{
    CList *list=new CList;
    //---
    if(list==NULL)
    {
        printf("Object create error");
        return;
    }
    //--- add list elements
    //--- . . .
    CObject *object=list.GetPrevNode();
    if(object==NULL)
    {
        printf("Get node error");
        delete list;
        return;
    }
    //--- use element
    //--- . . .
    //--- do not delete element
    //--- delete list
    delete list;
}
```

GetCurrentNode

Gets the current list item.

```
CObject* GetCurrentNode()
```

Return Value

Pointer to the current item, if successful, NULL - if you can not get a pointer.

Example:

```
//--- example for CList::GetCurrentNode()
#include <Arrays\List.mqh>
//---
void OnStart()
{
    CList *list=new CList;
    //---
    if(list==NULL)
    {
        printf("Object create error");
        return;
    }
    //--- add list elements
    //--- . . .
    CObject *object=list.GetCurrentNode();
    if(object==NULL)
    {
        printf("Get node error");
        delete list;
        return;
    }
    //--- use element
    //--- . . .
    //--- do not delete element
    //--- delete list
    delete list;
}
```

GetNextNode

Gets the next item in the list.

```
CObject* GetNextNode()
```

Return Value

Pointer to the next item if successful, NULL - if you can not get a pointer.

Example:

```
//--- example for CList::GetNextNode()
#include <Arrays\List.mqh>
//---
void OnStart()
{
    CList *list=new CList;
    //---
    if(list==NULL)
    {
        printf("Object create error");
        return;
    }
    //--- add list elements
    //--- . . .
    CObject *object=list.GetNextNode();
    if(object==NULL)
    {
        printf("Get node error");
        delete list;
        return;
    }
    //--- use element
    //--- . . .
    //--- do not delete element
    //--- delete list
    delete list;
}
```

GetLastNode

Gets the last element of the list.

```
CObject* GetLastNode()
```

Return Value

Pointer to the last element in the case of success, NULL - if you can not get a pointer.

Example:

```
//--- example for CList::GetLastNode()
#include <Arrays\List.mqh>
//---
void OnStart()
{
    CList *list=new CList;
    //---
    if(list==NULL)
    {
        printf("Object create error");
        return;
    }
    //--- add list elements
    //--- . . .
    CObject *object=list.GetLastNode();
    if(object==NULL)
    {
        printf("Get node error");
        delete list;
        return;
    }
    //--- use element
    //--- . . .
    //--- do not delete element
    //--- delete list
    delete list;
}
```

Sort

Sorts a list.

```
void Sort(  
    int mode    // Sorting mode  
)
```

Parameters

mode

[in] Sorting mode.

Return Value

No.

Note

Sorting the list is always in ascending order.

Example:

```
//--- example for CList::Sort(int)  
#include <Arrays\List.mqh>  
//---  
void OnStart()  
{  
    CList *list=new CList;  
    //---  
    if(list==NULL)  
    {  
        printf("Object create error");  
        return;  
    }  
    //--- sorting by mode 0  
    list.Sort(0);  
    //--- use list  
    //--- ...  
    //--- delete list  
    delete list;  
}
```

MoveToIndex

Moves the current item list to the specified position.

```
bool MoveToIndex(  
    int pos      // Position  
)
```

Parameters

pos

[in] Position in the list to move.

Return Value

true if successful, false - if you can not move the item.

Example:

```
///--- example for CList::MoveToIndex(int)  
#include <Arrays\List.mqh>  
///---  
void OnStart()  
{  
    CList *list=new CList;  
    ///---  
    if(list==NULL)  
    {  
        printf("Object create error");  
        return;  
    }  
    ///--- move current node to begin  
    list.MoveToIndex(0);  
    ///--- use list  
    ///--- . . .  
    ///--- delete list  
    delete list;  
}
```


Exchange

Changes elements of the list seats.

```
bool Exchange(  
    CObject* node1,    // List item  
    CObject* node2     // List item  
)
```

Parameters

node1

[in] List item

node2

[in] List item

Return Value

true if successful, false - if you can not change the elements in some places.

Example:

```
//--- example for CList::Exchange(CObject*,CObject*)  
#include <Arrays\List.mqh>  
//---  
void OnStart()  
{  
    CList *list=new CList;  
    //---  
    if(list==NULL)  
    {  
        printf("Object create error");  
        return;  
    }  
    //--- exchange  
    list.Exchange(list.GetFirstNode(),list.GetLastNode());  
    //--- use list  
    //--- . . .  
    //--- delete list  
    delete list;  
}
```

CompareList

Compares the list with another list.

```
bool CompareList(  
    CList* list    // With whom we compare  
)
```

Parameters

list

[in] A pointer to an instance of class CList-source elements for comparison.

Return Value

true if the lists are equal, false - if not.

Example:

```
//--- example for CList::CompareList(const CList*)  
#include <Arrays\List.mqh>  
//---  
void OnStart()  
{  
    CList *list=new CList;  
    //---  
    if(list==NULL)  
    {  
        printf("Object create error");  
        return;  
    }  
    //--- create source list  
    CList *src=new CList;  
    if(src==NULL)  
    {  
        printf("Object create error");  
        delete list;  
        return;  
    }  
    //--- fill source list  
    //--- . . .  
    //--- compare with another list  
    bool result=list.CompareList(src);  
    //--- delete lists  
    delete src;  
    delete list;  
}
```

Search

Searches for an element equal to the model in the sorted list.

```
CObject* Search(  
    CObject* element    // Sample  
)
```

Parameters

element

[in] Sample cell to search for in the list.

Return Value

Pointer to the found item if successful, NULL - if the item was not found.

Example:

```
//--- example for CList::Search(CObject*)  
#include <Arrays\List.mqh>  
//---  
void OnStart()  
{  
    CList *list=new CList;  
    //---  
    if(list==NULL)  
    {  
        printf("Object create error");  
        return;  
    }  
    //--- add lists elements  
    //--- . . .  
    //--- sort list  
    list.Sort(0);  
    //--- create sample  
    CObject *sample=new CObject;  
    if(sample==NULL)  
    {  
        printf("Sample create error");  
        delete list;  
        return;  
    }  
    //--- set sample attributes  
    //--- . . .  
    //--- search element  
    if(list.Search(sample)!=NULL) printf("Element found");  
    else                          printf("Element not found");  
    //--- delete list  
    delete list;  
}
```

Save

Saves data in the file list.

```
virtual bool Save(  
    int file_handle    // File handle  
)
```

Parameters

file_handle

[in] handle of the previously opened using the function FileOpen (...) file.

Return Value

true - if successfully completed, false - if an error.

Example:

```
//--- example for CList::Save(int)  
#include <Arrays\List.mqh>  
//---  
void OnStart()  
{  
    int file_handle;  
    CList *list=new CList;  
    //---  
    if(list!=NULL)  
    {  
        printf("Object create error");  
        return;  
    }  
    //--- add lists elements  
    //--- . . .  
    //--- open file  
    file_handle=FileOpen("MyFile.bin",FILE_WRITE|FILE_BIN|FILE_ANSI);  
    if(file_handle>=0)  
    {  
        if(!list.Save(file_handle))  
        {  
            //--- file save error  
            printf("File save: Error %d!",GetLastError());  
            delete list;  
            FileClose(file_handle);  
            //---  
            return;  
        }  
        FileClose(file_handle);  
    }  
    //--- delete list  
    delete list;
```

```
}
```

Load

Loads list data from a file.

```
virtual bool Load(  
    int file_handle    // File handle  
)
```

Parameters

file_handle

[in] Handle of the previously open, with the function FileOpen (...) , binary

Return Value

true - if successfully completed, false - if an error.

Note

When reading from the file list items to create each element of the method is called CList::CreateElement () .

Example:

```
//--- example for CLoad::Load(int)  
#include <Arrays\List.mqh>  
//---  
void OnStart()  
{  
    int file_handle;  
    CList *list=new CList;  
    //---  
    if(list!=NULL)  
    {  
        printf("Object create error");  
        return;  
    }  
    //--- open file  
    file_handle=FileOpen("MyFile.bin",FILE_READ|FILE_BIN|FILE_ANSI);  
    if(file_handle>=0)  
    {  
        if(!list.Load(file_handle))  
        {  
            //--- file load error  
            printf("File load: Error %d!",GetLastError());  
            delete list;  
            FileClose(file_handle);  
            //---  
            return;  
        }  
        FileClose(file_handle);  
    }
```

```
    }  
    //--- use list elements  
    //--- . . .  
    //--- delete list  
    delete list;  
}
```

Type

Gets the type identifier list.

```
virtual int Type()
```

Return Value

Type identifier list (for CList - 7779) .

Example:

```
//--- example for CList::Type()
#include <Arrays\List.mqh>
//---
void OnStart()
{
    CList *list=new CList;
    //---
    if(list==NULL)
    {
        printf("Object create error");
        return;
    }
    //--- get list type
    int type=list.Type();
    //--- delete list
    delete list;
}
```


CTreeNode

Class CTreeNode is a class of node of the binary tree CTree.

Description

Class CTreeNode provides the ability to work with nodes of the binary tree [CTree](#). Options of navigation through the tree is implemented in the class. Besides that methods of work with a file are implemented.

Declaration

```
class CTreeNode : public CObject
```

Title

```
#include <Arrays\TreeNode.mqh>
```

Class Methods

Attributes	
Owner	Gets/sets the pointer of the owner node
Left	Gets/sets the pointer of the left node
Right	Gets/sets the pointer of the right node
Balance	Gets the node balance
BalanceL	Gets the balance of the left sub-branch of the node
BalanceR	Gets the balance of the right sub-branch of the node
Creation of a new element	
CreateSample	Creates a new node instance
Comparison	
RefreshBalance	Recalculates the node balance
Search	
GetNext	Gets the pointer of the next node
Input/Output	
SaveNode	Saves the node data to a file
LoadNode	Downloads the node data from a file
virtual Type	Gets the identifier of the node type

Derived classes:

- [CTree](#)

Trees of CTreeNode class descendants get practical application.

A descendant of class CTreeNode must have predefined methods: [CreateSample](#) that creates a new instance of the descendant class of CTreeNode, [Compare](#) that compares values of key fields of the descendant class of CTreeNode, [Type](#) (if it's necessary to identify a node) , [SaveNode](#) and [LoadNode](#) (if it's necessary to work with a file) .

Let's consider an example of a CTree descendant class.

```
//+-----+
//|                                     MyTreeNode.mq5 |
//|                                     Copyright 2010, MetaQuotes Software Corp. |
//|                                     http://www.metaquotes.net/ |
//+-----+
#property copyright "2010, MetaQuotes Software Corp."
#property link      "http://www.mql5.com"
//---
#include <Arrays\TreeNode.mqh>
//+-----+
//| Describe class derived from CTreeNode. |
//+-----+
//| Class CMyTreeNode. |
//| Purpose: Class of element of a binary tree. |
//|           Descendant of class CTreeNode. |
//+-----+
class CMyTreeNode : public CTreeNode
{
protected:
    //--- user's data
    long      m_long;           // key field of type long
    double    m_double;        // custom variable of type double
    string     m_string;        // custom variable of type string
    datetime   m_datetime;      // custom variable of type datetime

public:
    CMyTreeNode();

    //--- methods of accessing these user's data
    long      GetLong(void)      { return(m_long); }
    void      SetLong(long value) { m_long=value; }
    double    GetDouble(void)    { return(m_double); }
    void      SetDouble(double value) { m_double=value; }
    string     GetString(void)    { return(m_string); }
    void      SetString(string value) { m_string=value; }
    datetime   GetDateTime(void)  { return(m_datetime); }
    void      SetDateTime(datetime value) { m_datetime=value; }

    //--- methods of working with files
    virtual bool Save(int file_handle);
    virtual bool Load(int file_handle);
protected:
```

```

    virtual int      Compare(const CObject *node,int mode);
    ///--- methods of creating class instances
    virtual CTreeNode* CreateSample();
};
//+-----+
///| CMyTreeNode class constructor. |
///| INPUT:  none. |
///| OUTPUT: none. |
///| REMARK: none. |
//+-----+
void CMyTreeNode::CMyTreeNode()
{
    ///--- initialization of user's data
    m_long      =0;
    m_double     =0.0;
    m_string     ="";
    m_datetime   =0;
}
//+-----+
///| Comparison with another three node by the specified algorithm. |
///| INPUT:  node - array element to compare, |
///|          mode - identifier of comparison algorithm. |
///| OUTPUT: result of comparison (>0,0,<0). |
///| REMARK: none. |
//+-----+
int CMyTreeNode::Compare(const CObject *node,int mode)
{
    ///--- parameter mode is ignored, because tree construction algorithm is the only one
    int res=0;
    ///--- explicit type casting
    CMyTreeNode *n=node;
    res=(int) (m_long-n.m_long);
    ///---
    return(res);
}
//+-----+
///| Creation of a new class instance. |
///| INPUT:  none. |
///| OUTPUT: pointer to a new instance of class CMyTreeNode. |
///| REMARK: none. |
//+-----+
CTreeNode* CMyTreeNode::CreateSample()
{
    CMyTreeNode *result=new CMyTreeNode;
    ///---
    return(result);
}
//+-----+

```

```

//| Write tree node data to a file. |
//| INPUT:  file_handle -handle of a file pre-opened for writing. |
//| OUTPUT: true if OK, otherwise false. |
//| REMARK: none. |
//+-----+
bool CMyTreeNode::Save(int file_handle)
{
    uint i=0,len;
//--- checks
    if(file_handle<0) return(false);
//--- writing user data
//--- writing custom variable of type long
    if(FileWriteLong(file_handle,m_long)!=sizeof(long)) return(false);
//--- writing custom variable of type double
    if(FileWriteDouble(file_handle,m_double)!=sizeof(double)) return(false);
//--- writing custom variable of type string
    len=StringLen(m_string);
//--- write string length
    if(FileWriteInteger(file_handle,len,INT_VALUE)!=INT_VALUE) return(false);
//--- write the string
    if(len!=0 && FileWriteString(file_handle,m_string,len)!=len) return(false);
//--- writing custom variable of type datetime
    if(FileWriteLong(file_handle,m_datetime)!=sizeof(long)) return(false);
//---
    return(true);
}
//+-----+
//| Read tree node data from a file. |
//| INPUT:  file_handle -handle of a file pre-opened for reading. |
//| OUTPUT: true if OK, otherwise false. |
//| REMARK: none. |
//+-----+
bool CMyTreeNode::Load(int file_handle)
{
    uint i=0,len;
//--- checks
    if(file_handle<0) return(false);
//--- reading
    if(FileIsEnding(file_handle)) return(false);
//--- reading custom variable of type char
//--- reading custom variable of type long
    m_long=FileReadLong(file_handle);
//--- reading custom variable of type double
    m_double=FileReadDouble(file_handle);
//--- reading custom variable of type string
//--- read the string length
    len=FileReadInteger(file_handle,INT_VALUE);
//--- read the string

```

```
    if(len!=0) m_string=FileReadString(file_handle,len);  
    else      m_string="";  
    //--- reading custom variable of type datetime  
    m_datetime=FileReadLong(file_handle);  
    //---  
    return(true);  
}
```

Owner

Gets the pointer of the owner node.

```
CTreeNode* Owner ()
```

Return Value

Pointer of the node-owner.

Owner

Sets the pointer of the owner node.

```
void Owner (  
    CTreeNode* node    // node  
)
```

Parameters

node

[in] New value of the pointer of the owner node.

Return Value

None.

Left

Gets the pointer of the left node.

```
CTreeNode* Left()
```

Return Value

Pointer of the left node.

Left

Sets the pointer of the left node.

```
void Left(  
    CTreeNode* node    // node  
)
```

Parameters

node

[in] New value of the pointer of the left node.

Return Value

None.

Right

Gets the pointer of the right node.

```
CTreeNode* Right()
```

Return Value

The pointer of the right node.

Right

Sets the pointer of the right node.

```
void Right(  
    CTreeNode* node    // node  
)
```

Parameters

node

[in] New value of the pointer of the right node.

Return Value

None.

Balance

Gets the node balance.

```
int Balance() const
```

Return Value

Node balance.

BalanceL

Gets the balance of the left sub-branch of the node.

```
int BalanceL() const
```

Return Value

Balance of the left sub-branch of the node.

BalanceR

Gets the balance of the right sub-branch of the node.

```
int BalanceR() const
```

Return Value

Balance of the right sub-branch of the node.

CreateSample

Creates a new node sample.

```
virtual CTreeNode* CreateSample()
```

Return Value

Pointer to the new node sample or NULL.

RefreshBalance

Recalculates the node balance.

```
int RefreshBalance()
```

Return Value

Node balance.

GetNext

Gets the pointer of the next node.

```
CTreeNode* GetNext(  
    CTreeNode* node    // node  
)
```

Parameters

node

[in] Node of the search start.

Return Value

Pointer of the next node.

SaveNode

Writes node data to a file.

```
bool SaveNode(  
    int file_handle    // handle  
)
```

Parameters

file_handle

[in] Handle of a binary file that was earlier opened for writing.

Return Value

true in case of success, otherwise false.

LoadNode

Reads node data from a file.

```
bool LoadNode(  
    int      file_handle,    // handle  
    CTreeNode* main          // node  
)
```

Parameters

file_handle

[in] Handle of a binary file that was earlier opened for reading.

main

[in] Node for data.

Return Value

true in case of success, otherwise false.

Type

Gets the identifier of the node type.

```
virtual int Type() const
```

Return Value

Identifier of the node type.

CTree

Class CTree is a class of the binary tree of samples of class CTreeNode and its descendants.

Description

Class CTree provides the possibility to work with a binary tree of [CTreeNode](#) class samples and its descendants. Options of adding/inserting/deleting of three elements and search in a tree are implemented in the class. Besides that, methods of work with a file are implemented.

Note that mechanism of dynamic memory management is not implemented in class CTree (as distinct from classes [CList](#) and [CArrayObj](#)) . All tree nodes are deleted with memory release.

Declaration

```
class CTree : public CTreeNode
```

Title

```
#include <Arrays\Tree.mqh>
```

Class Methods

Attributes	
Root	Gets a root node of the tree
Creation of a new element	
CreateElement	Creates a new node instance
Filling	
Insert	Adds a node to a tree
Deletion	
Detach	Detaches a specified node from a tree
Delete	Deletes a specified node from a tree
Clear	Deletes all nodes of a tree
Search	
Find	Searches for a node in a tree by sample
Input/output	
virtual Save	Saves all the tree data to a file
virtual Load	Downloads tree data from a file
virtual Type	Gets identifier of the tree type

Trees of CTreeNode class descendants - descendants of class CTree get practical application.

Descendant of class CTree must have a predefined method [CreateElement](#) that creates a new sample of descendant class [CTreeNode](#).

Let's consider an example of descendant class CTree.

```
//+-----+
//|                                     MyTree.mq5 |
//|                                     Copyright 2010, MetaQuotes Software Corp. |
//|                                     http://www.metaquotes.net/ |
//+-----+

#property copyright "2010, MetaQuotes Software Corp."
#property link      "http://www.mql5.com"
//---
#include <Arrays\Tree.mqh>
#include "MyTreeNode.mqh"
//---
input int extCountedNodes = 100;
//+-----+
//| Describe class CMyTree derived from CTree. |
//+-----+
//| Class CMyTree. |
//| Purpose: Construction and navigation of a binary search tree. |
//+-----+
class CMyTree : public CTree
{
public:
    //--- methods of search on the tree by custom data
    CMyTreeNode* FindByLong(long find_long);
    //--- method of creation of the tree element
    virtual CTreeNode *CreateElement();
};
//---
CMyTree MyTree;
//+-----+
//| Creation of a new tree node. |
//| INPUT: none. |
//| OUTPUT: pointer to the new tree node of OK, or NULL. |
//| REMARK: none. |
//+-----+
CTreeNode *CMyTree::CreateElement()
{
    CMyTreeNode *node=new CMyTreeNode;
//---
    return (node);
}
//+-----+
//| Search of element in a list by value m_long. |
//| INPUT: find_long - searched value. |
//| OUTPUT: pointer of a found list element, or NULL. |
```

```

//| REMARK: none. |
//+-----+
CMyTreeNode* CMyTree::FindByLong(long find_long)
{
    CMyTreeNode *res=NULL;
    CMyTreeNode *node;
    //--- create a tree node to pass the search parameter
    node=new CMyTreeNode;
    if(node==NULL) return(NULL);
    node.SetLong(find_long);
    //---
    res=Find(node);
    delete node;
    //---
    return(res);
}
//+-----+
//| script "testing of class CMyTree" |
//+-----+
//--- array for string initialization
string str_array[11]={ "p", "oo", "iii", "uuuu", "yyyyy", "ttttt", "rrrr", "eee", "ww", "q", "99"
//---
int OnStart() export
{
    int i;
    uint pos;
    int beg_time,end_time;
    CMyTreeNode *node; //--- temporary pointer to the sample of class CMyTreeNode
    //---
    printf("Start test %s.", __FILE__);
    //--- Fill out MyTree with samples of class MyTreeNode in the amount of extCountedNodes
    beg_time=GetTickCount();
    for(i=0;i<extCountedNodes;i++)
    {
        node=MyTree.CreateElement();
        if(node==NULL)
        {
            //--- emergency exit
            printf("%s (%4d): create error", __FILE__, __LINE__);
            return(__LINE__);
        }
        NodeSetData(node,i);
        node.SetLong(i);
        MyTree.Insert(node);
    }
    end_time=GetTickCount();
    printf("Filling time of MyTree is %d ms.",end_time-beg_time);
    //--- Create a temporary tree TmpMyTree.

```

```

    CMyTree TmpMyTree;
//--- Detach 50% of tree elements (all even)
//--- and add them to the temporary tree TmpMyTree.
    beg_time=GetTickCount();
    for(i=0;i<extCountedNodes;i+=2)
    {
        node=MyTree.FindByLong(i);
        if(node!=NULL)
            if(MyTree.Detach(node)) TmpMyTree.Insert(node);
    }
    end_time=GetTickCount();
    printf("Deletion time of %d elements from MyTree is %d ms.",extCountedNodes/2,end_
//--- Return the detached
    node=TmpMyTree.Root();
    while(node!=NULL)
    {
        if(TmpMyTree.Detach(node)) MyTree.Insert(node);
        node=TmpMyTree.Root();
    }
//--- Check work of method Save(int file_handle);
    int file_handle;
    file_handle=FileOpen("MyTree.bin",FILE_WRITE|FILE_BIN|FILE_ANSI);
    if(file_handle>=0)
    {
        if(!MyTree.Save(file_handle))
        {
            //--- error writing to a file
            //--- emergency exit
            printf("%s: Error %d in %d!",__FILE__,GetLastError(),__LINE__);
            //--- close file before leaving!!!
            FileClose(file_handle);
            return(__LINE__);
        }
        FileClose(file_handle);
    }
//--- Check work of method Load(int file_handle);
    file_handle=FileOpen("MyTree.bin",FILE_READ|FILE_BIN|FILE_ANSI);
    if(file_handle>=0)
    {
        if(!TmpMyTree.Load(file_handle))
        {
            //--- error reading from file
            //--- emergency exit
            printf("%s: Error %d in %d!",__FILE__,GetLastError(),__LINE__);
            //--- close file before leaving!!!
            FileClose(file_handle);
            return(__LINE__);
        }
    }

```

```

        FileClose(file_handle);
    }
//---
    MyTree.Clear();
    TmpMyTree.Clear();
//---
    printf("End test %. OK!", __FILE__);
//---
    return(0);
}
//+-----+
//| Function of output of node contents to journal |
//+-----+
void NodeToLog(CMyTreeNode *node)
{
    printf("    %I64d,%f,'%s','%s',
           node.GetLong(),node.GetDouble(),
           node.GetString(),TimeToString(node.GetDateTime()));
}
//+-----+
//| Function of "filling" of node with random values |
//+-----+
void NodeSetData(CMyTreeNode *node,int mode)
{
    if(mode%2==0)
    {
        node.SetLong(mode*MathRand());
        node.SetDouble(MathPow(2.02,mode)*MathRand());
    }
    else
    {
        node.SetLong(mode*(long)(-1)*MathRand());
        node.SetDouble(-MathPow(2.02,mode)*MathRand());
    }
    node.SetString(str_array[mode%10]);
    node.SetDateTime(10000*mode);
}

```

Root

Gets the root node of the tree.

```
CTreeNode* Root() const
```

Return Value

Pointer of the root node of the tree.

CreateElement

Creates a new instance of the node.

```
virtual CTreeNode* CreateElement()
```

Return Value

Pointer of the new instance of the node or NULL.

Insert

Adds a node to a tree.

```
CTreeNode* Insert(  
    CTreeNode* new_node    // node  
)
```

Parameters

new_node

[in] pointer of a node to insert to a tree.

Return Value

Pointer of the owner node or NULL.

Detach

Detaches a specified node from a tree.

```
bool Detach(  
    CTreeNode* node    // node  
)
```

Parameters

node

[in] Node pointer to detach.

Return Value

true in case of success, otherwise false.

Note

After detachment the node pointer is not released. The tree is balanced.

Delete

Deletes a specified node from a tree.

```
bool Delete(  
    CTreeNode* node    // node  
)
```

Parameters

node

[in] Node pointer to delete.

Return Value

true in case of success, otherwise false.

Note

After deletion a node pointer is released. The tree is balanced.

Clear

Deletes all nodes of a tree.

```
void Clear()
```

Return Value

None.

Note

After deletion node pointers are released.

Find

Searches for a node in a tree by sample.

```
CTreeNode* Find(  
    CTreeNode* node    // node  
)
```

Parameters

node

[in] Node that contains data-search sample.

Return Value

Pointer of the found node or NULL.

Save

Writes tree data to a file.

```
virtual bool Save(  
    int file_handle    // handle  
)
```

Parameters

file_handle

[in] Handle of a binary file that was earlier opened for writing.

Return Value

true in case of success, otherwise false.

Load

Reads tree data to a file.

```
virtual bool Load(  
    int file_handle    // handle  
)
```

Parameters

file_handle

[in] Handle of a binary file that was earlier opened for reading.

Return Value

true in case of success, otherwise false.

Type

Gets identifier of the tree type.

```
virtual int Type() const
```

Return Value

Identifier of the tree type.

Graphic Objects

This section contains the technical details of working with classes of graphical objects and a description of the relevant components of the MQL5 Standard Library .

The use of classes of graphical objects, will save time when creating custom programs (scripts, expert) .

MQL5 Standard Library (in terms of graphical objects) is placed in the working directory of the terminal in the Include\ChartObjects folder.

Class/Group	Description
Base class for graphical object CChartObject	Base class of a graphic object
Lines	Group classes "Lines"
Channels	Group classes "Channels"
Gann Tools	Group classes "Gann"
Fibonacci Tools	Group classes "Fibonacci"
Elliott Tools	Group classes "Elliott"
Shapes	Group classes "Shapes"
Arrows	Group classes "Arrows"
Controls	Group classes "Controls"

CChartObject

CChartObject is a base class of graphic objects of chart type of the Standard MQL5 library.

Description

Class CChartObject provides the simplified access for all of its descendants to MQL5 API functions.

Declaration

```
class CChartObject : public CObject
```

Title

```
#include <ChartObjects\ChartObject.mqh>
```

Class Methods

Attributes	
ChartId	Gets the ID chart, who owns a graphic
Window	Gets the number of windows in which the chart is a graphic
Name	Gets/sets the name of a graphic object
NumPoints	Gets the number of anchor points
Assign	
Attach	Binds a graphic chart
SetPoint	Sets the anchor point
Delete	
Delete	Deletes a graphic chart
Detach	Untie a graphic chart
Shift	
ShiftObject	The relative movement of the object
ShiftPoint	The relative movement of the object point
Object properties	
Time	Gets/sets the time coordinates of the object point
Price	Gets/sets the price coordinate of a point object
Color	Gets/sets the color of the object
Style	Gets/sets the line style object
Width	Gets/sets the width of the line object
BackGround	Gets/sets the flag drawing object background

Selected	Gets/sets the flag sit on an object
Selectable	Gets/sets the flag selectable object
Description	Gets/sets the text of the object
Tooltip	Gets/sets the tooltip of the object
Timeframes	Gets/sets the mask of flags visibility of the object
CreateTime	Gets the time object creation
Levels properties of the object	
LevelsCount	Gets/sets the number of levels of object
LevelColor	Gets/sets the color of the line level
LevelStyle	Gets/sets the line style level
LevelWidth	Gets/sets the width of the line level
LevelValue	Gets/sets the level
LevelDescription	Gets/sets the text level
Access to MQL5 API functions	
GetInteger	Gets the value of the object properties
SetInteger	Sets the object properties
GetDouble	Gets the value of the object properties
SetDouble	Sets the object properties
GetString	Gets the value of the object properties
SetString	Sets the object properties
Input/Output	
virtual Save	Virtual method entry in the file
virtual Load	Virtual method of reading from a file
virtual Type	Virtual method of identification

Derived classes:

- [CChartObjectArrow](#)
- [CChartObjectBitmap](#)
- [CChartObjectBmpLabel](#)
- [CChartObjectCycles](#)
- [CChartObjectElliottWave3](#)
- [CChartObjectEllipse](#)
- [CChartObjectFiboArc](#)

- [CChartObjectFiboFan](#)
- [CChartObjectFiboTimes](#)
- [CChartObjectHLine](#)
- [CChartObjectRectangle](#)
- [CChartObjectSubChart](#)
- [CChartObjectText](#)
- [CChartObjectTrend](#)
- [CChartObjectTriangle](#)
- [CChartObjectVLine](#)

ChartId

Gets the ID chart, who owns a graphic object.

```
long ChartId() const
```

Return Value

Id chart on which the graphic object. If object not found, it returns -1.

Example:

```
//--- example for CChartObject::ChartId
#include <ChartObjects\ChartObject.mqh>
//---
void OnStart()
{
    CChartObject object;
    //--- get chart identifier of chart object
    long chatr_id=object.ChartId();
}
```

Window

Gets the number of windows in which the chart is a graphic object.

```
int Window() const
```

Return Value

Number of windows in which the chart is a graphic object (0 - main window) . If object not found, it returns -1.

Example:

```
//--- example for CChartObject::Window
#include <ChartObjects\ChartObject.mqh>
//---
void OnStart()
{
    CChartObject object;
    //--- get window of chart object
    int window=object.Window();
}
```

Name (Get Method)

Gets the name of the graphic object.

```
string Name() const
```

Return Value

Name of the graphic object tied to an instance of the class. If object not found, returns NULL.

Name (Set Method)

Sets the name of the graphic object.

```
bool Name(  
    string name    // new name  
)
```

Parameters

name

[in] The new name of the graphic object.

Return Value

true if successful, false - if you can not change the name.

Example:

```
//--- example for CChartObject::Name  
#include <ChartObjects\ChartObject.mqh>  
//---  
void OnStart()  
{  
    CChartObject object;  
    //--- get name of chart object  
    string object_name=object.Name();  
    if(object_name!="MyChartObject")  
    {  
        //--- set name of chart object  
        object.Name("MyChartObject");  
    }  
}
```

NumPoints

Gets the number of anchor points of a graphic object.

```
int NumPoints() const
```

Return Value

Number of points linking a graphic object that is bound to an instance of the class. If not bound object, it returns 0.

Example:

```
//--- example for CChartObject::NumPoints
#include <ChartObjects\ChartObject.mqh>
//---
void OnStart()
{
    CChartObject object;
    //--- get points count of chart object
    int points=object.NumPoints();
}
```


Attach

Binds a graphical object to an instance of the class.

```
bool Attach(  
    long    chart_id,      // Chart ID  
    string  name,          // Name of the object  
    int     window,        // Chart window  
    int     points         // Number of points  
)
```

Parameters

chart_id

[out] Chart identifier.

name

[in] Name of the graphic object.

window

[in] Chart window number (0 - main window) .

points

[in] Number of points anchor graphic object.

Return Value

true - if successful, false - if you can not bind object.

Example:

```
///--- example for CChartObject::Attach  
#include <ChartObjects\ChartObject.mqh>  
///---  
void OnStart()  
{  
    CChartObject object;  
    ///--- attach chart object  
    if(!object.Attach(CharID(), "MyObject", 0, 2))  
    {  
        printf("Object attach error");  
        return;  
    }  
}
```

SetPoint

Sets the new coordinates of this anchor point graphic object.

```
bool SetPoint(  
    int      point,           // Point number  
    datetime new_time,       // Time coordinate  
    double   new_price       // Price coordinate  
)
```

Parameters

point

[in] Number anchor point graphic object.

new_time

[in] The new value of the coordinates of this point of time bindings.

new_price

[in] New value coordinates of the price specified anchor point.

Return Value

true - if successful, false - if you can not change the coordinates of the point.

Example:

```
///--- example for CChartObject::SetPoint  
#include <ChartObjects\ChartObject.mqh>  
///---  
void OnStart()  
{  
    CChartObject object;  
    double       price;  
    ///---  
    if(object.NumPoints()>0)  
    {  
        ///--- set point of chart object  
        object.SetPoint(0,CurrTime(),price);  
    }  
}
```

Delete

Removes a graphical object with the attached chart.

```
bool Delete()
```

Return Value

true - if successful, false - if you can not remove the object.

Example:

```
//--- example for CChartObject::Delete
#include <ChartObjects\ChartObject.mqh>
//---
void OnStart()
{
    CChartObject object;
    //--- detach chart object
    if(!object.Delete())
    {
        printf("Object delete error");
        return;
    }
}
```

Detach

Untie graphic object.

```
bool Detach()
```

Return Value

true - if successful, false - if you can not decouple the object.

Example:

```
//--- example for CChartObject::Detach
#include <ChartObjects\ChartObject.mqh>
//---
void OnStart()
{
    CChartObject object;
    //--- detach chart object
    if(!object.Detach())
    {
        printf("Object detach error");
        return;
    }
}
```

ShiftObject

Moves a graphic object.

```
bool ShiftObject(  
    datetime d_time,      // Increment of time coordinate  
    double   d_price      // Increment of price coordinate  
)
```

Parameters

d_time

[in] Increment the coordinates of all points of time bindings

d_price

[in] Increment the coordinates of the prices of all waypoints.

Return Value

true - if successful, false - if you can not move the object.

Example:

```
///--- example for CChartObject::ShiftObject  
#include <ChartObjects\ChartObject.mqh>  
///---  
void OnStart()  
{  
    CChartObject object;  
    datetime     d_time;  
    double       d_price;  
    ///--- shift chart object  
    object.ShiftObject(d_time,d_price);  
}
```

ShiftPoint

Moves a specified point anchor graphic.

```
bool ShiftPoint(  
    int      point,          // Point number  
    datetime d_time,        // Increment of time coordinate  
    double   d_price         // Increment of price coordinate  
)
```

Parameters

point

[in] Number anchor point graphic object.

d_time

[in] Increment the coordinates of time specified anchor point.

d_price

[in] Increment the coordinates of the price specified anchor point.

Return Value

true - if successful, false - if you can not move the point.

Example:

```
//--- example for CChartObject::ShiftPoint  
#include <ChartObjects\ChartObject.mqh>  
//---  
void OnStart()  
{  
    CChartObject object;  
    datetime     d_time;  
    double       d_price;  
    //---  
    if(object.NumPoints()>0)  
    {  
        //--- shift point of chart object  
        object.ShiftPoint(0,d_time,d_price);  
    }  
}
```

Time (Get Method)

The coordinates of time specified anchor point graphic object.

```
datetime Time(
    int point      // Point number
) const
```

Parameters

point

[in] Number anchor point graphic object.

Return Value

Coordinates of time specified anchor point graphic object that is bound to an instance of the class. If not bound object or the object is not this point, it returns 0.

Time (Set Method)

Sets the coordinate of time specified anchor point graphic object.

```
bool Time(
    int point,      // Point number
    datetime new_time // Time
)
```

Parameters

point

[in] Number anchor point graphic object.

new_time

[in] The new value of the coordinates of this point of time anchor graphic object.

Return Value

true - if successful, false - if you can not change the coordinate of time.

Example:

```
///--- example for CChartObject::Time
#include <ChartObjects\ChartObject.mqh>
///---
void OnStart()
{
    CChartObject object;
    ///---
    for(int i=0;i<object.NumPoints();i++)
    {
        ///--- get time of point chart object
        datetime point_time=object.Time(i);
        if(point_time==0)
```

```
{  
    //--- set time of point chart object  
    object.Time(i,TimeCurrent());  
}  
}  
}
```


Price (Get Method)

Gets the coordinate of the price specified anchor point graphic object.

```
double Price(  
    int point      // Point number  
) const
```

Parameters

point

[in] Number anchor point graphic object.

Return Value

Coordinate prices specified anchor point graphic object that is bound to an instance of the class. If not bound object or the object is not this point, it returns EMPTY_VALUE.

Price (Set Method)

Sets the coordinate of the price specified anchor point graphic object.

```
bool Price(  
    int point,      // Point number  
    double new_price // Price  
)
```

Parameters

point

[in] Number anchor point graphic object.

new_price

[in] News value coordinates of the price specified anchor point graphic object.

Return Value

true - if successful, false - if you can not change the coordinate prices.

Example:

```
///--- example for CChartObject::Price  
#include <ChartObjects\ChartObject.mqh>  
///---  
void OnStart()  
{  
    CChartObject object;  
    double price;  
    ///---  
    for(int i=0;i<object.NumPoints();i++)  
    {  
        ///--- get price of point chart object  
        double point_price=object.Price(i);
```

```
if(point_price!=price)
{
    //--- set price of point chart object
    object.Price(i,price);
}
}
```

Color (Get Method)

Gets the line color of the graphic object.

```
color Color() const
```

Return Value

Color line of graphic object, assigned to the class instance. If there is no object assigned, it returns CLR_NONE.

Color (Set Method)

Sets the color of the line for the graphic object.

```
bool Color(  
    color new_color    // New color  
)
```

Parameters

new_color

[in] New value line color graphic object.

Return Value

true - if successful, false - if you can not change the color.

Example:

```
//--- example for CChartObject::Color  
#include <ChartObjects\ChartObject.mqh>  
//---  
void OnStart()  
{  
    CChartObject object;  
    //--- get color of chart object  
    color object_color=object.Color();  
    if(object_color!=clrRed)  
    {  
        //--- set color of chart object  
        object.Color(clrRed);  
    }  
}
```

Style (Get Method)

Gets the line style graphic.

```
ENUM_LINE_STYLE Style() const
```

Return Value

Style line of the graphic object, assigned to the class instance. If there is no object assigned, it returns WRONG_VALUE.

Style (Set Method)

Sets the line style graphic.

```
bool Style(  
    ENUM_LINE_STYLE new_style    // Style  
)
```

Parameters

new_style

[in] New value-style line drawing object.

Return Value

true - if successful, false - if you can not change the style.

Example:

```
//--- example for CChartObject::Style  
#include <ChartObjects\ChartObject.mqh>  
//---  
void OnStart()  
{  
    CChartObject object;  
    //--- get style of chart object  
    ENUM_LINE_STYLE style=object.Style();  
    if(style!=STYLE_SOLID)  
    {  
        //--- set style of chart object  
        object.Style(STYLE_SOLID);  
    }  
}
```

Width (Get Method)

Gets the thickness of the line graphic object.

```
int Width() const
```

Return Value

The thickness of the line graphic object that is bound to an instance of the class. If not bound object, it returns -1.

Width (Set Method)

Sets the thickness of the line graphic object.

```
bool Width(  
    int new_width    // Thickness  
)
```

Parameters

new_width

[in] The new value of the thickness of the line graphic object.

Return Value

true - if successful, false - if you can not change the thickness.

Example:

```
//--- example for CChartObject::Width  
#include <ChartObjects\ChartObject.mqh>  
//---  
void OnStart()  
{  
    CChartObject object;  
    //--- get width of chart object  
    int width=object.Width();  
    if(width!=1)  
    {  
        //--- set width of chart object  
        object.Width(1);  
    }  
}
```

Background (Get Method)

Gets the flag drawing a graphic object in the background.

```
bool Background() const
```

Return Value

Flag of drawing in the background, a graphic object that is bound to an instance of the class. If not bound object returns false.

Background (Set Method)

Sets the flag drawing a graphic object in the background.

```
bool Background(  
    bool background    // Value of the flag  
)
```

Parameters

background

[in] New value of the flag drawing a graphic object in the background.

Return Value

true - if successful, false - if you can not change the flag.

Example:

```
//--- example for CChartObject::Background  
#include <ChartObjects\ChartObject.mqh>  
//---  
void OnStart()  
{  
    CChartObject object;  
    //--- get background flag of chart object  
    bool background_flag=object.Background();  
    if(!background_flag)  
    {  
        //--- set background flag of chart object  
        object.Background(true);  
    }  
}
```

Selected (Get Method)

Gets the flag "reprimand" graphic object.

```
bool Selected() const
```

Return Value

Flag "slate", a graphic object that is bound to an instance of the class. If not bound object returns false.

Selected (Set Method)

Sets the flag "reprimand" graphic object.

```
bool Selected(  
    bool selected    // Value of the flag  
)
```

Parameters

selected

[in] New value of the flag "reprimand" graphic object.

Return Value

true - if successful, false - if you can not change the flag.

Example:

```
//--- example for CChartObject::Selected  
#include <ChartObjects\ChartObject.mqh>  
//---  
void OnStart()  
{  
    CChartObject object;  
    //--- get selected flag of chart object  
    bool selected_flag=object.Selected();  
    if(selected_flag)  
    {  
        //--- set selected flag of chart object  
        object.Selected(false);  
    }  
}
```

Selectable (Get Method)

Gets the flag "selectable" graphic object.

```
bool Selectable() const
```

Return Value

Flag "selectable", a graphic object that is bound to an instance of the class. If not bound object returns false.

Selectable (Set Method)

Sets the flag "selectable" graphic object.

```
bool Selectable(  
    bool selectable    // Value of the flag  
)
```

Parameters

selectable

[in] New value of the flag "selectable" graphic object.

Return Value

true - if successful, false - if you can not change the flag.

Example:

```
//--- example for CChartObject::Selectable  
#include <ChartObjects\ChartObject.mqh>  
//---  
void OnStart()  
{  
    CChartObject object;  
    //--- get selectable flag of chart object  
    bool selectable_flag=object.Selectable();  
    if(selectable_flag)  
    {  
        //--- set selectable flag of chart object  
        object.Selectable(false);  
    }  
}
```


Description (Get Method)

Gets a description (text) graphic object.

```
string Description() const
```

Return Value

Description (text) graphic object that is bound to an instance of the class. If no bound object, it returns NULL.

Description (Set Method)

Sets the description (text) graphic object.

```
bool Description(  
    string text    // Text  
)
```

Parameters

text

[in] New value descriptions (text) graphic object.

Return Value

true - if successful, false - if you can not change the description (text) .

Example:

```
//--- example for CChartObject::Description  
#include <ChartObjects\ChartObject.mqh>  
//---  
void OnStart()  
{  
    CChartObject object;  
    //--- get description of chart object  
    string description=object.Description();  
    if(description=="")  
    {  
        //--- set description of chart object  
        object.Description("MyObject");  
    }  
}
```

Tooltip (Get Method)

Gets the text of the tooltip of graphic object.

```
string Tooltip() const
```

Returned value

The text of a tooltip. If the object is not assigned, it returns NULL.

Tooltip (Set Method)

Sets the text of a tooltip.

```
bool Tooltip(  
    string new_tooltip    // new text of a tooltip  
)
```

Parameters

new_tooltip

[in] New value of a tooltip

Returned value

true - if successful, false if tooltip change has failed.

Note:

If the property is not set, then the tooltip generated automatically by the terminal is shown. A tooltip can be disabled by setting the "\n" (line feed) value.

Timeframes (Get Method)

Gets the visibility flag graphic object.

```
int Timeframes() const
```

Return Value

Flags visibility graphic object that is bound to an instance of the class. If not bound object, it returns 0.

Timeframes (Set Method)

Sets the visibility flag graphic.

```
bool Timeframes (
    int new_timeframes    // Visibility flags
)
```

Parameters

new_timeframes

[in] New flags visibility graphic object.

Return Value

true - if successful, false - if you can not change the flags of visibility.

Example:

```
///--- example for CChartObject::Timeframes
#include <ChartObjects\ChartObject.mqh>
///---
void OnStart ()
{
    CChartObject object;
    ///--- get timeframes of chart object
    int timeframes=object.Timeframes();
    if (!(timeframes&OBJ_PERIOD_H1))
    {
        ///--- set timeframes of chart object
        object.Timeframes (timeframes|OBJ_PERIOD_H1);
    }
}
```

CreateTime

Gets the time to create graphical object.

```
datetime CreateTime() const
```

Return Value

Time to create graphical object that is bound to an instance of the class. If not bound object, it returns 0.

Example:

```
//--- example for CChartObject::CreateTime
#include <ChartObjects\ChartObject.mqh>
//---
void OnStart()
{
    CChartObject object;
    //--- get create time of chart object
    datetime create_time=object.CreateTime();
}
```

LevelsCount (Get Method)

Gets the number of levels of graphical object.

```
int LevelsCount() const
```

Return Value

Number of levels of graphical object that is bound to an instance of the class. If not bound object, it returns 0.

LevelsCount (Set Method)

Sets the number of levels of graphical object.

```
bool LevelsCount(  
    int levels    // Number of levels  
)
```

Parameters

levels

[in] the number of new levels of graphic object.

Return Value

true - if successful, false - if you can not change the number of levels.

Example:

```
//--- example for CChartObject::LevelsCount  
#include <ChartObjects\ChartObject.mqh>  
//---  
void OnStart()  
{  
    CChartObject object;  
    //--- get levels count of chart object  
    int levels_count=object.LevelsCount();  
    //--- set levels count of chart object  
    object.LevelsCount(levels_count+1);  
}
```

LevelColor (Get Method)

Gets the line color specified level of graphic object.

```
color LevelColor(
    int level      // Level number
) const
```

Parameters

level

[in] Number of levels of graphical object.

Return Value

Color Line level specified graphic object that is bound to an instance of the class. If not bound object or the object is no specified level, it returns CLR_NONE.

LevelColor (Set Method)

Sets the line color specified level of graphic object.

```
bool LevelColor(
    int level,      // Level number
    color new_color // New color
)
```

Parameters

level

[in] Number of levels of graphical object.

new_color

[in] New value line color of the level of graphic object.

Return Value

true - if successful, false - if you can not change the color.

Example:

```
//--- example for CChartObject::LevelColor
#include <ChartObjects\ChartObject.mqh>
//---
void OnStart()
{
    CChartObject object;
    //---
    for(int i=0;i<object.LevelsCount();i++)
    {
        //--- get level color of chart object
        color level_color=object.LevelColor(i);
        if(level_color!=clrRed)
```

```
{  
    //--- set level color of chart object  
    object.LevelColor(i,clrRed);  
}  
}  
}
```

LevelStyle (Get Method)

Gets the line style specified level of graphical object.

```
ENUM_LINE_STYLE LevelStyle(
    int level      // Level number
) const
```

Parameters

level

[in] Number of levels of graphical object.

Return Value

Line style specified level graphical object that is bound to an instance of the class. If not bound object or the object is no specified level of returns WRONG__VALUE.

LevelStyle (Set Method)

Sets the line style specified level of graphical object.

```
int LevelStyle(
    int level,      // Level number
    ENUM_LINE_STYLE style // Line Style
)
```

Parameters

level

[in] Number of levels of graphical object.

style

[in] New value-style line level specified graphical object.

Return Value

true - if successful, false - if you can not change the style.

Example:

```
//--- example for CChartObject::LevelStyle
#include <ChartObjects\ChartObject.mqh>
//---
void OnStart()
{
    CChartObject object;
    //---
    for(int i=0;i<object.LevelsCount();i++)
    {
        //--- get level style of chart object
        ENUM_LINE_STYLE level_style=object.LevelStyle(i);
        if(level_style!=STYLE_SOLID)
```



```
{
    //-- set level style of chart object
    object.LevelStyle(i,STYLE_SOLID);
}
}
```

LevelWidth (Get Method)

Gets the line thickness specified level of graphic object.

```
int LevelWidth(
    int level      // Level number
) const
```

Parameters

level

[in] Number of levels of graphical object.

Return Value

The thickness of the lines of the level of graphical object that is bound to an instance of the class. If not bound object or the object is no specified level, it returns -1.

LevelWidth (Set Method)

Finds the last element equal to the model in sorted array.

```
bool LevelWidth(
    int level,      // Level number
    int new_width   // New width
)
```

Parameters

level

[in] Number of levels of graphical object.

new_width

[in] New value line thickness specified level graphical object.

Return Value

position of the found element, if successful, -1 - if the item was not found.

Example:

```
///--- example for CChartObject::LevelWidth
#include <ChartObjects\ChartObject.mqh>
///---
void OnStart()
{
    CChartObject object;
    ///---
    for(int i=0;i<object.LevelsCount();i++)
    {
        ///--- get level width of chart object
        int level_width=object.LevelWidth(i);
        if(level_width!=1)
```

```
{  
    //--- set level width of chart object  
    object.LevelWidth(i,1);  
}  
}  
}
```

LevelValue (Get Method)

Gets the value of the level of graphic object.

```
double LevelValue(  
    int level      // Level number  
) const
```

Parameters

level

[in] Number of levels of graphical object.

Return Value

The value of this level of graphical object that is bound to an instance of the class. If not bound object or the object is no level specified, returns EMPTY_VALUE.

LevelValue (Set Method)

Sets the value of the specified level of graphic object.

```
bool LevelValue(  
    int level,      // Level number  
    double new_value // New value  
)
```

Parameters

level

[in] Number of levels of graphical object.

new_value

[in] New value of the level of graphic object.

Return Value

true - if successful, false - if you can not change the level.

Example:

```
///--- example for CChartObject::LevelValue  
#include <ChartObjects\ChartObject.mqh>  
///---  
void OnStart()  
{  
    CChartObject object;  
    ///---  
    for(int i=0;i<object.LevelsCount();i++)  
    {  
        ///--- get level value of chart object  
        double level_value=object.LevelValue(i);  
        if(level_value!=0.1*i)
```

```
{  
    //--- set level value of chart object  
    object.LevelValue(i,0.1*i);  
}  
}  
}
```

LevelDescription (Get Method)

Gets a description (text) of the level of graphical object.

```
string LevelDescription(  
    int level          // Level number  
) const
```

Parameters

level

[in] Number of levels of graphical object.

Return Value

Description (text) of the level of graphical object that is bound to an instance of the class. If not bound object or the object is no specified level, returns NULL.

LevelDescription (Set Method)

Sets the description (text) of the level of graphical object.

```
bool LevelDescription(  
    int level ,        // Level number  
    string text        // Text  
)
```

Parameters

level

[in] Number of level of graphical object.

text

[in] New value of description (text) of the level of graphic object.

Return Value

true - if successful, false - if you can not change the description (text) .

Example:

```
//--- example for CChartObject::LevelDescription  
#include <ChartObjects\ChartObject.mqh>  
//---  
void OnStart()  
{  
    CChartObject object;  
    //---  
    for(int i=0;i<object.LevelsCount();i++)  
    {  
        //--- get level description of chart object  
        string level_description=object.LevelDescription(i);  
        if(level_description=="")
```

```
{  
    ///--- set level description of chart object  
    object.LevelDescription(i,"Level_"+IntegerToString(i));  
}  
}  
}
```

GetInteger

Provides simplified access to the functions of API MQL5 [ObjectGetInteger\(\)](#) for integer-value properties (of type bool, char, uchar, short, ushort, int, uint, ling, ulong, datetime, color) bound to an instance of the class graphic. There are two versions of a function call:

Getting a property value without checking the correctness

```
long GetInteger (
    ENUM_OBJECT_PROPERTY_INTEGER prop_id,          // Identifier of integer-property
    int modifier=-1                                // Modifier
) const
```

Parameters

prop_id

[in] ID of double-graphic properties.

modifier=-1

[in] Modifier (index) double-features.

Return Value

If successfull, it returns the value of integer-type property, if error, it returns 0.

Getting a property value in verifying the correctness of such treatment

```
bool GetInteger (
    ENUM_OBJECT_PROPERTY_INTEGER prop_id,          // Identifier of integer-property
    int modifier,                                  // Modifier
    long& value                                     // Link to variable
) const
```

Parameters

prop_id

[in] ID integer-graphic properties of the object.

modifier

[in] Modifier (index) integer-property.

value

[out] Reference to a variable to accommodate the integer-value properties.

Return Value

true - if successful, false - if you can not get integer-property.

Example:

```
//--- example for CChartObject::GetInteger
#include <ChartObjects\ChartObject.mqh>
//---
void OnStart ()
```



```
{
    CChartObject object;
    //--- get color of chart object by easy method
    printf("Objects color is %s",ColorToString(object.GetInteger(OBJPROP_COLOR),true))
    //--- get color of chart object by classic method
    long color_value;
    if(!object.GetInteger(OBJPROP_COLOR,0,color_value))
    {
        printf("Get integer property error %d",GetLastError());
        return;
    }
    else
        printf("Objects color is %s",color_value);
    for(int i=0;i<object.LevelsCount();i++)
    {
        //--- get levels width by easy method
        printf("Level %d width is %d",i,object.GetInteger(OBJPROP_LEVELWIDTH,i));
        //--- get levels width by classic method
        long width_value;
        if(!object.GetInteger(OBJPROP_LEVELWIDTH,i,width_value))
        {
            printf("Get integer property error %d",GetLastError());
            return;
        }
        else
            printf("Level %d width is %d",i,width_value);
    }
}
```

SetInteger

Provides simplified access to the functions of API MQL5 [ObjectSetInteger\(\)](#) to change the integer-properties (with types bool, char, uchar, short, ushort, int, uint, ling, ulong, datetime, color) bound to an instance of the class graphic. There are two versions of a function call:

Setting a property value that does not require a modifier

```
bool SetInteger (
    ENUM_OBJECT_PROPERTY_INTEGER prop_id,    // Identifier of integer-property
    long value                                // Value
)
```

Parameters

prop_id

[in] ID integer-graphic properties of the object.

value

[in] new mutable integer-value properties.

Setting a property value indicating the modifier

```
bool SetInteger (
    ENUM_OBJECT_PROPERTY_INTEGER prop_id,    // Identifier of integer-property
    int modifier,                           // Modifier
    long value                               // Value
)
```

Parameters

prop_id

[in] ID integer-graphic properties of the object.

modifier

[in] Modifier (index) integer-property.

value

[in] new mutable integer-value properties.

Return Value

true - if successful, false - if you can not change the integer-property.

Example:

```
///--- example for CChartObject::SetInteger
#include <ChartObjects\ChartObject.mqh>
///---
void OnStart()
{
    CChartObject object;
    ///--- set new color of chart object
```

```
if(!object.SetInteger(OBJPROP_COLOR,clrRed))
{
    printf("Set integer property error %d",GetLastError());
    return;
}
for(int i=0;i<object.LevelsCount();i++)
{
    ///--- set levels width
    if(!object.SetInteger(OBJPROP_LEVELWIDTH,i,i))
    {
        printf("Set integer property error %d",GetLastError());
        return;
    }
}
}
```

GetDouble

Provides simplified access to the functions of API MQL5 [ObjectGetDouble\(\)](#) to get the values double-properties (having type float and double) of the graphic object, assigned to the class instance. There are two versions of a function call:

Getting a property value without checking the correctness

```
double  GetDouble (
    ENUM_OBJECT_PROPERTY_DOUBLE prop_id,      // Identifier of double-property
    int      modifier=-1                      // Modifier
) const
```

Parameters

prop_id
[in] ID of double-graphic properties.

modifier=-1
[in] Modifier (index) double-features.

Return Value

If successful, it returns the value of property of double type, if error, it returns EMPTY_VALUE.

Getting a property value in verifying the correctness of such treatment

```
bool  GetDouble (
    ENUM_OBJECT_PROPERTY_DOUBLE prop_id,      // Identifier of double-property
    int      modifier,                      // Modifier
    double&   value                        // Link to variable
) const
```

Parameters

prop_id
[in] ID of double-graphic properties.

modifier
[in] Modifier (index) double-features.

value
[out] Reference to a variable to accommodate the double-value properties.

Return Value

true - if successful, false - if you can not get a double-feature.

Example:

```
//--- example for CChartObject::GetDouble
#include <ChartObjects\ChartObject.mqh>
//---
void OnStart ()
```

```
{
    CChartObject object;
    //---
    for(int i=0;i<object.LevelsCount();i++)
    {
        //--- get levels value by easy method
        printf("Level %d value=%f",i,object.GetDouble(OBJPROP_LEVELVALUE,i));
        //--- get levels value by classic method
        double value;
        if(!object.SetDouble(OBJPROP_LEVELVALUE,i,value))
        {
            printf("Get double property error %d",GetLastError());
            return;
        }
        else
            printf("Level %d value=%f",i,value);
    }
}
```

SetDouble

Provides simplified access to the functions of API MQL5 [ObjectSetDouble\(\)](#) to change the double-properties (having type float and double) bound to an instance of the class graphic object. There are two versions of a function call:

Setting a property value that does not require a modifier

```
bool SetDouble (
    ENUM_OBJECT_PROPERTY_DOUBLE prop_id,    // Identifier of double-property
    double value                             // Value
)
```

Parameters

prop_id

[in] ID of double-graphic properties.

value

[in] New value mutable double-features.

Setting a property value indicating the modifier

```
bool SetDouble (
    ENUM_OBJECT_PROPERTY_DOUBLE prop_id,    // Identifier of double-property
    int modifier,                          // Modifier
    double value                             // Value
)
```

Parameters

prop_id

[in] ID of double-graphic properties.

modifier

[in] Modifier (index) of double-property.

value

[in] New value mutable double-property.

Return Value

true - if successful, false - if you can not change the double-feature.

Example:

```
//--- example for CChartObject::SetDouble
#include <ChartObjects\ChartObject.mqh>
//---
void OnStart()
{
    CChartObject object;
//---
```

```
for(int i=0;i<object.LevelsCount();i++)
{
    //--- set level value of chart object
    if(!object.SetDouble(OBJPROP_LEVELVALUE,i,0.1*i))
    {
        printf("Set double property error %d",GetLastError());
        return;
    }
}
}
```

GetString

Provides simplified access to the functions of API MQL5 [ObjectGetString\(\)](#) for string-value properties, is bound to an instance of the class `graphic object`. There are two versions of a function call:

Getting a property value without checking the correctness

```
string GetString(
    ENUM_OBJECT_PROPERTY_STRING prop_id,      // Identifier of string-property
    int modifier=-1                          // Modifier
) const
```

Parameters

prop_id

[in] Identifier string-properties of graphic object.

modifier=-1

[in] Modifier (index) string-properties.

Return Value

Value of string-property.

Getting a property value in verifying the correctness of such treatment

```
bool GetString(
    ENUM_OBJECT_PROPERTY_STRING prop_id,      // Identifier of string-property
    int modifier,                          // Modifier
    string& value                          // Link to variable
) const
```

Parameters

prop_id

[in] Identifier string-properties of graphic object.

modifier

[in] Modifier (index) string-properties.

value

[out] Reference to a variable to accommodate the string-value properties.

Return Value

true - if successful, false - if you can not get a string-property.

Example:

```
///--- example for CChartObject::GetString
#include <ChartObjects\ChartObject.mqh>
///---
void OnStart ()
{
```



```
CChartObject object;
string      value;
//--- get name of chart object by easy method
printf("Object name is '%s'",object.GetString(OBJPROP_NAME));
//--- get name of chart object by classic method
if(!object.GetString(OBJPROP_NAME,0,value))
{
    printf("Get string property error %d",GetLastError());
    return;
}
else
    printf("Object name is '%s'",value);
for(int i=0;i<object.LevelsCount();i++)
{
    //--- get levels description by easy method
    printf("Level %d description is '%s'",i,object.GetString(OBJPROP_LEVELTEXT,i));
    //--- get levels description by classic method
    if(!object.GetString(OBJPROP_LEVELTEXT,i,value))
    {
        printf("Get string property error %d",GetLastError());
        return;
    }
    else
        printf("Level %d description is '%s'",i,value);
}
}
```

SetString

Provides simplified access to the functions of API MQL5 [ObjectSetString\(\)](#) to change the properties of string-tied to the instance of the class graphic object. There are two versions of a function call:

Setting a property value that does not require a modifier

```
bool SetString(  
    ENUM_OBJECT_PROPERTY_STRING prop_id,    // Identifier of string-property  
    string value                             // Value  
)
```

Parameters

prop_id

[in] Identifier string-properties of graphic object.

value

[in] New value mutable string-properties.

Setting a property value indicating the modifier

```
bool SetString(  
    ENUM_OBJECT_PROPERTY_STRING prop_id,    // Identifier of string-property  
    int modifier,                          // Modifier  
    string value                             // Value  
)
```

Parameters

prop_id

[in] Identifier string-properties of graphic object.

modifier

[in] Modifier (index) string-properties.

value

[in] New value mutable string-properties.

Return Value

true - if successful, false - if you can not change the string-property.

Example:

```
//--- example for CChartObject::SetString  
#include <ChartObjects\ChartObject.mqh>  
//---  
void OnStart()  
{  
    CChartObject object;  
    //--- set new name of chart object  
    if(!object.SetString(OBJPROP_NAME, "MyObject"))
```

```
{
    printf("Set string property error %d", GetLastError());
    return;
}
for(int i=0; i<object.LevelsCount(); i++)
{
    ///--- set levels description
    if(!object.SetString(OBJPROP_LEVELTEXT, i, "Level_" + IntegerToString(i)))
    {
        printf("Set string property error %d", GetLastError());
        return;
    }
}
}
```

Save

Saves parameters of the object in the file.

```
virtual bool Save(  
    int file_handle    // File handle  
)
```

Parameters

file_handle

[in] handle of the previously opened using the function FileOpen (...) file.

Return Value

true - if successfully completed, false - if an error.

Example:

```
//--- example for CChartObject::Save  
#include <ChartObjects\ChartObject.mqh>  
//---  
void OnStart()  
{  
    int file_handle;  
    CChartObject object=new CChartObject;  
    //--- set object parameters  
    //--- . . .  
    //--- open file  
    file_handle=FileOpen("MyFile.bin",FILE_WRITE|FILE_BIN|FILE_ANSI);  
    if(file_handle>=0)  
    {  
        if(!object.Save(file_handle))  
        {  
            //--- file save error  
            printf("File save: Error %d!",GetLastError());  
            FileClose(file_handle);  
            //---  
            return;  
        }  
        FileClose(file_handle);  
    }  
}
```

Load

Loads the parameters of the object from the file.

```
virtual bool Load(  
    int file_handle    // File handle  
)
```

Parameters

file_handle

[in] handle of the previously opened using the function FileOpen (...) file.

Return Value

true - if successfully completed, false - if an error.

Example:

```
//--- example for CChartObject::Load  
#include <ChartObjects\ChartObject.mqh>  
//---  
void OnStart()  
{  
    int file_handle;  
    CChartObject object;  
    //--- open file  
    file_handle=FileOpen("MyFile.bin", FILE_READ|FILE_BIN|FILE_ANSI);  
    if(file_handle>=0)  
    {  
        if(!object.Load(file_handle))  
        {  
            //--- file load error  
            printf("File load: Error %d!", GetLastError());  
            FileClose(file_handle);  
            //---  
            return;  
        }  
        FileClose(file_handle);  
    }  
    //--- use object  
    //--- . . .  
}
```

Type

Gets the type identifier graphic object.

```
virtual int Type() const
```

Return Value

Object type identifier (0x8888 for [CChartObject](#)) .

Example:

```
//--- example for CChartObject::Type
#include <ChartObjects\ChartObject.mqh>
//---
void OnStart()
{
    CChartObject object;
    //--- get object type
    int type=object.Type();
}
```

Objects Lines

A group of graphic objects "Lines".

This section contains the technical details of working with a group of classes of graphical objects "Lines" and a description of the relevant components of the MQL5 Standard Library .

Class name	Object
CChartObjectVLine	Graphic object "Vertical Line"
CChartObjectHLine	Graphic object "Horizontal Line"
CChartObjectTrend	Graphic object "Trend Line"
CChartObjectTrendByAngle	Graphic object "Trend Line by Angle"
CChartObjectCycles	Graphic object "Cyclic Lines"

See also

[Object types](#), [Graphic objects](#)

CChartObjectVLine

Class CChartObjectVLine is a class for simplified access to "Vertical Line" graphic object properties.

Description

Class CChartObjectVLine provides access to "Vertical Line" object properties.

Declaration

```
class CChartObjectVLine : public CChartObject
```

Title

```
#include <ChartObjects\ChartObjectsLines.mqh>
```

Class Methods

Create	
Create	Creates graphic object "Vertical Line"
Input/output	
virtual Type	Virtual method of identification

See also

[Object types](#), [Graphic objects](#)

Create

Creates graphic object "Vertical Line".

```
bool Create(  
    long      chart_id,      // Chart identifier  
    string    name,          // Object name  
    int       window,        // Chart window  
    datetime  time           // Time coordinate  
)
```

Parameters

chart_id

[in] Chart identifier (0 - current chart) .

name

[in] An unique name of the object to create.

window

[in] Chart window number (0 - base window) .

time

[in] Time coordinate of the anchor point.

Returned value

true if successful, false if error.

Type

Returns object type identifier of the graphic object.

```
int Type() const
```

Returned value

Object type identifier (OBJ__VLINE for CChartObjectVline) .

CChartObjectHLine

Class CChartObjectHLine is a class for simplified access to "Horizontal Line" graphic object properties.

Description

Class CChartObjectHLine provides access to "Horizontal Line" object properties.

Declaration

```
class CChartObjectHLine : public CChartObject
```

Title

```
#include <ChartObjects\ChartObjectsLines.mqh>
```

Class Methods

Create	
Create	Creates graphic object "Horizontal Line"
Input/output	
virtual Type	Virtual method of identification

See also

[Object types](#), [Graphic objects](#)

Create

Creates graphic object "Horizontal Line".

```
bool Create(  
    long    chart_id,      // Chart identifier  
    string  name,          // Object name  
    long    window,        // Chart window  
    double  price          // Price coordinate  
)
```

Parameters

chart_id

[in] Chart identifier (0 - current chart) .

name

[in] An unique name of the object to create.

window

[in] Chart window number (0 - base window) .

price

[in] Price coordinate of the anchor point.

Returned value

true if successful, false if error.

Type

Returns object type identifier of the graphic object.

```
int Type() const
```

Returned value

Object type identifier (OBJ_HLINE for CChartObjectHLine) .

CChartObjectTrend

Class CChartObjectTrend is a class for simplified access to "Trend Line" graphic object properties.

Description

Class CChartObjectTrend provides access to "Trend Line" object properties.

Declaration

```
class CChartObjectTrend : public CChartObject
```

Title

```
#include <ChartObjects\ChartObjectsLines.mqh>
```

Class Methods

Create	
Create	Creates graphic object "Trend Line"
Properties	
RayLeft	Gets/Sets property "Ray Left"
RayRight	Gets/Sets property "Ray Right"
Input/output	
virtual Save	Virtual method for writing to file
virtual Load	Virtual method for reading from file
virtual Type	Virtual method of identification

Derived classes:

- [CChartObjectChannel](#)
- [CChartObjectFibo](#)
- [CChartObjectFiboChannel](#)
- [CChartObjectFiboExpansion](#)
- [CChartObjectGannFan](#)
- [CChartObjectGannGrid](#)
- [CChartObjectPitchfork](#)
- [CChartObjectRegression](#)
- [CChartObjectStdDevChannel](#)
- [CChartObjectTrendByAngle](#)

See also

[Object types](#), [Graphic objects](#)

Create

Creates graphic object "Trend Line".

```
bool Create(  
    long      chart_id,      // Chart identifier  
    string     name,         // Object name  
    int       window,        // Chart window  
    datetime   time1,        // 1st time coordinate  
    double     price1,        // 1st price coordinate  
    datetime   time2,        // 2nd time coordinate  
    double     price2        // 2nd price coordinate  
)
```

Parameters

chart_id

[in] Chart identifier (0 - current chart) .

name

[in] An unique name of the object to create.

window

[in] Chart window number (0 - base window) .

time1

[in] Time coordinate of the first anchor point.

price1

[in] Price coordinate of the first anchor point.

time2

[in] Time coordinate of the second anchor point.

price2

[in] Price coordinate of the second anchor point.

Returned value

true if successful, false if error.

RayLeft (Get Method)

Gets the value of "Ray Left" property.

```
bool RayLeft() const
```

Returned value

The value of "Ray Left" property, assigned to the class instance. If there is no object assigned, it returns false.

RayLeft (Set Method)

Sets new flag value for the "Ray Left" property.

```
bool RayLeft (
    bool ray      // flag
)
```

Parameters

ray

[in] New value of the "Ray Left" property.

Returned value

true if successful, false if flag hasn't changed error.

RayRight (Get Method)

Gets the value of "Ray Right" property.

```
bool RayRight() const
```

Returned value

The value of "Ray Right" property, assigned to the class instance. If there is no object assigned, it returns false.

RayRight (Set Method)

Sets new flag value for the "Ray Right" property.

```
bool RayRight (  
    bool ray      // flag  
)
```

Parameters

ray

[in] New value of the "Ray Right" property.

Returned value

true if successful, false if flag hasn't changed error.

Save

Saves object parameters to file.

```
virtual bool Save(  
    int file_handle    // File handle  
)
```

Parameters

file_handle

[in] handle of the file already opened using by function FileOpen(...) .

Returned value

true if successful, false if error.

Load

Loads object parameters from file.

```
virtual bool Load(  
    int file_handle    // File handle  
)
```

Parameters

file_handle

[in] handle of the file already opened using by function FileOpen(...) .

Returned value

true if successful, false if error.

Type

Returns object type identifier of the graphic object.

```
int Type() const
```

Returned value

Object type identifier (OBJ__TREND for CChartObjectTrend) .

CChartObjectTrendByAngle

Class CChartObjectTrendByAngle is a class for simplified access to "Trend Line by Angle" graphic object properties.

Description

Class CChartObjectTrendByAngle provides access to "Trend Line by Angle" object properties.

Declaration

```
class CChartObjectTrendByAngle : public CChartObjectTrend
```

Title

```
#include <ChartObjects\ChartObjectsLines.mqh>
```

Class Methods

Create	
Create	Creates graphic object "Trend Line by Angle"
Properties	
Angle	Gets/Sets property "Angle"
Input/output	
virtual Type	Virtual method of identification

Derived classes:

- [CChartObjectGannLine](#)

See also

[Object types](#), [Graphic objects](#)

Create

Creates graphic object "Trend Line by Angle"

```
bool Create(  
    long      chart_id,      // Chart identifier  
    string     name,         // Object name  
    long      window,        // Chart window  
    datetime   time1,        // 1st time coordinate  
    double     price1,        // 1st price coordinate  
    datetime   time2,        // 2nd time coordinate  
    double     price2        // 2nd price coordinate  
)
```

Parameters

chart_id

[in] Chart identifier (0 - current chart) .

name

[in] An unique name of the object to create.

window

[in] Chart window number (0 - base window) .

time1

[in] Time coordinate of the first anchor point.

price1

[in] Price coordinate of the first anchor point.

time2

[in] Time coordinate of the second anchor point.

price2

[in] Price coordinate of the second anchor point.

Returned value

true if successful, false if error.

Angle (Get Method)

Gets the value of "Angle" property.

```
double Angle() const
```

Returned value

The value of "Angle" property, assigned to the class instance. If there is no object assigned, it returns EMPTY_VALUE.

Angle (Set Method)

Sets new value for the "Angle" property.

```
bool Angle(  
    double angle    // Angle  
)
```

Parameters

angle

[in] New value of the "Angle" property.

Returned value

true if successful, false if property hasn't changed.

Type

Returns object type identifier of the graphic object.

```
int Type() const
```

Returned value

Object type identifier (OBJ__TRENDBYANGLE for CChartObjectTrendByAngle) .

CChartObjectCycles

Class CChartObjectCycles is a class for simplified access to "Cyclic Lines" graphic object properties.

Description

Class CChartObjectCycles provides access to "Cyclic Lines" object properties.

Declaration

```
class CChartObjectCycles : public CChartObject
```

Title

```
#include <ChartObjects\ChartObjectsLines.mqh>
```

Class Methods

Create	
Create	Creates graphic object "Cycle Lines"
Input/output	
virtual Type	Virtual method of identification

See also

[Object types](#), [Graphic objects](#)

Create

Creates graphic object "Cyclic Lines"

```
bool Create(  
    long      chart_id,      // Chart identifier  
    string    name,          // Object name  
    long      window,        // Chart window  
    datetime  time1,         // 1st time coordinate  
    double    price1,        // 1st price coordinate  
    datetime  time2,         // 2nd time coordinate  
    double    price2         // 2nd price coordinate  
)
```

Parameters

chart_id

[in] Chart identifier (0 - current chart) .

name

[in] An unique name of the object to create.

window

[in] Chart window number (0 - base window) .

time1

[in] Time coordinate of the first anchor point.

price1

[in] Price coordinate of the first anchor point.

time2

[in] Time coordinate of the second anchor point.

price2

[in] Price coordinate of the second anchor point.

Returned value

true if successful, false if error.

Type

Returns object type identifier of the graphic object.

```
int Type() const
```

Returned value

Object type identifier (OBJ__CYCLES for CChartObjectCycles) .

Objects Channels

A group of graphic objects "Channels".

This section contains the technical details of working with a group of classes of graphical objects "Channels" and a description of the relevant components of the MQL5 Standard Library .

Class name	Object
<u>CChartObjectChannel</u>	Graphic object "Equidistant Channel"
<u>CChartObjectRegression</u>	Graphic object "Linear Regression Channel"
<u>CChartObjectStdDevChannel</u>	Graphic object "Standard deviations Channel"
<u>CChartObjectPitchfork</u>	Graphic object "Andrew's Pitchfork"

See also

[Object types](#), [Graphic objects](#)

CChartObjectChannel

The CChartObjectChannel is a class for simplified access to "Equidistant Channel" graphic object properties.

Description

The class CChartObjectChannel provides access to "Equidistant Channel" object properties.

Declaration

```
class CChartObjectChannel : public CChartObjectTrend
```

Title

```
#include <ChartObjects\ChartObjectsChannels.mqh>
```

Class Methods

Create	
Create	Creates graphic object "Equidistant Channel"
Input/output	
virtual Type	Virtual method of identification

See also

[Object types](#), [Graphic objects](#)

Create

Creates graphic object "Equidistant Channel"

```
bool Create(  
    long      chart_id,      // Chart identifier  
    string    name,          // Object name  
    int       window,        // Chart window  
    datetime  time1,         // Time coordinate  
    double    price1,  
    datetime  time2,  
    double    price2,  
    datetime  time3,  
    double    price3  
)
```

Parameters

chart_id

[in] Chart ID (0 - current chart) .

name

[in] An unique name of the object to create.

window

[in] Chart window number (0 - base window) .

time1

[in] Time coordinate of the first anchor point.

price1

[in] Price coordinate of the first anchor point.

time2

[in] Time coordinate of the second anchor point.

price2

[in] Price coordinate of the second anchor point.

time3

[in] Time coordinate of the third anchor point.

price3

[in] Price coordinate of the third anchor point.

Returned value

true if successful, false if error.

Type

Returns object type identifier of the graphic object.

```
int Type() const
```

Returned value

Object type identifier (OBJ__CHANNEL for CChartObjectChannel) .

CChartObjectRegression

Class CChartObjectRegression is a class for simplified access to "Linear Regression Channel" graphic object properties.

Description

Class CChartObjectRegression provides access to "Linear Regression Channel" object properties.

Declaration

```
class CChartObjectRegression : public CChartObjectTrend
```

Title

```
#include <ChartObjects\ChartObjectsChannels.mqh>
```

Class Methods

Create	
Create	Creates graphic object "Linear Regression Channel"
Input/output	
virtual Type	Virtual method of identification

See also

[Object types](#), [Graphic objects](#)

Create

Creates graphic object "Linear Regression Channel"

```
bool Create(  
    long      chart_id,      // Chart identifier  
    string    name,          // Object name  
    long      window,        // Chart window  
    datetime  time1,         // First time coordinate  
    datetime  time2          // Second time coordinate  
)
```

Parameters

chart_id

[in] Chart identifier (0 - current chart) .

name

[in] An unique name of the object to create.

window

[in] Chart window number (0 - base window) .

time1

[in] Time coordinate of the first anchor point.

time2

[in] Time coordinate of the second anchor point.

Returned value

true if successful, false if error.

Type

Returns object type identifier of the graphic object.

```
int Type() const
```

Returned value

Object type identifier (OBJ__REGRESSION for CChartObjectRegression) .

CChartObjectStdDevChannel

Class CChartObjectStdDevChannel is a class for simplified access to "Standard Deviation Channel" graphic object properties.

Description

Class CChartObjectStdDevChannel provides access to "Standard Deviation Channel" object properties.

Declaration

```
class CChartObjectStdDevChannel : public CChartObjectTrend
```

Title

```
#include <ChartObjects\ChartObjectsChannels.mqh>
```

Class Methods

Create	
Create	Creates graphic object "Standard Deviation Channel"
Properties	
Deviations	Gets/Sets property "Deviation"
Input/output	
virtual Save	Virtual method for writing to file
virtual Load	Virtual method for reading from file
virtual Type	Virtual method of identification

See also

[Object types](#), [Graphic objects](#)

Create

Creates graphic object "Standard Deviation Channel"

```
bool Create(  
    long      chart_id,      // Chart identifier  
    string    name,          // Object name  
    int       window,        // Chart window  
    datetime  time1,         // First time coordinate  
    datetime  time2,         // Second time coordinate  
    double    deviation      // Deviation  
)
```

Parameters

chart_id

[in] Chart identifier (0 - current chart) .

name

[in] An unique name of the object to create.

window

[in] Chart window number (0 - base window) .

time1

[in] Time coordinate of the first anchor point.

time2

[in] Time coordinate of the second anchor point.

deviation

[in] Numerical value for "Deviation" property.

Returned value

true if successful, false if error.

Deviation (Get Method)

Gets numerical value for "Deviation" property.

```
double Deviation() const
```

Returned value

Numerical value of "Deviation" property, assigned to the class instance. If there is no object assigned, it returns EMPTY_VALUE.

Deviation (Set Method)

Sets numerical value for "Deviation" property.

```
bool Deviation(  
    double deviation    // Deviation  
)
```

Parameters

deviation

[in] New value for "Deviation" property.

Returned value

true if successful, false if property hasn't changed.

Save

Saves object parameters to file.

```
virtual bool Save(  
    int file_handle    // File handle  
)
```

Parameters

file_handle

[in] handle of the file already opened using by function FileOpen(...) .

Returned value

true if successful, false if error.

Load

Loads object parameters from file.

```
virtual bool Load(  
    int file_handle    // File handle  
)
```

Parameters

file_handle

[in] handle of the file already opened using by function FileOpen(...) .

Returned value

true if successful, false if error.

Type

Returns object type identifier of the graphic object.

```
int Type() const
```

Returned value

Object type identifier (OBJ__STDDEVCHANNEL for CChartObjectStdDevChannel) .

CChartObjectPitchfork

Class CChartObjectPitchfork is a class for simplified access to "Andrew's Pitchfork" graphic object properties.

Description

Class CChartObjectPitchfork provides access to "Andrew's Pitchfork" object properties.

Declaration

```
class CChartObjectPitchfork : public CChartObjectTrend
```

Title

```
#include <ChartObjects\ChartObjectsChannels.mqh>
```

Class Methods

Create	
Create	Creates graphic object "Andrew's Pitchfork"
Input/output	
virtual Type	Virtual method of identification

See also

[Object types](#), [Graphic objects](#)

Create

Creates graphic object "Andrew's Pitchfork"

```
bool Create(  
    long      chart_id,      // Chart identifier  
    string    name,          // Object name  
    long      window,        // Chart window  
    datetime  time1,         // 1st time coordinate  
    double    price1,        // 1st price coordinate  
    datetime  time2,         // ...  
    double    price2,  
    datetime  time3,  
    double    price3  
)
```

Parameters

chart_id

[in] Chart identifier (0 - current chart) .

name

[in] An unique name of the object to create.

window

[in] Chart window number (0 - base window) .

time1

[in] Time coordinate of the first anchor point.

price1

[in] Price coordinate of the first anchor point..

time2

[in] Time coordinate of the second anchor point.

price2

[in] Price coordinate of the first anchor point.

time3

[in] Time coordinate of the third anchor point.

price3

[in] Price coordinate of the third anchor point.

Returned value

true if successful, false if error.

Type

Returns object type identifier of the graphic object.

```
int Type() const
```

Returned value

Object type identifier (OBJ_PITCHFORK for CChartObjectPitchfork) .

Gann Tools

A group of graphic objects "Gann Tools".

This section contains the technical details of working with a group of classes of graphical objects "Gann Tools" and a description of the relevant components of the MQL5 Standard Library .

Class name	Object
CChartObjectGannLine	Graphic object "Gann Line"
CChartObjectGannFan	Graphic object "Gann Fan"
CChartObjectGannGrid	Graphic object "Gann Grid"

See also

[Object types](#), [Graphic objects](#)

CChartObjectGannLine

Class CChartObjectGannLine is a class for simplified access to "Gann Line" graphic object properties.

Description

Class CChartObjectGannLine provides access to "Gann Line" object properties.

Declaration

```
class CChartObjectGannLine : public CChartObjectTrendByAngle
```

Title

```
#include <ChartObjects\ChartObjectsGann.mqh>
```

Class Methods

Create	
Create	Creates graphic object "Gann Line"
Properties	
PipsPerBar	Gets/Sets property "Scale"
Input/output	
virtual Save	Virtual method for writing to file
virtual Load	Virtual method for reading from file
virtual Type	Virtual method of identification

See also

[Object types](#), [Graphic objects](#)

Create

Creates graphic object "Gann Line".

```
bool Create(  
    long      chart_id,      // Chart identifier  
    string     name,         // Object name  
    int       window,       // Chart window  
    datetime  time1,        // First time coordinate  
    double    price1,       // First price coordinate  
    datetime  time2,        // Second time coordinate  
    double    ppb           // Pips per bar  
)
```

Parameters

chart_id

[in] Chart identifier (0 - current chart) .

name

[in] An unique name of the object to create.

window

[in] Chart window number (0 - base window) .

time1

[in] Time coordinate of the first anchor point.

price1

[in] Price coordinate of the first anchor point.

time2

[in] Time coordinate of the second anchor point.

ppb

[in] Pips per bar.

Returned value

true if successful, false if error.

PipsPerBar (Get Method)

Gets the value of "Pips per bar" property.

```
double PipsPerBar() const
```

Returned value

Value of "Pips per bar" property of the object, assigned to the class instance. If there is no object assigned, it returns EMPTY_VALUE.

PipsPerBar (Set Method)

Sets new value for "Pips per bar" property.

```
bool PipsPerBar(  
    double ppb    // Pips per bar  
)
```

Parameters

ppb

[in] New value for "Pips per bar" property.

Returned value

true if successful, false if property hasn't changed.

Save

Saves object parameters to file.

```
virtual bool Save(  
    int file_handle    // File handle  
)
```

Parameters

file_handle

[in] handle of the binary file already opened using by function FileOpen(...) .

Returned value

true if successful, false if error.

Load

Loads object parameters from file.

```
virtual bool Load(  
    int file_handle    // File handle  
)
```

Parameters

file_handle

[in] handle of the binary file already opened using by function FileOpen(...) .

Returned value

true if successful, false if error.

Type

Returns graphic object type identifier.

```
virtual int Type() const
```

Returned value

Object type identifier (OBJ__GANNLINE for CChartObjectGannLine) .

CChartObjectGannFan

Class CChartObjectGannFan is a class for simplified access to "Gann Fan" graphic object properties.

Description

Class CChartObjectGannFan provides access to "Gann Fan" object properties.

Declaration

```
class CChartObjectGannFan : public CChartObjectTrend
```

Title

```
#include <ChartObjects\ChartObjectsGann.mqh>
```

Class Methods

Create	
Create	Creates graphic object "Gann Fan"
Properties	
PipsPerBar	Gets/Sets property "Pips per bar"
Downtrend	Gets/Sets property "Downtrend"
Input/output	
virtual Save	Virtual method for writing to file
virtual Load	Virtual method for reading from file
virtual Type	Virtual method of identification

See also

[Object types](#), [Graphic objects](#)

Create

Creates graphic object "Gann Fan".

```
bool Create(  
    long      chart_id,      // Chart identifier  
    string     name,         // Object name  
    int        window,       // Chart window  
    datetime   time1,        // First time coordinate  
    double     price1,       // First price coordinate  
    datetime   time2,        // Second time coordinate  
    double     ppb           // Pips per bar  
)
```

Parameters

chart_id

[in] Chart identifier (0 - current chart) .

name

[in] An unique name of the object to create.

window

[in] Chart window number (0 - base window) .

time1

[in] Time coordinate of the first anchor point.

price1

[in] Price coordinate of the first anchor point.

time2

[in] Time coordinate of the second anchor point.

ppb

[in] Pips per bar.

Returned value

true if successful, false if error.

PipsPerBar (Get Method)

Gets the value of "Pips per bar" property.

```
double PipsPerBar() const
```

Returned value

Value of "Pips per bar" property of the object, assigned to the class instance. If there is no object assigned, it returns EMPTY_VALUE.

PipsPerBar (Set Method)

Sets new value for "Pips per bar" property.

```
bool PipsPerBar(  
    double ppb    // Pips per bar  
)
```

Parameters

ppb

[in] New value for "Pips per bar" property.

Returned value

true if successful, false if property hasn't changed.

Downtrend (Get Method)

Gets the value of "Downtrend" property.

```
bool Downtrend() const
```

Returned value

Value of the "Downtrend" property of the object, assigned to the class instance. If there is no object assigned, it returns false.

Downtrend (Set Method)

Sets new value of "Downtrend" property.

```
bool Downtrend(  
    bool downtrend    // Flag value  
)
```

Parameters

downtrend

[in] New value for "Downtrend" property.

Returned value

true if successful, false if property hasn't changed.

Save

Saves object parameters to file.

```
virtual bool Save(  
    int file_handle    // File handle  
)
```

Parameters

file_handle

[in] handle of the binary file already opened using by function FileOpen(...) .

Returned value

true if successful, false if error.

Load

Loads object parameters from file.

```
virtual bool Load(  
    int file_handle    // File handle  
)
```

Parameters

file_handle

[in] handle of the binary file already opened using by function FileOpen(...) .

Returned value

true if successful, false if error.

Type

Returns graphic object type identifier.

```
virtual int Type() const
```

Returned value

Object type identifier (OBJ__GANNFAN for CChartObjectGannFan) .

CChartObjectGannGrid

Class CChartObjectGannGrid is a class for simplified access to "Gann Grid" graphic object properties.

Description

Class CChartObjectGannGrid provides access to "Gann Grid" object properties.

Declaration

```
class CChartObjectGannGrid : public CChartObjectTrend
```

Title

```
#include <ChartObjects\ChartObjectsGann.mqh>
```

Class Methods

Create	
Create	Creates graphic object "Gann Grid"
Properties	
PipsPerBar	Gets/Sets property "Pips per bar"
Downtrend	Gets/Sets property "Downtrend"
Input/output	
virtual Save	Virtual method for writing to file
virtual Load	Virtual method for reading from file
virtual Type	Virtual method of identification

See also

[Object types](#), [Graphic objects](#)

Create

Creates graphic object "Gann Grid".

```
bool Create(  
    long      chart_id,      // Chart identifier  
    string    name,          // Object name  
    int       window,        // Chart window  
    datetime  time1,         // First time coordinate  
    double    price1,        // First price coordinate  
    datetime  time2,         // Second time coordinate  
    double    ppb            // Pips per bar  
)
```

Parameters

chart_id

[in] Chart identifier (0 - current chart) .

name

[in] An unique name of the object to create.

window

[in] Chart window number (0 - base window) .

time1

[in] Time coordinate of the first anchor point.

price1

[in] Price coordinate of the first anchor point.

time2

[in] Time coordinate of the second anchor point.

ppb

[in] Pips per bar.

Returned value

true if successful, false if error.

PipsPerBar (Get Method)

Gets the value of "Pips per bar" property.

```
double PipsPerBar() const
```

Returned value

Value of "Pips per bar" property of the object, assigned to the class instance. If there is no object assigned, it returns EMPTY_VALUE.

PipsPerBar (Set Method)

Sets new value for "Pips per bar" property.

```
bool PipsPerBar(  
    double ppb    // Pips per bar  
)
```

Parameters

ppb

[in] New value for "Pips per bar" property.

Returned value

true if successful, false if property hasn't changed.

Downtrend (Get Method)

Gets the value of "Downtrend" property.

```
bool Downtrend() const
```

Returned value

Value of "Downtrend" property of the object, assigned to the class instance. If there is no object assigned, it returns false.

Downtrend (Set Method)

Sets new value of "Downtrend" property.

```
bool Downtrend(  
    bool downtrend    // Flag value  
)
```

Parameters

downtrend

[in] New value for "Downtrend" property.

Returned value

true if successful, false if property hasn't changed.

Save

Saves object parameters to file.

```
virtual bool Save(  
    int file_handle    // File handle  
)
```

Parameters

file_handle

[in] handle of the binary file already opened using by function FileOpen(...) .

Returned value

true if successful, false if error.

Load

Loads object parameters from file.

```
virtual bool Load(  
    int file_handle    // File handle  
)
```

Parameters

file_handle

[in] handle of the binary file already opened using by function FileOpen(...) .

Returned value

true if successful, false if error.

Type

Returns graphic object type identifier.

```
virtual int Type() const
```

Returned value

Object type identifier (OBJ__GANNGRID for CChartObjectGannGrid) .

Fibonacci Tools

A group of graphic objects "Fibonacci Tools".

This section contains the technical details of working with a group of classes of graphical objects "Fibonacci Tools" and a description of the relevant components of the MQL5 Standard Library .

Class name	Object
CChartObjectFibo	Graphic object "Fibonacci Retracement"
CChartObjectFiboTimes	Graphic object "Fibonacci Time Zones"
CChartObjectFiboFan	Graphic object "Fibonacci Fan"
CChartObjectFiboArc	Graphic object "Fibonacci Arc"
CChartObjectFiboChannel	Graphic object "Fibonacci Channel"
CChartObjectFiboExpansion	Graphic object "Fibonacci Expansion"

See also

[Object types](#), [Graphic objects](#)

CChartObjectFibo

Class CChartObjectFibo is a class for simplified access to "Fibonacci Retracement" graphic object properties.

Description

Class CChartObjectFibo provides access to "Fibonacci Retracement" object properties.

Declaration

```
class CChartObjectFibo : public CChartObjectTrend
```

Title

```
#include <ChartObjects\ChartObjectsFibo.mqh>
```

Class Methods

Create	
Create	Creates graphic object "Fibonacci Retracement"
Input/output	
virtual Type	Virtual method of identification

See also

[Object types](#), [Graphic objects](#)

Create

Creates graphic object "Fibonacci Retracement".

```
bool Create(  
    long      chart_id,      // Chart identifier  
    string     name,         // Object name  
    int        window,       // Chart window  
    datetime   time1,        // First time coordinate  
    double     price1,       // First price coordinate  
    datetime   time2,        // Second time coordinate  
    double     price2        // Second price coordinate  
)
```

Parameters

chart_id

[in] Chart identifier (0 - current chart) .

name

[in] An unique name of the object to create.

window

[in] Chart window number (0 - base window) .

time1

[in] Time coordinate of the first anchor point.

price1

[in] Price coordinate of the first anchor point.

time2

[in] Time coordinate of the second anchor point.

price2

[in] Price coordinate of the second anchor point.

Returned value

true if successful, false if error.

Type

Returns graphic object type identifier.

```
virtual int Type() const
```

Returned value

Object type identifier (OBJ__FIBO for CChartObjectFibo) .

CChartObjectFiboTimes

Class CChartObjectFiboTimes is a class for simplified access to "Fibonacci Time Zones" graphic object properties.

Description

Class CChartObjectFiboTimes provides access to "Fibonacci Time Zones" object properties.

Declaration

```
class CChartObjectFiboTimes : public CChartObject
```

Title

```
#include <ChartObjects\ChartObjectsFibo.mqh>
```

Class Methods

Create	
Create	Creates graphic object "Fibonacci Time Zones"
Input/output	
virtual Type	Virtual method of identification

See also

[Object types](#), [Graphic objects](#)

Create

Creates graphic object "Fibonacci Time Zones".

```
bool Create(  
    long      chart_id,      // Chart identifier  
    string    name,          // Object name  
    int       window,        // Chart window  
    datetime  time1,         // First time coordinate  
    double    price1,        // First price coordinate  
    datetime  time2,         // Second time coordinate  
    double    price2         // Second price coordinate  
)
```

Parameters

chart_id

[in] Chart identifier (0 - current chart) .

name

[in] An unique name of the object to create.

window

[in] Chart window number (0 - base window) .

time1

[in] Time coordinate of the first anchor point.

price1

[in] Price coordinate of the first anchor point.

time2

[in] Time coordinate of the second anchor point.

price2

[in] Price coordinate of the second anchor point.

Returned value

true if successful, false if error.

Type

Returns graphic object type identifier.

```
virtual int Type() const
```

Returned value

Object type identifier (OBJ__FIBOTIMES for CChartObjectFiboTimes) .

CChartObjectFiboFan

Class CChartObjectFiboFan is a class for simplified access to "Fibonacci Fan" graphic object properties.

Description

Class CChartObjectFiboFan provides access to "Fibonacci Fan" object properties.

Declaration

```
class CChartObjectFiboFan : public CChartObject
```

Title

```
#include <ChartObjects\ChartObjectsFibo.mqh>
```

Class Methods

Create	
Create	Creates graphic object "Fibonacci Fan"
Input/output	
virtual Type	Virtual method of identification

See also

[Object types](#), [Graphic objects](#)

Create

Creates graphic object "Fibonacci Fan".

```
bool Create(  
    long      chart_id,      // Chart identifier  
    string    name,          // Object name  
    int       window,        // Chart window  
    datetime  time1,         // First time coordinate  
    double    price1,        // First price coordinate  
    datetime  time2,         // Second time coordinate  
    double    price2         // Second price coordinate  
)
```

Parameters

chart_id

[in] Chart identifier (0 - current chart) .

name

[in] An unique name of the object to create.

window

[in] Chart window number (0 - base window) .

time1

[in] Time coordinate of the first anchor point.

price1

[in] Price coordinate of the first anchor point.

time2

[in] Time coordinate of the second anchor point.

price2

[in] Price coordinate of the second anchor point.

Returned value

true if successful, false if error.

Type

Returns graphic object type identifier.

```
virtual int Type() const
```

Returned value

Object type identifier (OBJ__FIBOFAN for CChartObjectFiboFan) .

CChartObjectFiboArc

Class CChartObjectFiboArc is a class for simplified access to "Fibonacci Arc" graphic object properties.

Description

Class CChartObjectFiboArc provides access to "Fibonacci Arc" object properties.

Declaration

```
class CChartObjectFiboArc : public CChartObject
```

Title

```
#include <ChartObjects\ChartObjectsFibo.mqh>
```

Class Methods

Create	
Create	Creates graphic object "Fibonacci Arc"
Properties	
Scale	Gets/Sets property "Scale"
Ellipse	Gets/Sets property "Ellipse"
Input/output	
virtual Save	Virtual method for writing to file
virtual Load	Virtual method for reading from file
virtual Type	Virtual method of identification

See also

[Object types](#), [Graphic objects](#)

Create

Creates graphic object "Fibonacci Arc"

```
bool Create(  
    long      chart_id,      // Chart identifier  
    string    name,          // Object name  
    int       window,        // Chart window  
    datetime  time1,         // First time coordinate  
    double    price1,        // First price coordinate  
    datetime  time2,         // Second time coordinate  
    double    price2,        // Second price coordinate  
    double    scale          // Scale  
)
```

Parameters

chart_id

[in] Chart identifier (0 - current chart) .

name

[in] An unique name of the object to create.

window

[in] Chart window number (0 - base window) .

time1

[in] Time coordinate of the first anchor point.

price1

[in] Price coordinate of the first anchor point.

time2

[in] Time coordinate of the second anchor point.

price2

[in] Price coordinate of the second anchor point.

scale

[in] Scale.

Returned value

true if successful, false if error.

Scale (Get Method)

Gets the value of "Scale" property.

```
double Scale() const
```

Returned value

Value of "Scale" property of the object, assigned to the class instance. If there is no object assigned, it returns EMPTY_VALUE.

Scale (Set Method)

Sets new value for "Scale" property.

```
bool Scale(  
    double scale    // Scale  
)
```

Parameters

scale

[in] New value for "Scale" property.

Returned value

true if successful, false if property hasn't changed.

Ellipse (Get Method)

Gets the value of "Ellipse" property.

```
bool Ellipse() const
```

Returned value

Value of "Ellipse" property of the object, assigned to the class instance. If there is no object assigned, it returns false.

Ellipse (Set Method)

Sets new flag value for "Ellipse" property.

```
bool Ellipse(  
    bool ellipse    // flag value  
)
```

Parameters

ellipse

[in] New value for "Scale" property.

Returned value

true if successful, false if property hasn't changed.

Save

Saves object parameters to file.

```
virtual bool Save(  
    int file_handle    // File handle  
)
```

Parameters

file_handle

[in] handle of the binary file already opened using by function FileOpen(...) .

Returned value

true if successful, false if error.

Load

Loads object parameters from file.

```
virtual bool Load(  
    int file_handle    // File handle  
)
```

Parameters

file_handle

[in] handle of the binary file already opened using by function FileOpen(...) .

Returned value

true if successful, false if error.

Type

Returns graphic object type identifier.

```
virtual int Type() const
```

Returned value

Object type identifier (OBJ_FIBOARC for CChartObjectFiboArc) .

CChartObjectFiboChannel

Class CChartObjectFiboChannel is a class for simplified access to "Fibonacci Channel" graphic object properties.

Description

Class CChartObjectFiboChannel provides access to "Fibonacci Channel" object properties.

Declaration

```
class CChartObjectFiboChannel : public CChartObjectTrend
```

Title

```
#include <ChartObjects\ChartObjectsFibo.mqh>
```

Class Methods

Create	
Create	Creates graphic object "Fibonacci Channel"
Input/output	
virtual Type	Virtual method of identification

See also

[Object types](#), [Graphic objects](#)

Create

Creates graphic object "Fibonacci Channel".

```
bool Create(  
    long      chart_id,      // Chart identifier  
    string     name,         // Object name  
    int       window,        // Chart window  
    datetime  time1,         // First time coordinate  
    double    price1,        // First price coordinate  
    datetime  time2,         // Second time coordinate  
    double    price2,        // Second price coordinate  
    datetime  time3,         // Third time coordinate  
    double    price3         // Third price coordinate  
)
```

Parameters

chart_id

[in] Chart identifier (0 - current chart) .

name

[in] An unique name of the object to create.

window

[in] Chart window number (0 - base window) .

time1

[in] Time coordinate of the first anchor point.

price1

[in] Price coordinate of the first anchor point.

time2

[in] Time coordinate of the second anchor point.

price2

[in] Price coordinate of the second anchor point.

time3

[in] Time coordinate of the third anchor point.

price3

[in] Price coordinate of the third anchor point.

Returned value

true if successful, false if error.

Type

Returns graphic object type identifier.

```
virtual int Type() const
```

Returned value

Object type identifier (OBJ__FIBOCHANNEL for CChartObjectFiboChannel) .

CChartObjectFiboExpansion

Class CChartObjectFiboExpansion is a class for simplified access to "Fibonacci Expansion" graphic object properties.

Description

Class CChartObjectFiboExpansion provides access to "Fibonacci Expansion" object properties.

Declaration

```
class CChartObjectFiboExpansion : public CChartObjectTrend
```

Title

```
#include <ChartObjects\ChartObjectsFibo.mqh>
```

Class Methods

Create	
Create	Creates graphic object "Fibonacci Expansion"
Input/output	
virtual Type	Virtual method of identification

See also

[Object types](#), [Graphic objects](#)

Create

Creates graphic object "Fibonacci Expansion".

```
bool Create(  
    long      chart_id,      // Chart identifier  
    string    name,         // Object name  
    int       window,       // Chart window  
    datetime  time1,        // First time coordinate  
    double    price1,       // First price coordinate  
    datetime  time2,        // Second time coordinate  
    double    price2,       // Second price coordinate  
    datetime  time3,        // Third time coordinate  
    double    price3        // Third price coordinate  
)
```

Parameters

chart_id

[in] Chart identifier (0 - current chart) .

name

[in] An unique name of the object to create.

window

[in] Chart window number (0 - base window) .

time1

[in] Time coordinate of the first anchor point.

price1

[in] Price coordinate of the first anchor point.

time2

[in] Time coordinate of the second anchor point.

price2

[in] Price coordinate of the second anchor point.

time3

[in] Time coordinate of the third anchor point.

price3

[in] Price coordinate of the third anchor point.

Returned value

true if successful, false if error.

Type

Returns graphic object type identifier.

```
virtual int Type() const
```

Returned value

Object type identifier (OBJ__EXPANSION for CChartObjectFiboExpansion) .

Elliott Tools

A group of graphic objects "Elliott Tools".

This section contains the technical details of working with a group of classes of graphical objects "Elliott Tools" and a description of the relevant components of the MQL5 Standard Library .

Class name	Object
<u>CChartObjectElliottWave3</u>	Graphic object "Correcting Wave"
<u>CChartObjectElliottWave5</u>	Graphic object "Impulse Wave"

See also

[Object types](#), [Graphic objects](#)

CChartObjectElliottWave3

Class CChartObjectElliottWave3 is a class for simplified access to "Correcting Wave" graphic object properties.

Description

Class CChartObjectElliottWave3 provides access to "Correcting Wave" object properties.

Declaration

```
class CChartObjectElliottWave3 : public CChartObject
```

Title

```
#include <ChartObjects\ChartObjectsElliott.mqh>
```

Class Methods

Create	
Create	Creates graphic object "Correcting Wave"
Properties	
Degree	Gets/Sets property "Degree"
Lines	Gets/Sets property "Lines"
Input/output	
virtual Save	Virtual method for writing to file
virtual Load	Virtual method for reading from file
virtual Type	Virtual method of identification

Derived classes:

- [CChartObjectElliottWave5](#)

See also

[Object types](#), [Graphic objects](#)

Create

Creates graphic object "Correcting Wave".

```
bool Create(  
    long      chart_id,      // Chart identifier  
    string     name,         // Object name  
    int       window,        // Chart window  
    datetime   time1,        // First time coordinate  
    double     price1,       // First price coordinate  
    datetime   time2,        // Second time coordinate  
    double     price2,       // Second price coordinate  
    datetime   time3,        // Third time coordinate  
    double     price3        // Third price coordinate  
)
```

Parameters

chart_id

[in] Chart identifier (0 - current chart) .

name

[in] An unique name of the object to create.

window

[in] Chart window number (0 - base window) .

time1

[in] Time coordinate of the first anchor point.

price1

[in] Price coordinate of the first anchor point.

time2

[in] Time coordinate of the second anchor point.

price2

[in] Price coordinate of the second anchor point.

time3

[in] Time coordinate of the third anchor point.

price3

[in] Time coordinate of the third anchor point.

Returned value

true if successful, false if error.

Degree (Get Method)

Gets the value of "Degree" property.

```
ENUM_ELLIOT_WAVE_DEGREE Degree() const
```

Returned value

Value of "Degree" property of the object, assigned to the class instance. If there is no object assigned, it returns WRONG_VALUE.

Degree (Set Method)

Sets new value for "Degree" property.

```
bool Degree(  
    ENUM_ELLIOT_WAVE_DEGREE degree // property value  
)
```

Parameters

degree

[in] New value for "Degree" property.

Returned value

true if successful, false if property hasn't changed.

Lines (Get Method)

Gets the value of "Lines" property.

```
bool Lines() const
```

Returned value

Value of "Lines" property of the object, assigned to the class instance. If there is no object assigned, it returns false.

Lines (Set Method)

Sets new value for "Lines" property.

```
bool Lines(  
    bool lines    // flag value  
)
```

Parameters

lines

[in] New value for "Lines" property.

Returned value

true if successful, false if flag hasn't changed.

Save

Saves object parameters to file.

```
virtual bool Save(  
    int file_handle    // File handle  
)
```

Parameters

file_handle

[in] handle of the binary file already opened using by function FileOpen(...) .

Returned value

true if successful, false if error.

Load

Loads object parameters from file.

```
virtual bool Load(  
    int file_handle    // File handle  
)
```

Parameters

file_handle

[in] handle of the binary file already opened using by function FileOpen(...) .

Returned value

true if successful, false if error.

Type

Returns graphic object type identifier.

```
virtual int Type() const
```

Returned value

Object type identifier (OBJ__ELLIOTWAVE3 for CChartObjectElliottWave3) .

CChartObjectElliottWave5

Class CChartObjectElliottWave5 is a class for simplified access to "Impulse Wave" graphic object properties.

Description

Class CChartObjectElliottWave5 provides access to "Impulse Wave" object properties.

Declaration

```
class CChartObjectElliottWave5 : public CChartObjectElliottWave3
```

Title

```
#include <ChartObjects\ChartObjectsElliott.mqh>
```

Class Methods

Create	
Create	Creates graphic object "Impulse Wave"
Input/output	
virtual Type	Virtual method of identification

See also

[Object types](#), [Graphic objects](#)

Create

Creates graphic object "Impulse Wave".

```
bool Create(
    long      chart_id,      // Chart identifier
    string     name,         // Object name
    int        window,       // Chart window
    datetime   time1,        // First time coordinate
    double     price1,       // First price coordinate
    datetime   time2,        // Second time coordinate
    double     price2,       // Second price coordinate
    datetime   time3,        // Third time coordinate
    double     price3,       // Third price coordinate
    datetime   time4,        // Fourth time coordinate
    double     price4,       // Fourth price coordinate
    datetime   time5,        // Fifth time coordinate
    double     price5,       // Fifth price coordinate
)
```

Parameters

chart_id

[in] Chart identifier (0 - current chart) .

name

[in] An unique name of the object to create.

window

[in] Chart window number (0 - base window) .

time1

[in] Time coordinate of the first anchor point.

price1

[in] Price coordinate of the first anchor point.

time2

[in] Time coordinate of the second anchor point.

price2

[in] Price coordinate of the second anchor point.

time3

[in] Time coordinate of the third anchor point.

price3

[in] Price coordinate of the third anchor point.

time4

[in] Time coordinate of the fourth anchor point.

price4

[in] Price coordinate of the fourth anchor point.

time5

[in] Time coordinate of the fifth anchor point.

price5

[in] Price coordinate of the fifth anchor point.

Returned value

true if successful, false if error.

Type

Returns graphic object type identifier.

```
virtual int Type() const
```

Returned value

Object type identifier (OBJ__ELLIOTWAVE5 for CChartObjectElliottWave5) .

Objects Shapes

A group of graphic objects "Shapes".

This section contains the technical details of working with a group of classes of graphical objects "Shapes" and a description of the relevant components of the MQL5 Standard Library .

Class name	Object
CChartObjectRectangle	Graphic object "Rectangle"
CChartObjectTriangle	Graphic object "Triangle"
CChartObjectEllipse	Graphic object "Ellipse"

See also

[Object types](#), [Graphic objects](#)

CChartObjectRectangle

Class CChartObjectRectangle is a class for simplified access to "Rectangle" graphic object properties.

Description

Class CChartObjectRectangle provides access to "Rectangle" object properties.

Declaration

```
class CChartObjectRectangle : public CChartObject
```

Title

```
#include <ChartObjects\ChartObjectsShapes.mqh>
```

Class Methods

Create	
Create	Creates graphic object "Rectangle"
Input/output	
virtual Type	Virtual method of identification

See also

[Object types](#), [Graphic objects](#)

Create

Creates graphic object "Rectangle".

```
bool Create(  
    long      chart_id,      // Chart identifier  
    string    name,          // Object name  
    long      window,        // Chart window  
    datetime  time1,         // First time coordinate  
    double    price1,        // First price coordinate  
    datetime  time2,         // Second time coordinate  
    double    price2         // Second price coordinate  
)
```

Parameters

chart_id

[in] Chart identifier (0 - current chart) .

name

[in] An unique name of the object to create.

window

[in] Chart window number (0 - base window) .

time1

[in] Time coordinate of the first anchor point.

price1

[in] Price coordinate of the first anchor point.

time2

[in] Time coordinate of the second anchor point.

price2

[in] Price coordinate of the second anchor point.

Returned value

true if successful, 0 if error

Type

Returns object type identifier of the graphic object.

```
int Type() const
```

Returned value

Object type identifier (OBJ__RECTANGLE for CChartObjectRectangle) .

CChartObjectTriangle

Class CChartObjectTriangle is a class for simplified access to "Triangle" graphic object properties.

Description

Class CChartObjectTriangle provides access to "Triangle" object properties.

Declaration

```
class CChartObjectTriangle : public CChartObject
```

Title

```
#include <ChartObjects\ChartObjectsShapes.mqh>
```

Class Methods

Create	
Create	Creates graphic object "Triangle"
Input/output	
virtual Type	Virtual method of identification

See also

[Object types](#), [Graphic objects](#)

Create

Creates graphic object "Triangle".

```
bool Create(  
    long      chart_id,      // Chart identifier  
    string    name,          // Object name  
    long      window,        // Chart window  
    datetime  time1,         // First time coordinate  
    double    price1,        // First price coordinate  
    datetime  time2,         // Second time coordinate  
    double    price2,        // Second price coordinate  
    datetime  time3,         // Third time coordinate  
    double    price3         // Third price coordinate  
)
```

Parameters

chart_id

[in] Chart identifier (0 - current chart) .

name

[in] An unique name of the object to create.

window

[in] Chart window number (0 - base window) .

time1

[in] Time coordinate of the first anchor point.

price1

[in] Price coordinate of the first anchor point.

time2

[in] Time coordinate of the second anchor point.

price2

[in] Price coordinate of the second anchor point.

time3

[in] Time coordinate of the third anchor point.

price3

[in] Price coordinate of the third anchor point.

Returned value

true if successful, false if error

Type

Returns object type identifier of the graphic object.

```
int Type() const
```

Returned value

Object type identifier (OBJ__TRIANGLE for CChartObjectTriangle) .

CChartObjectEllipse

Class CChartObjectEllipse is a class for simplified access to "Ellipse" graphic object properties.

Description

Class CChartObjectEllipse provides access to "Ellipse" object properties.

Declaration

```
class CChartObjectEllipse : public CChartObject
```

Title

```
#include <ChartObjects\ChartObjectsShapes.mqh>
```

Class Methods

Create	
Create	Creates graphic object "Ellipse"
Input/output	
virtual Type	Virtual method of identification

See also

[Object types](#), [Graphic objects](#)

Create

Creates graphic object "Ellipse".

```
bool Create(  
    long      chart_id,      // Chart identifier  
    string     name,         // Object name  
    int       window,        // Chart window  
    datetime   time1,        // First time coordinate  
    double     price1,       // First price coordinate  
    datetime   time2,        // Second time coordinate  
    double     price2,       // Second price coordinate  
    datetime   time3,        // Third time coordinate  
    double     price3        // Third price coordinate  
)
```

Parameters

chart_id

[in] Chart identifier (0 - current chart) .

name

[in] An unique name of the object to create.

window

[in] Chart window number (0 - base window) .

time1

[in] Time coordinate of the first anchor point.

price1

[in] Price coordinate of the first anchor point.

time2

[in] Time coordinate of the second anchor point.

price2

[in] Price coordinate of the second anchor point.

time3

[in] Time coordinate of the third anchor point.

price3

[in] Price coordinate of the third anchor point.

Returned value

true if successful, false if error

Type

Returns object type identifier of the graphic object.

```
int Type() const
```

Returned value

Object type identifier (OBJ__ELLIPSE for CChartObjectEllipse) .

Objects Arrows

Group for graphic objects Arrows.

This section contains the technical details of working with a group of classes of graphical objects "Arrow" and a description of the relevant components of the MQL5 Standard Library . In essence, the arrow - this is some icon to be displayed to the user, which matches a certain code. There are two types of graphical objects "Arrow" to display icons on the charts:

- Object "Arrow", which allows you to specify the code icon displayed object.
- Group objects to display certain types of icons (and the corresponding certain fixed code) .

Class for the arrow displays icons of arbitrary code

Class name	Name of the object arrow
CChartObjectArrow	Arrow

Classes for the arrow icon fixed code

Class name	Name of the object arrow
CChartObjectArrowCheck	Check
CChartObjectArrowDown	Arrow Up
CChartObjectArrowUp	Arrow Down
CChartObjectArrowStop	Stop Sign
CChartObjectArrowThumbDown	Thumbs Up
CChartObjectArrowThumbUp	Thumbs Down
CChartObjectArrowLeftPrice	Left Price Label
CChartObjectArrowRightPrice	Right Price Label

See also

[Object types](#), [Methods of binding sites](#), [Graphic objects](#)

CChartObjectArrow

Class CChartObjectArrow is a class for simplified access to "Arrow" graphic object properties.

Description

Class CChartObjectArrow provides access to common properties of objects "Arrow" to all of its descendants.

Declaration

```
class CChartObjectArrow : public CChartObject
```

Title

```
#include <ChartObjects\ChartObjectsArrow.mqh>
```

Class Methods

Create	
Create	Creates graphic object "Arrow"
Properties	
ArrowCode	Gets/Sets property "Arrow Code"
Anchor	Gets/Sets property "Anchor"
Input/output	
virtual Save	Virtual method for writing to file
virtual Load	Virtual method for reading from file
virtual Type	Virtual method of identification

See also

[Object types](#), [Methods of binding sites](#), [Graphic objects](#)

Create

Creates graphic object "Arrow".

```
bool Create(
    long      chart_id,      // Chart ID
    string     name,         // Object Name
    int       window,        // Chart Window
    datetime  time,          // Time
    double     price,        // Price
    char      code           // Arrow Code
)
```

Parameters

chart_id

[in] Chart Identifier (0 - current chart) .

name

[in] Object name (Should be unique) .

window

[in] Chart window number (0 - base window) .

time

[in] Time coordinate.

price

[in] Price coordinate.

code

[in] "Arrow" code (Wingdings) .

Returned value

true - if successful, otherwise false.

Example:

```
//--- example for CChartObjectArrow::Create
#include <ChartObjects\ChartObjectsArrows.mqh>
//---
void OnStart()
{
    CChartObjectArrow arrow;
    //--- set object parameters
    double price=SymbolInfoDouble(Symbol(),SYMBOL_BID);
    if(!arrow.Create(0,"Arrow",0,TimeCurrent(),price,181))
    {
        //--- arrow create error
        printf("Arrow create: Error %d!",GetLastError());
    }
    //---
}
```



```
        return;  
    }  
    //--- use arrow  
    //--- . . .  
}
```

ArrowCode (Get Method)

Gets code of the symbol for "Arrow".

```
char ArrowCode() const
```

Returned value

Symbol code of "Arrow" object, assigned to the class instance. If there is no object assigned, it returns 0.

ArrowCode (Set Method)

Sets symbol code for "Arrow"

```
bool ArrowCode (
    char code      // Code value
)
```

Parameters

code

[in] new value for "arrow" code (Wingdings) .

Returned value

true - if successful, false - if code hasn't changed.

Example:

```
//--- example for CChartObjectArrow::ArrowCode
#include <ChartObjects\ChartObjectsArrows.mqh>
//---
void OnStart()
{
    CChartObjectArrow arrow;
    char code=181;
    //--- set object parameters
    double price=SymbolInfoDouble(Symbol(),SYMBOL_BID);
    if(!arrow.Create(0,"Arrow",0,TimeCurrent(),price,code))
    {
        //--- arrow create error
        printf("Arrow create: Error %d!",GetLastError());
        //---
        return;
    }
    //--- use arrow to possible changes code
    //--- . . .
    //--- get code of arrow
    if(arrow.ArrowCode()!=code)
    {
        //--- set code of arrow
```

```
        arrow.ArrowCode(code);  
    }  
    //--- use arrow  
    //--- . . .  
}
```

Anchor (Get Method)

Gets anchor type of the "Anchor" object

```
ENUM_ARROW_ANCHOR Anchor() const
```

Returned value

Anchor type of "Arrow" object, assigned to the class instance. If there is no object assigned, it returns WRONG_VALUE.

Anchor (Set Method)

Sets color for graphic object

```
bool Anchor(
    ENUM_ARROW_ANCHOR new_color    // new color
)
```

Parameters

new_color

[in] New value of color for line of the graphic object.

Returned value

true if successful, false if color hasn't changed.

Example:

```
//--- example for CChartObject::Anchor
#include <ChartObjects\ChartObjectsArrows.mqh>
//---
void OnStart()
{
    CChartObjectArrow arrow;
    ENUM_ARROW_ANCHOR anchor=ANCHOR_BOTTOM;
    //--- set object parameters
    double price=SymbolInfoDouble(Symbol(),SYMBOL_BID);
    if(!arrow.Create(0,"Arrow",0,TimeCurrent(),price,181))
    {
        //--- arrow create error
        printf("Arrow create: Error %d!",GetLastError());
        //---
        return;
    }
    //--- get anchor of arrow
    if(arrow.Anchor()!=anchor)
    {
        //--- set anchor of arrow
        arrow.Anchor(anchor);
    }
}
```

```
//--- use arrow  
//--- . . .  
}
```

Save

Saves object parameters to file.

```
virtual bool Save(
    int file_handle    // file handle
)
```

Parameters

file_handle

[in] handle of the file already opened using by function FileOpen(...) .

Returned value

true if successful, otherwise false.

Example:

```
//--- example for CChartObjectArrow::Save
#include <ChartObjects\ChartObjectsArrows.mqh>
//---
void OnStart()
{
    int file_handle;
    CChartObjectArrow arrow;
    //--- set object parameters
    double price=SymbolInfoDouble(Symbol(),SYMBOL_BID);
    if(!arrow.Create(0,"Arrow",0,TimeCurrent(),price,181))
    {
        //--- arrow create error
        printf("Arrow create: Error %d!",GetLastError());
        //---
        return;
    }
    //--- open file
    file_handle=FileOpen("MyFile.bin",FILE_WRITE|FILE_BIN|FILE_ANSI);
    if(file_handle>=0)
    {
        if(!arrow.Save(file_handle))
        {
            //--- file save error
            printf("File save: Error %d!",GetLastError());
            FileClose(file_handle);
            //---
            return;
        }
        FileClose(file_handle);
    }
}
```

Load

Loads object parameters from file.

```
virtual bool Load(
    int file_handle    // file handle
)
```

Parameters

file_handle

[in] handle of the file already opened using by function FileOpen(...) .

Returned value

true if successful, otherwise false.

Example:

```
//--- example for CChartObjectArrow::Load
#include <ChartObjects\ChartObjectsArrows.mqh>
//---
void OnStart()
{
    int file_handle;
    CChartObjectArrow arrow;
    //--- open file
    file_handle=FileOpen("MyFile.bin", FILE_READ|FILE_BIN|FILE_ANSI);
    if(file_handle>=0)
    {
        if(!arrow.Load(file_handle))
        {
            //--- file load error
            printf("File load: Error %d!", GetLastError());
            FileClose(file_handle);
            //---
            return;
        }
        FileClose(file_handle);
    }
    //--- use arrow
    //--- . . .
}
```

Type

Returns graphic object type identifier

```
virtual int Type() const
```

Returned value

Object type identifier (for example OBJ__ARROW for CChartObjectArrow)

Example:

```
//--- example for CChartObjectArrow::Type
#include <ChartObjects\ChartObjectsArrows.mqh>
//---
void OnStart()
{
    CChartObjectArrow arrow;
    //--- get arrow type
    int type=arrow.Type();
}
```


Arrows with fixed code

Classes "Arrows with fixed code" are classes for simplified access to the properties of the following graphic objects:

Class name	Arrow object name
CChartObjectArrowCheck	"Arrow Check"
CChartObjectArrowDown	"Arrow Down"
CChartObjectArrowUp	"Arrow Up"
CChartObjectArrowStop	"Arrow Stop"
CChartObjectArrowThumbDown	"Good" ("Big finger up")
CChartObjectArrowThumbUp	"Bad" ("Big finger down")
CChartObjectArrowLeftPrice	"Left price" arrow
CChartObjectArrowRightPrice	"Right price" arrow

Description

Classes "Arrows with fixed code" provides access to the object properties.

Declarations

```

class CChartObjectArrowCheck      : public CChartObjectArrow;
class CChartObjectArrowDown      : public CChartObjectArrow;
class CChartObjectArrowUp        : public CChartObjectArrow;
class CChartObjectArrowStop      : public CChartObjectArrow;
class CChartObjectArrowThumbDown : public CChartObjectArrow;
class CChartObjectArrowThumbUp   : public CChartObjectArrow;
class CChartObjectArrowLeftPrice : public CChartObjectArrow;
class CChartObjectArrowRightPrice: public CChartObjectArrow;
```

Title

```
<ChartObjects\ChartObjectsArrow.mqh>
```

Class Methods

Create	
<u>Create</u>	Creates graphic object specified
Properties	
<u>ArrowCode</u>	"Gag" for method of code change
Input/output	
virtual <u>Type</u>	Virtual method of identification

See also

[Object types](#), [Methods of binding sites](#), [Graphic objects](#)

Create

Creates graphic object "Arrow with fixed code".

```
bool Create(
    long      chart_id,      // Chart ID
    string     name,         // Object Name
    int       window,        // Chart Window
    datetime  time,          // Time
    double     price         // Price
)
```

Parameters

chart_id

[in] Chart identifier (0 - current chart) .

name

[in] Unique name of the object to create.

window

[in] Chart window number (0 - base window) .

time

[in] Time coordinate.

price

[in] Price coordinate.

Returned value

true if successful, false if error.

Example:

```
//--- example for CChartObjectArrowCheck::Create
//--- example for CChartObjectArrowDown::Create
//--- example for CChartObjectArrowUp::Create
//--- example for CChartObjectArrowStop::Create
//--- example for CChartObjectArrowThumbDown::Create
//--- example for CChartObjectArrowThumbUp::Create
//--- example for CChartObjectArrowLeftPrice::Create
//--- example for CChartObjectArrowRightPrice::Create
#include <ChartObjects\ChartObjectsArrows.mqh>
//---
void OnStart()
{
    //--- for example, take CChartObjectArrowCheck
    CChartObjectArrowCheck arrow;
    //--- set object parameters
    double price=SymbolInfoDouble(Symbol(),SYMBOL_BID);
    if(!arrow.Create(0,"ArrowCheck",0,TimeCurrent(),price))
```

```
{
    //--- arrow create error
    printf("Arrow create: Error %d!", GetLastError());
    //---
    return;
}
//--- use arrow
//--- . . .
}
```

ArrowCode

Prohibits code changes for "Arrow".

```
bool ArrowCode(  
    char code    // code value  
)
```

Parameters

code

[in] any value

Returned value

Always false.

Example:

```
//--- example for CChartObjectArrowCheck::ArrowCode  
//--- example for CChartObjectArrowDown::ArrowCode  
//--- example for CChartObjectArrowUp::ArrowCode  
//--- example for CChartObjectArrowStop::ArrowCode  
//--- example for CChartObjectArrowThumbDown::ArrowCode  
//--- example for CChartObjectArrowThumbUp::ArrowCode  
//--- example for CChartObjectArrowLeftPrice::ArrowCode  
//--- example for CChartObjectArrowRightPrice::ArrowCode  
#include <ChartObjects\ChartObjectsArrows.mqh>  
//---  
void OnStart()  
{  
    //--- for example, take CChartObjectArrowCheck  
    CChartObjectArrowCheck arrow;  
    //--- set object parameters  
    double price=SymbolInfoDouble(Symbol(),SYMBOL_BID);  
    if(!arrow.Create(0,"ArrowCheck",0,TimeCurrent(),price))  
    {  
        //--- arrow create error  
        printf("Arrow create: Error %d!",GetLastError());  
        //---  
        return;  
    }  
    //--- set code of arrow  
    if(!arrow.ArrowCode(181))  
    {  
        //--- it is not error  
        printf("Arrow code can not be changed");  
    }  
    //--- use arrow  
    //--- . . .  
}
```

Type

Returns graphic object type identifier

```
virtual int Type() const
```

Returned value

Object type identifier (OBJ_ARROW_CHECK for CChartObjectArrowCheck, OBJ_ARROW_DOWN for CChartObjectArrowDown, OBJ_ARROW_UP for CChartObjectArrowUp, OBJ_ARROW_STOP for CChartObjectArrowStop, OBJ_ARROW_THUMB_DOWN for CChartObjectArrowThumbDown, OBJ_ARROW_THUMB_UP for CChartObjectArrowThumbUp, OBJ_ARROW_LEFT_PRICE for CChartObjectArrowLeftPrice, OBJ_ARROW_RIGHT_PRICE for CChartObjectArrowRightPrice) .

Example:

```
//--- example for CChartObjectArrowCheck::Type
//--- example for CChartObjectArrowDown::Type
//--- example for CChartObjectArrowUp::Type
//--- example for CChartObjectArrowStop::Type
//--- example for CChartObjectArrowThumbDown::Type
//--- example for CChartObjectArrowThumbUp::Type
//--- example for CChartObjectArrowLeftPrice::Type
//--- example for CChartObjectArrowRightPrice::Type
#include <ChartObjects\ChartObjectsArrows.mqh>
//---
void OnStart ()
{
//--- for example, take CChartObjectArrowCheck
    CChartObjectArrowCheck arrow;
//--- get arrow type
    int type=arrow.Type();
}
```

Object Controls

A group of graphic objects "Object Controls".

This section contains the technical details of working with a group of classes of graphical objects "Object Controls" and a description of the relevant components of the MQL5 Standard Library .

Class name	Object
CChartObjectText	Graphic object "Text"
CChartObjectLabel	Graphic object "Text Label"
CChartObjectEdit	Graphic object "Edit"
CChartObjectButton	Graphic object "Button"
CChartObjectSubChart	Graphic object "Chart"
CChartObjectBitmap	Graphic object "Bitmap"
CChartObjectBmpLabel	Graphic object "Bitmap Label"
CChartObjectRectLabel	Graphic object "Rectangle Label"

See also

[Object types](#), [Graphic objects](#)

CChartObjectText

Class CChartObjectText is a class for simplified access to "Text" graphic object properties.

Description

Class CChartObjectText provides access to "Text" object properties.

Declaration

```
class CChartObjectText : public CChartObject
```

Title

```
#include <ChartObjects\ChartObjectsTxtControls.mqh>
```

Class Methods

Create	
Create	Creates graphic object "Text"
Properties	
Angle	Gets/Sets property "Angle"
Font	Gets/Sets property "Font"
FontSize	Gets/Sets property "FontSize"
Anchor	Gets/Sets property "Anchor"
Input/output	
virtual Save	Virtual method for writing to file
virtual Load	Virtual method for reading from file
virtual Type	Virtual method of identification

Derived classes:

- [CChartObjectLabel](#)

See also

[Object types](#), [Object properties](#), [Methods of binding sites](#), [Graphic objects](#)

Create

Creates graphic object "Text".

```
bool Create(  
    long      chart_id,      // Chart identifier  
    string    name,          // Object name  
    int       window,        // Chart window  
    datetime  time,          // Time coordinate  
    double    price          // Price coordinate  
)
```

Parameters

chart_id

[in] Chart identifier (0 - current chart) .

name

[in] An unique name of the object to create.

window

[in] Chart window number (0 - base window) .

time

[in] Time coordinate of the anchor point.

price

[in] Price coordinate of the anchor point.

Returned value

true if successful, false if error.

Angle (Get Method)

Gets the value of "Angle" property.

```
double Angle() const
```

Returned value

Value of "Angle" property of the object, assigned to the class instance. If there is no object assigned, it returns EMPTY_VALUE.

Angle (Set Method)

Sets new value for "Angle" property.

```
bool Angle(  
    double angle    // new angle  
)
```

Parameters

angle

[in] New value for "Angle" property.

Returned value

true if successful, false if property hasn't changed.

Font (Get Method)

Gets the value of "Font" property.

```
string Font() const
```

Returned value

Value of "Font" property of the object, assigned to the class instance. If there is no object assigned, it returns "".

Font (Set Method)

Sets new value for "Font" property.

```
bool Font(  
    string font    // new font  
)
```

Parameters

font

[in] New value for "Font" property.

Returned value

true if successful, false if property hasn't changed.

FontSize (Get Method)

Gets the value of "FontSize" property.

```
int FontSize() const
```

Returned value

Value of "FontSize" property of the object, assigned to the class instance. If there is no object assigned, it returns 0.

FontSize (Set Method)

Sets new value for "FontSize" property.

```
bool FontSize(  
    int size    // new font size  
)
```

Parameters

size

[in] New value for "Font" property.

Returned value

true if successful, false if property hasn't changed.

Anchor (Get Method)

Gets the value of "Anchor" property.

```
ENUM_ANCHOR_POINT Anchor() const
```

Returned value

Value of "Anchor" property of the object, assigned to the class instance. If there is no object assigned, it returns WRONG_VALUE.

Anchor (Set Method)

Sets new value for "Anchor" property.

```
bool Anchor(  
    ENUM_ANCHOR_POINT anchor    // new value  
)
```

Parameters

anchor

[in] New value for "Anchor" property.

Returned value

true if successful, false if property hasn't changed.

Save

Saves object parameters to file.

```
virtual bool Save(  
    int file_handle    // File handle  
)
```

Parameters

file_handle

[in] handle of the binary file already opened by [FileOpen](#) function.

Returned value

true if successful, false if error.

Load

Loads object parameters from file.

```
virtual bool Load(  
    int file_handle    // File handle  
)
```

Parameters

file_handle

[in] handle of the binary file already opened by [FileOpen](#) function.

Returned value

true if successful, false if error.

Type

Returns graphic object type identifier.

```
virtual int Type() const
```

Returned value

Object type identifier (OBJ__TEXT for [CChartObjectText](#)) .

CChartObjectLabel

Class CChartObjectLabel is a class for simplified access to "Label" graphic object properties.

Description

Class CChartObjectLabel provides access to "Label" object properties.

Declaration

```
class CChartObjectLabel : public CChartObjectText
```

Title

```
#include <ChartObjects\ChartObjectsTxtControls.mqh>
```

Class Methods

Create	
Create	Creates graphic object "Label"
Properties	
X_Distance	Gets/Sets property "X_Distance"
Y_Distance	Gets/Sets property "Y_Distance"
X_Size	Gets/Sets property "X_Size"
Y_Size	Gets/Sets property "Y_Size"
Corner	Gets/Sets property "Corner"
Time	"Gag" for Time Coordinate change
Price	"Gag" for Price Coordinate change
Input/output	
virtual Save	Virtual method for writing to file
virtual Load	Virtual method for reading from file
virtual Type	Virtual method of identification

Derived classes:

- [CChartObjectEdit](#)

See also

[Object types](#), [Object properties](#), [Chart angle](#), [Methods of binding sites](#), [Graphic objects](#)

Create

Creates graphic object "Label".

```
bool Create(  
    long    chart_id,    // Chart identifier  
    string  name,        // Object name  
    int     window,      // Chart window  
    int     X,           // X coordinate  
    int     Y,           // Y coordinate  
    int     sizeX,       // X size  
    int     sizeY        // Y size  
)
```

Parameters

chart_id

[in] Chart identifier (0 - current chart) .

name

[in] An unique name of the object to create.

window

[in] Chart window number (0 - base window) .

X

[in] X coordinate.

Y

[in] Y coordinate.

sizeX

[in] X size.

sizeY

[in] Y size.

Returned value

true if successful, false if error.

X__Distance (Get Method)

Gets the value of "X__Distance" property.

```
int X__Distance() const
```

Returned value

Value of "X__Distance" property of the object, assigned to the class instance. If there is no object assigned, it returns 0.

X__Distance (Set Method)

Sets new value for "X__Distance" property.

```
bool X__Distance(  
    int x        // new value  
)
```

Parameters

x

[in] New value for "X__Distance" property.

Returned value

true if successful, false if property hasn't changed.

Y__Distance (Get Method)

Gets the value of "Y__Distance" property.

```
int Y__Distance() const
```

Returned value

Value of "Y__Distance" property of the object, assigned to the class instance. If there is no object assigned, it returns 0.

Y__Distance (Set Method)

Sets new value for "Y__Distance" property.

```
bool Y__Distance(  
    int Y        // new value  
)
```

Parameters

Y

[in] New value for "Y__Distance" property.

Returned value

true if successful, false if property hasn't changed.

X__Size

Gets the value of "X__Size" property.

```
int X__Size() const
```

Returned value

Value of "X__Size" property of the object, assigned to the class instance. If there is no object assigned, it returns 0.

Y_Size

Gets the value of "Y_Size" property.

```
int Y_Size() const
```

Returned value

Value of "Y_Size" property of the object, assigned to the class instance. If there is no object assigned, it returns 0.

Corner (Get Method)

Gets the value of "Corner" property.

```
ENUM_BASE_CORNER Corner() const
```

Returned value

Value of "Corner" property of the object, assigned to the class instance. If there is no object assigned, it returns WRONG_VALUE.

Corner (Set Method)

Sets new value for "Corner" property.

```
bool Corner(  
    ENUM_BASE_CORNER corner    // new value  
)
```

Parameters

corner

[in] New value for "Corner" property.

Returned value

true if successful, false if property hasn't changed.

Time

Prohibits changes of the time coordinate.

```
bool Time(  
    datetime time    // any value  
)
```

Parameters

time

[in] Any value of datetime type.

Returned value

always false.

Price

Prohibits changes of the price coordinate.

```
bool Price(  
    double price    // any value  
)
```

Parameters

price

[in] Any value of double type.

Returned value

always false.

Save

Saves object parameters to file.

```
virtual bool Save(  
    int file_handle    // File handle  
)
```

Parameters

file_handle

[in] handle of the binary file already opened by [FileOpen](#) function.

Returned value

true if successful, false if error.

Load

Loads object parameters from file.

```
virtual bool Load(  
    int file_handle    // File handle  
)
```

Parameters

file_handle

[in] handle of the binary file already opened by [FileOpen](#) function.

Returned value

true if successful, false if error.

Type

Returns graphic object type identifier.

```
virtual int Type() const
```

Returned value

Object type identifier (OBJ__LABEL for [CChartObjectLabel](#)) .

CChartObjectEdit

Class CChartObjectEdit is a class for simplified access to "Edit" graphic object properties.

Description

Class CChartObjectEdit provides access to "Edit" object properties.

Declaration

```
class CChartObjectEdit : public CChartObjectLabel
```

Title

```
#include <ChartObjects\ChartObjectsTxtControls.mqh>
```

Class Methods

Create	
Create	Creates graphic object "Edit"
Properties	
X_Size	Gets property "X Size"
Y_Size	Gets property "Y Size"
BackColor	Gets/Sets property "Background Color"
Angle	Gets/Sets property "Angle"
Input/output	
virtual Save	Virtual method for writing to file
virtual Load	Virtual method for reading from file
virtual Type	Virtual method of identification

Derived classes:

- [CChartObjectButton](#)

See also

[Object types](#), [Object properties](#), [Chart angle](#), [Methods of binding sites](#), [Graphic objects](#)

Create

Creates graphic object "Edit".

```
bool Create(  
    long    chart_id,    // Chart identifier  
    string  name,        // Object name  
    int     window,     // Chart window  
    int     X,           // X coordinate  
    int     Y,           // Y coordinate  
    int     sizeX,       // X size  
    int     sizeY        // Y size  
)
```

Parameters

chart_id

[in] Chart identifier (0 - current chart) .

name

[in] An unique name of the object to create.

window

[in] Chart window number (0 - base window) .

X

[in] X coordinate.

Y

[in] Y coordinate.

sizeX

[in] X size.

sizeY

[in] Y size.

Returned value

true if successful, false if error.

X__Size

Sets new value for "X__Size" property.

```
bool X_Size(  
    int size    // new size  
)
```

Parameters

size

[in] New value for "X__Size" property.

Returned value

true if successful, false if property hasn't changed.

Y_Size

Sets new value for "Y_Size" property.

```
bool Y_Size(  
    int size    // new size  
)
```

Parameters

size

[in] New value for "Y_Size" property.

Returned value

true if successful, false if property hasn't changed.

BackColor (Get Method)

Gets the value of "BackColor" property.

```
color BackColor() const
```

Returned value

Value of "BackColor" property of the object, assigned to the class instance. If there is no object assigned, it returns CLR_NONE.

BackColor (Set Method)

Sets new value for "BackColor" property.

```
bool BackColor (
    color new_color      // new background color
)
```

Parameters

new_color

[in] New value for "BackColor" property.

Returned value

true if successful, false if property hasn't changed.

Angle

Prohibits changes of the "Angle" property.

```
bool Angle(  
    double angle    // any value  
)
```

Parameters

angle

[in] Any value of double type.

Returned value

always false.

Save

Saves object parameters to file.

```
virtual bool Save(  
    int file_handle    // File handle  
)
```

Parameters

file_handle

[in] handle of the binary file already opened by [FileOpen](#) function.

Returned value

true if successful, false if error.

Load

Loads object parameters from file.

```
virtual bool Load(  
    int file_handle    // File handle  
)
```

Parameters

file_handle

[in] handle of the binary file already opened by [FileOpen](#) function.

Returned value

true if successful, false if error.

Type

Returns graphic object type identifier.

```
virtual int Type() const
```

Returned value

Object type identifier (OBJ__EDIT for [CChartObjectEdit](#)) .

CChartObjectButton

Class CChartObjectButton is a class for simplified access to "Button" graphic object properties.

Description

Class CChartObjectButton provides access to "Button" object properties.

Declaration

```
class CChartObjectButton : public CChartObjectEdit
```

Title

```
#include <ChartObjects\ChartObjectsTxtControls.mqh>
```

Class Methods

Create	
Create	Inherited form class CChartObjectEdit
Properties	
State	Gets/Sets button state (Pressed/Depressed)
Input/output	
virtual Save	Virtual method for writing to file
virtual Load	Virtual method for reading from file
virtual Type	Virtual method of identification

See also

[Object types](#), [Object properties](#), [Chart angle](#), [Methods of binding sites](#), [Graphic objects](#)

State (Get Method)

Gets the value of "State" property.

```
bool State() const
```

Returned value

Value of "State" property of the object, assigned to the class instance. If there is no object assigned, it returns false.

State (Set Method)

Sets new value for "State" property.

```
bool State(  
    bool state    // new state value  
)
```

Parameters

x

[in] New value for "State" property.

Returned value

true if successful, false if property hasn't changed.

Save

Saves object parameters to file.

```
virtual bool Save(  
    int file_handle    // File handle  
)
```

Parameters

file_handle

[in] handle of the binary file already opened by [FileOpen](#) function.

Returned value

true if successful, false if error.

Load

Loads object parameters from file.

```
virtual bool Load(  
    int file_handle    // File handle  
)
```

Parameters

file_handle

[in] handle of the binary file already opened by [FileOpen](#) function.

Returned value

true if successful, false if error.

Type

Returns graphic object type identifier.

```
virtual int Type() const
```

Returned value

Object type identifier (OBJ__BUTTON for [CChartObjectButton](#)) .

CChartObjectSubChart

Class CChartObjectSubChart is a class for simplified access to "Chart" graphic object properties.

Description

Class CChartObjectSubChart provides access to "Chart" object properties.

Declaration

```
class CChartObjectSubChart : public CChartObject
```

Title

```
#include <ChartObjects\ChartObjectsSubChart.mqh>
```

Class Methods

Create	
Create	Creates graphic object "Chart"
Properties	
X_Distance	Gets/Sets property "X_Distance"
Y_Distance	Gets/Sets property "Y_Distance"
Corner	Gets/Sets property "Corner"
X_Size	Gets/Sets property "X_Size"
Y_Size	Gets/Sets property "Y_Size"
Symbol	Gets/Sets property "Symbol"
Period	Gets/Sets property "Period"
Scale	Gets/Sets property "Scale"
DateScale	Gets/Sets property "Show date scale"
PriceScale	Gets/Sets property "Show price scale"
Time	"Gag" for time coordinate change
Price	"Gag" for price coordinate change
Input/output	
virtual Save	Virtual method for writing to file
virtual Load	Virtual method for reading from file
virtual Type	Virtual method of identification

See also

[Object types](#), [Object properties](#), [Chart angle](#), [Graphic objects](#)

Create

Creates graphic object "SubChart".

```
bool Create(  
    long    chart_id,    // Chart identifier  
    string  name,        // Object name  
    int     window,      // Chart window  
    int     X,           // X coordinate  
    int     Y,           // Y coordinate  
    int     sizeX,       // X size  
    int     sizeY        // Y size  
)
```

Parameters

chart_id

[in] Chart identifier (0 - current chart) .

name

[in] An unique name of the object to create.

window

[in] Chart window number (0 - base window) .

X

[in] X coordinate.

Y

[in] Y coordinate.

sizeX

[in] X size.

sizeY

[in] Y size.

Returned value

true if successful, false if error.

X__Distance (Get Method)

Gets the value of "X__Distance" property.

```
int X__Distance() const
```

Returned value

Value of "X__Distance" property of the object, assigned to the class instance. If there is no object assigned, it returns 0.

X__Distance (Set Method)

Sets new value for "X__Distance" property.

```
bool X__Distance(  
    int X        // new value  
)
```

Parameters

X

[in] New value for "X__Distance" property.

Returned value

true if successful, false if property hasn't changed.

Y__Distance (Get Method)

Gets the value of "Y__Distance" property.

```
int Y__Distance() const
```

Returned value

Value of "Y__Distance" property of the object, assigned to the class instance. If there is no object assigned, it returns 0.

Y__Distance (Set Method)

Sets new value for "Y__Distance" property.

```
bool Y__Distance(  
    int Y        // new value  
)
```

Parameters

Y

[in] New value for "Y__Distance" property.

Returned value

true if successful, false if property hasn't changed.

Corner (Get Method)

Gets the value of "Corner" property.

```
ENUM_BASE_CORNER Corner() const
```

Returned value

Value of "Corner" property of the object, assigned to the class instance. If there is no object assigned, it returns WRONG_VALUE.

Corner (Set Method)

Sets new value for "Corner" property.

```
bool Corner(  
    ENUM_BASE_CORNER corner    // new value  
)
```

Parameters

corner

[in] New value for "Corner" property.

Returned value

true if successful, false if property hasn't changed.

X__Size (Get Method)

Gets the value of "X__Size" property.

```
int X__Size() const
```

Returned value

Value of "X__Size" property of the object, assigned to the class instance. If there is no object assigned, it returns 0.

X__Size (Set Method)

Sets new value for "X__Size" property.

```
bool X__Size(  
    int X    // new value  
)
```

Parameters

X

[in] New value for "X__Size" property.

Returned value

true if successful, false if property hasn't changed.

Y__Size (Get Method)

Gets the value of "Y__Size" property.

```
int Y__Size() const
```

Returned value

Value of "Y__Size" property of the object, assigned to the class instance. If there is no object assigned, it returns 0.

Y__Size (Set Method)

Sets new value for "Y__Size" property.

```
bool Y__Size(  
    int Y        // new value  
)
```

Parameters

Y

[in] New value for "Y__Size" property.

Returned value

true if successful, false if property hasn't changed.

Symbol (Get Method)

Gets the value of "Symbol" property.

```
string Symbol() const
```

Returned value

Value of "Symbol" property of the object, assigned to the class instance. If there is no object assigned, it returns "".

Symbol (Set Method)

Sets new value for "Symbol" property.

```
bool Symbol(  
    string symbol    // new symbol  
)
```

Parameters

symbol

[in] New value for "Symbol" property.

Returned value

true if successful, false if property hasn't changed.

Period (Get Method)

Gets the value of "Period" property.

```
int Period() const
```

Returned value

Value of "Period" property of the object, assigned to the class instance. If there is no object assigned, it returns 0.

Period (Set Method)

Sets new value for "Period" property.

```
bool Period(  
    int period    // new period  
)
```

Parameters

period

[in] New value for "Period" property.

Returned value

true if successful, false if property hasn't changed.

Scale (Get Method)

Gets the value of "Scale" property.

```
double Scale() const
```

Returned value

Value of "Scale" property of the object, assigned to the class instance. If there is no object assigned, it returns EMPTY_VALUE.

Scale (Set Method)

Sets new value for "Scale" property.

```
bool Scale(  
    double scale    // new scale  
)
```

Parameters

scale

[in] New value for "Scale" property.

Returned value

true if successful, false if property hasn't changed.

DateScale (Get Method)

Gets the value of "DateScale" property.

```
bool DateScale() const
```

Returned value

Value of "DateScale" property of the object, assigned to the class instance. If there is no object assigned, it returns false.

DateScale (Set Method)

Sets new value for "DateScale" property.

```
bool DateScale (  
    bool scale    // new value  
)
```

Parameters

scale

[in] New value for "DateScale" property.

Returned value

true if successful, false if property hasn't changed.

PriceScale (Get Method)

Gets the value of "PriceScale" property.

```
bool PriceScale() const
```

Returned value

Value of "PriceScale" property of the object, assigned to the class instance. If there is no object assigned, it returns false.

PriceScale (Set Method)

Sets new value for "PriceScale" property.

```
bool PriceScale(  
    bool scale    // new value  
)
```

Parameters

scale

[in] New value for "PriceScale" property.

Returned value

true if successful, false if property hasn't changed.

Time

Prohibits changes of the time coordinate.

```
bool Time(  
    datetime time    // any value  
)
```

Parameters

time

[in] Any value of datetime type.

Returned value

always false.

Price

Prohibits changes of the price coordinate.

```
bool Price(  
    double price    // any value  
)
```

Parameters

price

[in] Any value of double type.

Returned value

always false.

Save

Saves object parameters to file.

```
virtual bool Save(  
    int file_handle    // File handle  
)
```

Parameters

file_handle

[in] handle of the binary file already opened by [FileOpen](#) function.

Returned value

true if successful, false if error.

Load

Loads object parameters from file.

```
virtual bool Load(  
    int file_handle    // File handle  
)
```

Parameters

file_handle

[in] handle of the binary file already opened by [FileOpen](#) function.

Returned value

true if successful, false if error.

Type

Returns graphic object type identifier.

```
virtual int Type() const
```

Returned value

Object type identifier (OBJ__CHART for [CChartObjectSubChart](#)).

CChartObjectBitmap

Class CChartObjectBitmap is a class for simplified access to "Bitmap" graphic object properties.

Description

Class CChartObjectBitmap provides access to "Bitmap" object properties.

Declaration

```
class CChartObjectBitmap : public CChartObject
```

Title

```
#include <ChartObjects\ChartObjectsBmpControls.mqh>
```

Class Methods

Create	
Create	Creates graphic object "Bitmap"
Properties	
BmpFile	Gets/Sets property "BMP Filename"
X__Offset	Gets/Sets property "X__Offset"
Y__Offset	Gets/Sets property "Y__Offset"
Input/output	
virtual Save	Virtual method for writing to file
virtual Load	Virtual method for reading from file
virtual Type	Virtual method of identification

See also

[Object types](#), [Object properties](#), [Graphic objects](#)

Create

Creates graphic object "Bitmap".

```
bool Create(  
    long      chart_id,      // Chart identifier  
    string    name,          // Object name  
    int       window,        // Chart window  
    datetime  time,          // Time coordinate  
    double    price          // Price coordinate  
)
```

Parameters

chart_id

[in] Chart identifier (0 - current chart) .

name

[in] An unique name of the object to create.

window

[in] Chart window number (0 - base window) .

time

[in] Time coordinate of the anchor point.

price

[in] Price coordinate of the anchor point.

Returned value

true if successful, false if error.

BmpFile (Get Method)

Gets the value of "BmpFile" property.

```
string BmpFile() const
```

Returned value

Value of "BmpFile" property of the object, assigned to the class instance. If there is no object assigned, it returns false.

BmpFile (Set Method)

Sets new value for "BmpFile" property.

```
bool BmpFile(  
    string name    // new file name  
)
```

Parameters

x

[in] New value for "BmpFile" property.

Returned value

true if successful, false if property hasn't changed.

X__Offset (Get Method)

Gets the value of "X__Offset" property (the X coordinate of the upper left corner of the rectangular visible area in the graphical objects) .

```
int X__Offset () const
```

Returned value

Value of "X__Offset" property of the object, assigned to the class instance. If there is no object assigned, it returns 0.

X__Offset (Set Method)

Sets new value for "X__Offset" property (the X coordinate of the upper left corner of the rectangular visible area in the graphical objects) . The value is set in pixels relative to the upper left corner of the original image.

```
bool X__Offset (
    int X          // new value
)
```

Parameters

X

[in] New value for "X__Offset" property.

Returned value

true if successful, false if property hasn't changed.

Y__Offset (Get Method)

Gets the value of "Y__Offset" property (the Y coordinate of the upper left corner of the rectangular visible area in the graphical objects) .

```
int Y__Offset () const
```

Returned value

Value of "Y__Offset" property of the object, assigned to the class instance. If there is no object assigned, it returns 0.

Y__Offset (Set Method)

Sets new value for "Y__Offset" property (the Y coordinate of the upper left corner of the rectangular visible area in the graphical objects) . The value is set in pixels relative to the upper left corner of the original image.

```
bool Y__Offset (
    int Y      // new value
)
```

Parameters

Y

[in] New value for "Y__Offset" property.

Returned value

true if successful, false if property hasn't changed.

Save

Saves object parameters to file.

```
virtual bool Save(  
    int file_handle    // File handle  
)
```

Parameters

file_handle

[in] handle of the binary file already opened by [FileOpen](#) function.

Returned value

true if successful, false if error.

Load

Loads object parameters from file.

```
virtual bool Load(  
    int file_handle    // File handle  
)
```

Parameters

file_handle

[in] handle of the binary file already opened by [FileOpen](#) function.

Returned value

true if successful, false if error.

Type

Returns graphic object type identifier.

```
virtual int Type() const
```

Returned value

Object type identifier (OBJ__BITMAP for [CChartObjectBitmap](#)) .

CChartObjectBmpLabel

Class CChartObjectBmpLabel is a class for simplified access to "Bitmap Label" graphic object properties.

Description

Class CChartObjectBmpLabel provides access to "Bitmap Label" object properties.

Declaration

```
class CChartObjectBmpLabel : public CChartObject
```

Title

```
#include <ChartObjects\ChartObjectsBmpControls.mqh>
```

Class Methods

Create	
Create	Creates graphic object "BmpLabel"
Properties	
X_Distance	Gets/Sets property "X_Distance"
Y_Distance	Gets/Sets property "Y_Distance"
X_Offset	Gets/Sets property "X_Offset"
Y_Offset	Gets/Sets property "Y_Offset"
Corner	Gets/Sets property "Corner"
X_Size	Gets/Sets property "X_Size"
Y_Size	Gets/Sets property "Y_Size"
BmpFileOn	Gets/Sets property "BmpFileOn" for button pressed state (On)
BmpFileOff	Gets/Sets property "BmpFileOff" for button depressed state (Off)
State	Gets/Sets property "Button State" (Pressed/Depressed)
Time	"Gag" for time coordinate change
Price	"Gag" for price coordinate change
Input/output	
virtual Save	Virtual method for writing to file
virtual Load	Virtual method for reading from file
virtual Type	Virtual method of identification

See also

[Object types](#), [Object properties](#), [Chart angle](#), [Graphic objects](#)

Create

Creates graphic object "BmpLabel".

```
bool Create(  
    long    chart_id,    // Chart identifier  
    string  name,        // Object name  
    int     window,      // Chart window  
    int     X,           // X coordinate  
    int     Y            // Y coordinate  
)
```

Parameters

chart_id

[in] Chart identifier (0 - current chart) .

name

[in] A unique name of the object to create.

window

[in] Chart window number (0 - base window) .

X

[in] X coordinate.

Y

[in] Y coordinate.

Returned value

true if successful, false if error.

X__Distance (Get Method)

Gets the value of "X__Distance" property.

```
int X__Distance() const
```

Returned value

Value of "X__Distance" property of the object, assigned to the class instance. If there is no object assigned, it returns 0.

X__Distance (Set Method)

Sets new value for "X__Distance" property.

```
bool X__Distance(  
    int X        // new value  
)
```

Parameters

X

[in] New value for "X__Distance" property.

Returned value

true if successful, false if property hasn't changed.

Y__Distance (Get Method)

Gets the value of "Y__Distance" property.

```
int Y__Distance() const
```

Returned value

Value of "Y__Distance" property of the object, assigned to the class instance. If there is no object assigned, it returns 0.

Y__Distance (Set Method)

Sets new value for "Y__Distance" property.

```
bool Y__Distance(  
    int Y        // new value  
)
```

Parameters

Y

[in] New value for "Y__Distance" property.

Returned value

true if successful, false if property hasn't changed.

X__Offset (Get Method)

Gets the value of "X__Offset" property (the X coordinate of the upper left corner of the rectangular visible area in the graphical objects) .

```
int X__Offset () const
```

Returned value

Value of "X__Offset" property of the object, assigned to the class instance. If there is no object assigned, it returns 0.

X__Offset (Set Method)

Sets new value for "X__Offset" property (the X coordinate of the upper left corner of the rectangular visible area in the graphical objects) . The value is set in pixels relative to the upper left corner of the original image.

```
bool X__Offset (
    int X          // new value
)
```

Parameters

X

[in] New value for "X__Offset" property.

Returned value

true if successful, false if property hasn't changed.

Y__Offset (Get Method)

Gets the value of "Y__Offset" property (the Y coordinate of the upper left corner of the rectangular visible area in the graphical objects) .

```
int Y__Offset () const
```

Returned value

Value of "Y__Offset" property of the object, assigned to the class instance. If there is no object assigned, it returns 0.

Y__Offset (Set Method)

Sets new value for "Y__Offset" property (the Y coordinate of the upper left corner of the rectangular visible area in the graphical objects) . The value is set in pixels relative to the upper left corner of the original image.

```
bool Y__Offset (
    int Y      // new value
)
```

Parameters

Y

[in] New value for "Y__Offset" property.

Returned value

true if successful, false if property hasn't changed.

Corner (Get Method)

Gets the value of "Corner" property.

```
ENUM_BASE_CORNER Corner() const
```

Returned value

Value of "Corner" property of the object, assigned to the class instance. If there is no object assigned, it returns WRONG_VALUE.

Corner (Set Method)

Sets new value for "Corner" property.

```
bool Corner(  
    ENUM_BASE_CORNER corner    // new value  
)
```

Parameters

corner

[in] New value for "Corner" property.

Returned value

true if successful, false if property hasn't changed.

X__Size

Gets the value of "X__Size" property.

```
int X__Size() const
```

Returned value

Value of "X__Size" property of the object, assigned to the class instance. If there is no object assigned, it returns 0.

Y_Size

Gets the value of "Y_Size" property.

```
int Y_Size() const
```

Returned value

Value of "Y_Size" property of the object, assigned to the class instance. If there is no object assigned, it returns 0.

BmpFileOn (Get Method)

Gets the value of "BmpFileOn" property.

```
string BmpFileOn() const
```

Returned value

Value of "BmpFileOn" property of the object, assigned to the class instance. If there is no object assigned, it returns "".

BmpFileOn (Set Method)

Sets new value for "BmpFileOn" property.

```
bool BmpFileOn (
    string name      // file name
)
```

Parameters

name

[in] New value for "BmpFileOn" property.

Returned value

true if successful, false if property hasn't changed.

BmpFileOff (Get Method)

Gets the value of "BmpFileOff" property.

```
string BmpFileOff() const
```

Returned value

Value of "BmpFileOff" property of the object, assigned to the class instance. If there is no object assigned, it returns "".

BmpFileOff (Set Method)

Sets new value for "BmpFileOff" property.

```
bool BmpFileOff(  
    string name    // file name  
)
```

Parameters

name

[in] New value for "BmpFileOff" property.

Returned value

true if successful, false if property hasn't changed.

State (Get Method)

Gets the value of "State" property.

```
bool State() const
```

Returned value

Value of "State" property of the object, assigned to the class instance. If there is no object assigned, it returns false.

State (Set Method)

Sets new value for "State" property.

```
bool State(  
    bool state    // new state value  
)
```

Parameters

x

[in] New value for "State" property.

Returned value

true if successful, false if property hasn't changed.

Time

Prohibits changes of the time coordinate.

```
bool Time(  
    datetime time    // any value  
)
```

Parameters

time

[in] Any value of datetime type.

Returned value

always false.

Price

Prohibits changes of the price coordinate.

```
bool Price(  
    double price    // any value  
)
```

Parameters

price

[in] Any value of double type.

Returned value

always false.

Save

Saves object parameters to file.

```
virtual bool Save(  
    int file_handle    // File handle  
)
```

Parameters

file_handle

[in] handle of the binary file already opened by [FileOpen](#) function.

Returned value

true if successful, false if error.

Load

Loads object parameters from file.

```
virtual bool Load(  
    int file_handle    // File handle  
)
```

Parameters

file_handle

[in] handle of the binary file already opened by [FileOpen](#) function.

Returned value

true if successful, false if error.

Type

Returns graphic object type identifier.

```
virtual int Type() const
```

Returned value

Object type identifier (OBJ__BITMAP__LABEL for [CChartObjectBmpLabel](#)) .

CChartObjectRectLabel

Class CChartObjectRectLabel is a class for simplified access to "Rectangle Label" graphic object properties.

Description

Class CChartObjectRectLabel provides access to "Rectangle Label" object properties.

Declaration

```
class CChartObjectRectLabel : public CChartObjectLabel
```

Title

```
#include <ChartObjects\ChartObjectsTxtControls.mqh>
```

Class Methods

Create	
Create	Creates "RectLabel" graphic object
Properties	
X_Size	Sets the horizontal size
Y_Size	Sets the vertical size
BackColor	Gets/Sets the background color
Angle	A gag method
BorderType	Gets/Sets type of the border
Input/output	
virtual Save	Virtual method for writing to file
virtual Load	Virtual method for reading from file
virtual Type	Virtual method of identification

See also

[Object types](#), [Object properties](#), [Graphic objects](#)

Create

Creates the "CChartObjectRectLabel" graphic object.

```
bool Create(  
    long    chart_id,    // Chart ID  
    string  name,        // Object name  
    int     window,      // Chart window  
    int     X,           // X coordinate  
    int     Y,           // Y coordinate  
    int     sizeX,       // Horizontal size  
    int     sizeY        // Vertical size  
)
```

Parameters

chart_id

[in] Chart identifier (0 - current chart) .

name

[in] A unique name of the object to create.

window

[in] Chart window number (0 - base window) .

X

[in] X coordinate.

Y

[in] Y coordinate.

sizeX

[in] Horizontal size.

sizeY

[in] Vertical size.

Returned value

true if successful, false if error.

X__Size

Sets the value of "X__Size" property.

```
bool X__Size(  
    int size    // Horizontal size  
)
```

Parameters

size

[in] New horizontal size.

Returned value

true if successful, false if the property hasn't been changed.

Note

To get the values of "X__Size" and "Y__Size" properties use the [X__Size](#) and [Y__Size](#) methods of the parent [CChartObjectLabel](#) class.

Y_Size

Sets the value of "Y_Size" property.

```
bool Y_Size(  
    int size    // Vertical size  
)
```

Parameters

size

[in] New vertical size.

Returned value

true if successful, false if the property hasn't been changed.

Note

To get the values of "X_Size" and "Y_Size" properties use the [X_Size](#) and [Y_Size](#) methods of the parent [CChartObjectLabel](#) class.

BackColor

Gets the background color.

```
color BackColor() const
```

Returned value

Background color of the object, assigned to the class instance. If there is no object assigned, it returns 0.

BackColor

Sets the background color.

```
bool BackColor(  
    color new_color    // New color  
)
```

Parameters

new_color

[in] New background color.

Returned value

true if successful, false if property hasn't been changed.

Angle

A gag method.

```
bool Angle(  
    double angle    // any value  
)
```

Parameters

angle

[in] Any value of [double](#) type.

Returned value

Always false.

BorderType

Gets border type.

```
int BorderType() const
```

Returned value

Border type of the object, assigned to the class instance. If there is no object assigned, it returns 0.

BorderType

Sets border type.

```
bool BorderType(  
    int type    // Border type  
)
```

Parameters

type

[in] New border type.

Returned value

true if successful, false if property hasn't been changed.

Save

Saves object parameters to file.

```
virtual bool Save(  
    int file_handle    // File handle  
)
```

Parameters

file_handle

[in] handle of the binary file already opened by [FileOpen](#) function.

Returned value

true if successful, false if error.

Load

Loads object parameters from file.

```
virtual bool Load(  
    int file_handle    // File handle  
)
```

Parameters

file_handle

[in] handle of the binary file already opened by [FileOpen](#) function.

Returned value

true if successful, false if error.

Type

Returns graphic object type identifier.

```
virtual int Type() const
```

Returned value

Object type identifier (OBJ__RECTANGLE__LABEL for [CChartObjectRectangleLabel](#)) .

CChart

Class CChart is a class for simplified access to "Chart" graphic object properties.

Description

Class CChart provides access to "Chart" object properties.

Declaration

```
class CChart : public CObject
```

Title

```
<Charts\Chart.mqh>
```

Class Methods

Access to protected data	
ChartID	Gets identifier of the chart
General properties	
Mode	Gets/Sets the value of "Mode" property (bar, candle or line)
Foreground	Gets/Sets the value of "Foreground" property
Shift	Gets/Sets the value of "Shift" property
ShiftSize	Gets/Sets the value of "ShiftSize" property (in percents)
AutoScroll	Gets/Sets the value of "AutoScroll" property
Scale	Gets/Sets the value of "Scale" property
ScaleFix	Gets/Sets the value of "ScaleFix" property (fixed chart scale or not)
ScaleFix__11	Gets/Sets the value of "ScaleFix__11" property (chart scale is 1:1, or not)
FixedMax	Gets/Sets the value of "FixedMax" property (fixed maximal price)
FixedMin	Gets/Sets the value of "FixedMin" property (fixed minimal price)
ScalePPB	Gets/Sets the value of "ScalePPB" property (scale is "point per bar" or not)
PointsPerBar	Gets/Sets the value of "PointsPerBar" property (in points per bar)
Show properties	
ShowOHLC	Gets/Sets the value of "ShowOHLC" property

<u>ShowLineBid</u>	Gets/Sets the value of "ShowLineBid" property
<u>ShowLineAsk</u>	Gets/Sets the value of "ShowLineAsk" property
<u>ShowLastLine</u>	Gets/Sets the value of "ShowLastLine" property
<u>ShowPeriodSep</u>	Gets/Sets the value of "ShowPeriodSep" property (show period separators)
<u>ShowGrid</u>	Gets/Sets the value of "ShowGrid" property
<u>ShowVolumes</u>	Gets/Sets the value of "ColorVolumes" property (color for volumes and levels of opened positions)
<u>ShowObjectDescr</u>	Gets/Sets the value of "ShowObjectDescr" property (show description for graphic objects)
<u>ShowDateScale</u>	Sets the value of "ShowDateScale" property (date scale of the chart)
<u>ShowPriceScale</u>	Sets the value of "ShowPriceScale" property (price scale of the chart)
Colors properties	
<u>ColorBackground</u>	Gets/Sets the value of "ColorBackground" property (background color of the chart)
<u>ColorForeground</u>	Gets/Sets the value of "ColorForeground" property (color of axes, scale and OHLC strings of the chart)
<u>ColorGrid</u>	Gets/Sets the value of "ColorGrid" property (color of the grid)
<u>ColorBarUp</u>	Gets/Sets the value of "ColorBarUp" property (color for bull bars, its shadow and candle body outlines)
<u>ColorBarDown</u>	Gets/Sets the value of "ColorBarDown" property (color for bear bars, its shadow and candle body outlines)
<u>ColorCandleBull</u>	Gets/Sets the value of "ColorCandleBull" property (body color of the bull candle)
<u>ColorCandleBear</u>	Gets/Sets the value of "ColorCandleBear" property (body color of the bear candle)
<u>ColorChartLine</u>	Gets/Sets the value of "ColorChartLine" property (color for line chart and Doji candles)
<u>ColorVolumes</u>	Gets/Sets the value of "ColorVolumes" property (color for volumes and levels of opened positions)
<u>ColorLineBid</u>	Gets/Sets the value of "ColorLineBid" property (color of Bid line)
<u>ColorLineAsk</u>	Gets/Sets the value of "ColorLineAsk" property (color of Ask line)
<u>ColorLineLast</u>	Gets/Sets the value of "ColorLineLast" property (color of the

	last deal price line)
ColorStopLevels	Gets/Sets the value of "ColorStopLevels" property (color of the SL and TP levels)
Read only properties	
VisibleBars	Gets total number of visible chart bars
WindowsTotal	Gets total number of chart windows, including the chart indicator subwindows
WindowsVisible	Gets visibility flag of the specified chart subwindow
WindowHandle	Gets window handle of the chart (HWND)
FirstVisibleBar	Gets the number of the first visible bar of the chart
WidthInBars	Gets window width in bars.
WidthInPixels	Gets subwindow width in pixels.
HeightInPixels	Gets subwindow height in pixels.
PriceMin	Gets minimal price of the specified subwindow
PriceMax	Gets maximal price of the specified subwindow
Properties	
Attach	Assigns the current chart to the class instance
FirstChart	Assigns the first chart of the client terminal to the class instance
NextChart	Assigns the next chart of the client terminal to the class instance
Open	Opens chart with specified parameters and assign it to the class instance
Detach	Detaches chart from the class instance
Close	Closes chart, assigned to the class instance
BringToTop	Show chart on top of other charts
EventObjectCreate	Sets a flag to send notifications of an event of new object creation on a chart
EventObjectDelete	Sets a flag to send notifications of an event of object deletion on a chart
Indicators	
IndicatorAdd	Adds an indicator with the specified handle into a specified chart subwindow
IndicatorDelete	Removes an indicator with a specified name from the specified chart subwindow
IndicatorsTotal	Returns the number of all indicators applied to the specified

	chart subwindow
IndicatorName	Returns the short name of the indicator on the specified chart subwindow
Navigation	
Navigate	Navigates the chart
Access to MQL5 API	
Symbol	Gets symbol of the chart
Period	Gets period of the chart
Redraw	Redraws chart, assigned to the class instance
GetInteger	The function returns the value of the corresponding object property
SetInteger	Sets new value for the property of the integer type
GetDouble	The function returns the value of the corresponding object property
SetDouble	Sets new value for the property of the double type
GetString	The function returns the value of the corresponding object property
SetString	Sets new value for the property of the string type
SetSymbolPeriod	Changes symbol and period of the chart, assigned to the class instance
ApplyTemplate	Applies specified template to the chart
ScreenShot	Creates screenshot of the specified chart and saves it to .gif file
WindowOnDropped	Gets chart subwindow number corresponding to the object (expert or script) drop point
PriceOnDropped	Gets price coordinate corresponding to the object (expert or script) drop point
TimeOnDropped	Gets time coordinate corresponding to the object (expert or script) drop point
XOnDropped	Gets X coordinate corresponding to the object (expert or script) drop point
YOnDropped	Gets Y coordinate corresponding to the object (expert or script) drop point
Input/Output	
virtual Save	Saves object parameters to file
virtual Load	Loads object parameters from file

virtual Type	Gets graphic object type identifier
------------------------------	-------------------------------------

ChartID

Returns identifier of the chart.

```
long ChartID() const
```

Returned value

Chart identifier, assigned to the class instance. If there is no object assigned, it returns -1.

Mode (Get Method)

Gets the value of "Mode" property (bar, candle or line) .

```
ENUM_CHART_MODE Mode() const
```

Returned value

Value of "Mode" property of the object, assigned to the class instance. If there is no object assigned, it returns [WRONG_VALUE](#).

Mode (Set Method)

Sets new value for "Mode" property (bar, candle or line) .

```
bool Mode(  
    ENUM_CHART_MODE mode    // new chart mode  
)
```

Parameters

mode

[in] Chart mode (candle, bar or line) of [ENUM_CHART_MODE](#) enumeration.

Returned value

true if successful, false if mode hasn't changed.

Foreground (Get Method)

Gets the value of "Foreground" property.

```
bool Foreground() const
```

Returned value

Value of "Foreground" property of the object, assigned to the class instance. If there is no object assigned, it returns false.

Foreground (Set Method)

Sets new value for "Foreground" property.

```
bool Foreground(  
    bool foreground    // new flag value  
)
```

Parameters

foreground

[in] New value for "Foreground" property.

Returned value

true if successful, false if property hasn't changed.

Shift (Get Method)

Gets the value of "Shift" property.

```
bool Shift() const
```

Returned value

Value of "Shift" property of the object, assigned to the class instance. If there is no object assigned, it returns false.

Shift (Set Method)

Sets new value for "Shift" property.

```
bool Shift(  
    bool shift    // new flag value  
)
```

Parameters

shift

[in] New value for "Shift" property.

Returned value

true if successful, false if property hasn't changed.

ShiftSize (Get Method)

Gets the value of "ShiftSize" property (in percents) .

```
double ShiftSize() const
```

Returned value

Value of "ShiftSize" property of the object, assigned to the class instance. If there is no object assigned, it returns [EMPTY_VALUE](#).

ShiftSize (Set Method)

Sets new value for "Shift" property (in percents) .

```
bool ShiftSize (
    double shift_size    // new property value
)
```

Parameters

shift_size

[in] New value for "ShiftSize" property (in percents) .

Returned value

true if successful, false if property hasn't changed.

AutoScroll (Get Method)

Gets the value of "AutoScroll" property.

```
bool AutoScroll() const
```

Returned value

Value of "AutoScroll" property of the object, assigned to the class instance. If there is no object assigned, it returns false.

AutoScroll (Set Method)

Sets new value for "AutoScroll" property.

```
bool AutoScroll(  
    bool autoscroll    // new flag value  
)
```

Parameters

autoscroll

[in] New value for "Autoscroll" property.

Returned value

true if successful, false if property hasn't changed.

Scale (Get Method)

Gets the value of "Scale" property.

```
int Scale() const
```

Returned value

Value of "Scale" property of the object, assigned to the class instance. If there is no object assigned, it returns 0.

Scale (Set Method)

Sets new value for "Scale" property.

```
bool Scale(  
    int scale    // new value  
)
```

Parameters

scale

[in] New value for "Scale" property.

Returned value

true if successful, false if property hasn't changed.

ScaleFix (Get Method)

Gets the value of "ScaleFix" property (fixed chart scale or not) .

```
bool ScaleFix() const
```

Returned value

Value of "ScaleFix" property of the object, assigned to the class instance. If there is no object assigned, it returns false.

ScaleFix (Set Method)

Sets new value for "ScaleFix" property.

```
bool ScaleFix(  
    bool scale_fix    // new value  
)
```

Parameters

scale_fix

[in] New value for "ScaleFix" property.

Returned value

true if successful, false if property hasn't changed.

ScaleFix__11 (Get Method)

Gets the value of "ScaleFix__11" property (chart scale is 1:1, or not) .

```
bool ScaleFix__11() const
```

Returned value

Value of "ScaleFix__11" property of the object, assigned to the class instance. If there is no object assigned, it returns false.

ScaleFix__11 (Set Method)

Sets new value for "ScaleFix__11" property.

```
bool ScaleFix__11(  
    string scale__11    // new value  
)
```

Parameters

scale__11

[in] New value for "ScaleFix__11" property.

Returned value

true if successful, false if property hasn't changed.

FixedMax (Get Method)

Gets the value of "FixedMax" property (fixed maximal price) .

```
double FixedMax() const
```

Returned value

Value of "FixedMax" property of the object, assigned to the class instance. If there is no object assigned, it returns [EMPTY_VALUE](#).

FixedMax (Set Method)

Sets new value for "FixedMax" property.

```
bool FixedMax(  
    double max    // new fixed maximum  
)
```

Parameters

max

[in] New value for "FixedMax" property.

Returned value

true if successful, false if property hasn't changed.

FixedMin (Get Method)

Gets the value of "FixedMin" property (fixed minimal price) .

```
double FixedMin() const
```

Returned value

Value of "FixedMin" property of the object, assigned to the class instance. If there is no object assigned, it returns [EMPTY_VALUE](#).

FixedMin (Set Method)

Sets new value for "FixedMin" property.

```
bool FixedMax(  
    double min    // new fixed minimum  
)
```

Parameters

max

[in] New value for "FixedMin" property.

Returned value

true if successful, false if property hasn't changed.

PointsPerBar (Get Method)

Gets the value of "PointsPerBar" property (in points per bar) .

```
double PointsPerBar() const
```

Returned value

Value of "PointsPerBar" property of the object, assigned to the class instance. If there is no object assigned, it returns [EMPTY_VALUE](#).

PointsPerBar (Set Method)

Sets new value for "PointsPerBar" property.

```
bool PointsPerBar (
    double ppb      // new scale (in points per bar)
)
```

Parameters

ppb

[in] New value for scale (in points per bar) .

Returned value

true if successful, false if scale hasn't changed.

ScalePPB (Get Method)

Gets the value of "ScalePPB" property (scale is "point per bar" or not) .

```
bool ScalePPB() const
```

Returned value

Value of "ScalePPB" property of the object, assigned to the class instance. If there is no object assigned, it returns false.

ScalePPB (Set Method)

Sets new value for "ScalePPB" property.

```
bool ScalePPB(  
    bool scale_ppb    // new flag value  
)
```

Parameters

scale_ppb

[in] New value for "ScalePPB" property.

Returned value

true if successful, false if property hasn't changed.

ShowOHLC (Get Method)

Gets the value of "ShowOHLC" property.

```
bool ShowOHLC() const
```

Returned value

Value of "ShowOHLC" property of the object, assigned to the class instance. If there is no object assigned, it returns false.

ShowOHLC (Set Method)

Sets new value for "ShowOHLC" property.

```
bool ShowOHLC(  
    bool show    // new value  
)
```

Parameters

show

[in] New value for "ShowOHLC" property.

Returned value

true if successful, false if property hasn't changed.

ShowLineBid (Get Method)

Gets the value of "ShowLineBid" property.

```
bool ShowLineBid() const
```

Returned value

Value of "ShowLineBid" property of the chart, assigned to the class instance. If there is no chart assigned, it returns false.

ShowLineBid (Set Method)

Sets new value for "ShowLineBid" property.

```
bool ShowLineBid(  
    bool show    // new value  
)
```

Parameters

show

[in] New value for "ShowLineBid" property.

Returned value

true if successful, false if property hasn't changed.

ShowLineAsk (Get Method)

Gets the value of "ShowLineAsk" property.

```
bool ShowLineAsk() const
```

Returned value

Value of "ShowLineAsk" property of the chart, assigned to the class instance. If there is no chart assigned, it returns false.

ShowLineAsk (Set Method)

Sets new value for "ShowLineAsk" property.

```
bool ShowLineAsk(  
    bool show    // new value  
)
```

Parameters

show

[in] New value for "ShowLineAsk" property.

Returned value

true if successful, false if property hasn't changed.

ShowLastLine (Get Method)

Gets the value of "ShowLastLine" property.

```
bool ShowLastLine() const
```

Returned value

Value of "ShowLastLine" property of the chart, assigned to the class instance. If there is no chart assigned, it returns false.

ShowLastLine (Set Method)

Sets new value for "ShowLastLine" property.

```
bool ShowLastLine(  
    bool show // new flag value  
)
```

Parameters

show

[in] New value for "ShowLastLine" property.

Returned value

true if successful, false if property hasn't changed.

ShowPeriodSep (Get Method)

Gets the value of "ShowPeriodSep" property (show period separators) .

```
bool ShowPeriodSep() const
```

Returned value

Value of "ShowPeriodSep" property of the chart, assigned to the class instance. If there is no chart assigned, it returns false.

ShowPeriodSep (Set Method)

Sets new value for "ShowPeriodSep" property.

```
bool ShowPeriodSep(  
    bool show // new value  
)
```

Parameters

show

[in] New value for "ShowPeriodSep" property.

Returned value

true if successful, false if property hasn't changed.

ShowGrid (Get Method)

Gets the value of "ShowGrid" property.

```
bool ShowGrid() const
```

Returned value

Value of "ShowGrid" property of the chart, assigned to the class instance. If there is no chart assigned, it returns false.

ShowGrid (Set Method)

Sets new value for "ShowGrid" property.

```
bool ShowGrid(  
    bool show    // new value  
)
```

Parameters

show

[in] New value for "ShowGrid" property.

Returned value

true if successful, false if property hasn't changed.

ShowVolumes (Get Method)

Gets the value of "ShowVolumes" property.

```
bool ShowVolumes() const
```

Returned value

Value of "ShowVolumes" property of the chart, assigned to the class instance. If there is no chart assigned, it returns false.

ShowVolumes (Set Method)

Sets new value for "ShowVolumes" property.

```
bool ShowVolumes (  
    bool show // new value  
)
```

Parameters

show

[in] New value for "ShowVolumes" property.

Returned value

true if successful, false if property hasn't changed.

ShowObjectDescr (Get Method)

Gets the value of "ShowObjectDescr" property (show description for graphic objects) .

```
bool ShowObjectDescr() const
```

Returned value

Value of "ShowObjectDescr" property of the chart, assigned to the class instance. If there is no chart assigned, it returns false.

ShowObjectDescr (Set Method)

Sets new value for "ShowObjectDescr" property.

```
bool ShowObjectDescr(  
    bool show // New value  
)
```

Parameters

show

[in] New value for "ShowObjectDescr" property.

Returned value

true if successful, false if property hasn't changed.

ShowDateScale

Sets new value for "ShowDateScale" property.

```
bool ShowDateScale(  
    bool show    // New value  
)
```

Parameters

show

[in] New value for "ShowDateScale" property.

Returned value

true if successful, false if property hasn't changed.

ShowPriceScale

Sets new value for "ShowPriceScale" property.

```
bool ShowPriceScale(  
    bool show    // New value  
)
```

Parameters

show

[in] New value for "ShowPriceScale" property.

Returned value

true if successful, false if property hasn't changed.

ColorBackground (Get Method)

Gets the value of "ColorBackground" property (background color of the chart) .

```
color ColorBackground() const
```

Returned value

Value of "ColorBackground" property of the chart, assigned to the class instance. If there is no chart assigned, it returns [CLR_NONE](#).

ColorBackground (Set Method)

Sets new value for "ColorBackground" property.

```
bool ColorBackground(  
    color new_color    // new background color  
)
```

Parameters

new_color

[in] New background color.

Returned value

true if successful, false if color hasn't changed.

ColorForeground (Get Method)

Gets the value of "ColorForeground" property (color of axes, scale and OHLC strings of the chart) .

```
color ColorForeground() const
```

Returned value

Value of "ColorForeground" property of the chart, assigned to the class instance. If there is no chart assigned, it returns [CLR_NONE](#).

ColorForeground (Set Method)

Sets new value for "ColorForeground" property (for axes, scale, and OHLC string) .

```
bool ColorForeground(  
    color new_color    // New color  
)
```

Parameters

new_color

[in] New color for axes, scale and OHLC string.

Returned value

true if successful, false if color hasn't changed.

ColorGrid (Get Method)

Gets the value of "ColorGrid" property (color of the grid) .

```
color ColorGrid() const
```

Returned value

Value of "ColorGrid" property of the chart, assigned to the class instance. If there is no chart assigned, it returns [CLR_NONE](#).

ColorGrid (Set Method)

Sets new value for "ColorGrid" property.

```
bool ColorGrid(  
    color new_color    // new grid color  
)
```

Parameters

new_color
[in] New grid color.

Returned value

true if successful, false if color hasn't changed.

ColorBarUp (Get Method)

Gets the value of "ColorBarUp" property (color for bull bars, its shadow and candle body outlines) .

```
color ColorBarUp() const
```

Returned value

Value of "ColorBarUp" property of the chart, assigned to the class instance. If there is no chart assigned, it returns [CLR_NONE](#).

ColorBarUp (Set Method)

Sets new value for "ColorBarUp" property.

```
bool ColorBarUp (
    color new_color      // new color for bull bars
)
```

Parameters

new_color

[in] New color for bull bars.

Returned value

true if successful, false if color hasn't changed.

ColorBarDown (Get Method)

Gets the value of "ColorBarDown" property (color for bear bars, its shadow and candle body outlines) .

```
color ColorBarDown() const
```

Returned value

Value of "ColorBarDown" property of the chart, assigned to the class instance. If there is no chart assigned, it returns [CLR_NONE](#).

ColorBarDown (Set Method)

Sets new value for "ColorBarDown" property.

```
bool ColorBarDown(  
    color new_color    // new color for bear bars  
)
```

Parameters

new_color

[in] New color for bear bars.

Returned value

true if successful, false if color hasn't changed.

ColorCandleBull (Get Method)

Gets the value of "ColorCandleBull" property (body color of the bull candle) .

```
color ColorCandleBull() const
```

Returned value

Value of "ColorCandleBull" property of the chart, assigned to the class instance. If there is no chart assigned, it returns [CLR_NONE](#).

ColorCandleBull (Set Method)

Sets new value for "ColorBarBull" property.

```
bool ColorCandleBull(  
    color new_color    // new color for bull candle body  
)
```

Parameters

new_color

[in] New color of the bull candle body.

Returned value

true if successful, false if color hasn't changed.

ColorCandleBear (Get Method)

Gets the value of "ColorCandleBear" property (body color of the bear candle) .

```
color ColorCandleBear() const
```

Returned value

Value of "ColorCandleBear" property of the chart, assigned to the class instance. If there is no chart assigned, it returns [CLR_NONE](#).

ColorCandleBear (Set Method)

Sets new value for "ColorBarBear" property.

```
bool ColorCandleBear(  
    color new_color    // new color for bear candle body  
)
```

Parameters

new_color

[in] New color of the bear candle body.

Returned value

true if successful, false if color hasn't changed.

ColorChartLine (Get Method)

Gets the value of "ColorChartLine" property (color for line chart and Doji candles) .

```
color ColorChartLine() const
```

Returned value

Value of "ColorChartLine" property of the chart, assigned to the class instance. If there is no chart assigned, it returns [CLR_NONE](#).

ColorChartLine (Set Method)

Sets new value for "ColorChartLine" property.

```
bool ColorChartLine(  
    color new_color    // new color of the chart lines  
)
```

Parameters

new_color

[in] New color of the chart lines (Doji candles) .

Returned value

true if successful, false if color hasn't changed.

ColorVolumes (Get Method)

Gets the value of "ColorVolumes" property (color for volumes and levels of opened positions) .

```
color ColorVolumes() const
```

Returned value

Value of "ColorVolumes" property of the chart, assigned to the class instance. If there is no chart assigned, it returns [CLR_NONE](#).

ColorVolumes (Set Method)

Sets new value for "ColorVolumes" property.

```
bool ColorVolumes (
    color new_color      // new color of the volumes (open position levels)
)
```

Parameters

new_color

[in] New color of the volumes (open position levels) .

Returned value

true if successful, false if color hasn't changed.

ColorLineBid (Get Method)

Gets the value of "ColorLineBid" property (color of Bid line) .

```
color ColorLineBid() const
```

Returned value

Value of "ColorLineBid" property of the chart, assigned to the class instance. If there is no chart assigned, it returns [CLR_NONE](#).

ColorLineBid (Set Method)

Sets new value for "ColorLineBid" property.

```
bool ColorLineBid(  
    color new_color    // new color for Bid line  
)
```

Parameters

new_color

[in] New color for Bid line.

Returned value

true if successful, false if color hasn't changed.

ColorLineAsk (Get Method)

Gets the value of "ColorLineAsk" property (color of Ask line) .

```
color ColorLineAsk() const
```

Returned value

Value of "ColorLineAsk" property of the chart, assigned to the class instance. If there is no chart assigned, it returns [CLR_NONE](#).

ColorLineAsk (Set Method)

Sets new value for "ColorLineAsk" property.

```
bool ColorLineAsk(  
    color new_color    // new color for Ask line  
)
```

Parameters

new_color

[in] New color for Ask line.

Returned value

true if successful, false if color hasn't changed.

ColorLineLast (Get Method)

Gets the value of "ColorLineLast" property (color of the last deal price line) .

```
color ColorLineLast() const
```

Returned value

Value of "ColorLineLast" property of the chart, assigned to the class instance. If there is no chart assigned, it returns [CLR_NONE](#).

ColorLineLast (Set Method)

Sets new value for "ColorLineLast" property.

```
bool ColorLineLast(  
    color new_color    // new color of the last deal price line  
)
```

Parameters

new_color

[in] New color of the last deal price line.

Returned value

true if successful, false if color hasn't changed.

ColorStopLevels (Get Method)

Gets the value of "ColorStopLevels" property (color of the SL and TP levels) .

```
color ColorStopLevels() const
```

Returned value

Value of "ColorStopLevels" property of the chart, assigned to the class instance. If there is no chart assigned, it returns [CLR_NONE](#).

ColorStopLevels (Set Method)

Sets new value for "ColorStopLevels" property.

```
bool ColorStopLevels (
    color new_color    // new color of the SL and TP price levels
)
```

Parameters

new_color

[in] New color of the Stop Loss and Take Profit price levels.

Returned value

true if successful, false if color hasn't changed.

VisibleBars

Gets total number of visible chart bars.

```
int VisibleBars() const
```

Returned value

Gets total number of visible bars of the chart, assigned to the class instance. If there is no chart assigned, it returns 0.

WindowsTotal

Gets total number of chart windows, including the chart indicator subwindows.

```
int WindowsTotal() const
```

Returned value

Total number of windows, including the chart indicator subwindows, assigned to the class instance. If there is no chart assigned, it returns 0.

WindowIsVisible

Gets visibility flag of the specified chart subwindow.

```
bool WindowIsVisible(  
    int num      // subwindow number  
) const
```

Parameters

num

[in] Subwindow number (0 means base window) .

Returned value

Returns visibility flag of the specified chart subwindow, assigned to the chart instance. If there is no chart assigned, it returns false.

WindowHandle

Gets window handle of the chart (HWND) .

```
int WindowHandle() const
```

Returned value

Window handle of the chart, assigned to the chart instance. If there is no chart assigned, it returns [INVALID_HANDLE](#).

FirstVisibleBar

Gets the number of the first visible bar of the chart.

```
int FirstVisibleBar() const
```

Returned value

Number of the first visible bar of the chart, assigned to the chart instance. If there is no chart assigned, it returns -1.

WidthInBars

Gets window width in bars.

```
int WidthInBars() const
```

Returned value

Window width in chart bars, assigned to the chart instance. If there is no chart assigned, it returns 0.

WidthInPixels

Gets subwindow width in pixels.

```
int WidthInPixels() const
```

Returned value

Subwindow width in chart pixels, assigned to the chart instance. If there is no chart assigned, it returns 0.

HeightInPixels

Gets subwindow height in pixels.

```
int HeightInPixels(  
    int num      // subwindow number  
) const
```

Parameters

num

[in] Subwindow number (0 means base window) .

Returned value

Subwindow height in chart pixels, assigned to the chart instance. If there is no chart assigned, it returns 0.

PriceMin

Gets minimal price of the specified subwindow.

```
double PriceMin(  
    int num      // subwindow number  
) const
```

Parameters

num

[in] Subwindow number (0 means base window) .

Returned value

Minimal price value of the chart, assigned to the class instance. If there is not chart assigned, it returns [EMPTY_VALUE](#).

PriceMax

Gets maximal price of the specified subwindow.

```
double PriceMax(  
    int num      // subwindow number  
) const
```

Parameters

num

[in] Subwindow number (0 means base window) .

Returned value

Maximal price value of the chart, assigned to the class instance. If there is not chart assigned, it returns [EMPTY_VALUE](#).

Attach

Assigns the current chart to the class instance.

```
void Attach()
```

Attach

Assigns the specified chart to the class instance.

```
void Attach(  
    long chart    // Chart identifier  
)
```

Parameters

chart

[in] Identifier of the chart to assign.

FirstChart

Assigns the first chart of the client terminal to the class instance.

```
void FirstChart()
```

NextChart

Assigns the next chart of the client terminal to the class instance.

```
void NextChart()
```

Open

Opens chart with specified parameters and assign it to the class instance.

```
long Open(  
    const string      symbol_name,      // Symbol name  
    ENUM_TIMEFRAMES  timeframe         // Period  
)
```

Parameters

symbol_name

[in] Symbol name. [NULL](#) means the symbol of the current chart (to which expert attached) .

timeframe

[in] Chart timeframe ([ENUM_TIMEFRAMES](#) enumeration) . 0 means the current timeframe.

Returned value

Chart identifier.

Detach

Detaches chart from the class instance.

```
void Detach()
```


Close

Closes chart, assigned to the class instance.

```
void Close()
```

BringToTop

Show chart on top of other charts.

```
bool WindowHandle() const
```

Returned value

true if successful, false if error.

EventObjectCreate

Sets a flag to send notifications of an [event](#) of new object creation to all MQL5-programs on a chart.

```
bool EventObjectCreate(  
    bool flag    // flag  
)
```

Parameters

flag

[in] New flag value.

Returned value

true - if successful, false - if flag hasn't been changed.

EventObjectDelete

Sets a flag to send notifications of an [event](#) of object deletion to all MQL5-programs on a chart.

```
bool EventObjectDelete(  
    bool flag    // flag  
)
```

Parameters

flag

[in] New flag value.

Returned value

true - if successful, false - if flag hasn't been changed.

IndicatorAdd

Adds an indicator with the specified handle into a specified chart window.

```
bool IndicatorAdd(  
    int sub_win           // number of the sub-window  
    int handle            // handle of the indicator  
);
```

Parameters

sub_win

[in] The number of the chart sub-window. 0 means the main chart window. if the number of a not-existing window is specified, a new window will be created.

handle

[in] The handle of the indicator.

Return Value

The function returns true in case of success, otherwise it returns false. In order to obtain information about the [error](#), call the [GetLastError\(\)](#) function.

See Also

[IndicatorDelete\(\)](#), [IndicatorsTotal\(\)](#), [IndicatorName\(\)](#).

IndicatorDelete

Removes an indicator with a specified name from the specified chart window.

```
bool IndicatorDelete(  
    int          sub_win      // number of the subwindow  
    const string name        // short name of the indicator  
);
```

Parameters

sub_win

[in] Number of the chart subwindow. 0 denotes the main chart subwindow.

const name

[in] The short name of the indicator which is set in the [INDICATOR_SHORTNAME](#) property with the [IndicatorSetString\(\)](#) function. To get the short name of an indicator use the [IndicatorName\(\)](#) function.

Return Value

Returns true in case of successful deletion of the indicator. Otherwise it returns false. To get [error](#) details use the [GetLastError\(\)](#) function.

Note

If two indicators with identical short names exist in the chart subwindow, the first one in a row will be deleted.

If other indicators on this chart are based on the values of the indicator that is being deleted, such indicators will also be deleted.

Do not confuse the indicator short name and the file name that is specified when creating an indicator using functions [iCustom\(\)](#) and [IndicatorCreate\(\)](#). If the short name of an indicator is not set explicitly, then the name of the file containing the source code of the indicator will be specified during compilation.

Deletion of an indicator from a chart doesn't mean that its calculation part will be deleted from the terminal memory. To release the indicator handle use the [IndicatorRelease\(\)](#) function.

The indicator's short name should be formed correctly. It will be written to the [INDICATOR_SHORTNAME](#) property using the [IndicatorSetString\(\)](#) function. It is recommended that the short name should contain values of all the input parameters of the indicator, because the indicator to be deleted from the chart by the [IndicatorDelete\(\)](#) function is identified by the short name.

See also

[IndicatorAdd\(\)](#), [IndicatorsTotal\(\)](#), [IndicatorName\(\)](#), [iCustom\(\)](#), [IndicatorCreate\(\)](#), [IndicatorSetString\(\)](#).

IndicatorsTotal

Returns the number of all indicators applied to the specified chart window.

```
int IndicatorsTotal(  
    int sub_win    // number of the subwindow  
);
```

Parameters

sub_win

[in] Number of the chart subwindow. 0 denotes the main chart subwindow.

Return Value

The number of indicators in the specified chart window. To get [error](#) details use the [GetLastError\(\)](#) function.

Note

The function allows going searching through all the indicators attached to the chart. The number of all the windows of the chart can be obtained from the [CHART_WINDOWS_TOTAL](#) property using the [GetInteger\(\)](#) function.

See also

[IndicatorAdd\(\)](#), [IndicatorDelete\(\)](#), [IndicatorsTotal\(\)](#), [iCustom\(\)](#), [IndicatorCreate\(\)](#), [IndicatorSetString\(\)](#).

IndicatorName

Returns the short name of the indicator by the index in the indicators list on the specified chart window.

```
string IndicatorName(  
    int    sub_win    // number of the subwindow  
    int    index      // index of the indicator in the list of indicators added to the  
);
```

Parameters

sub_win

[in] Number of the chart subwindow. 0 denotes the main chart subwindow.

index

[in] the index of the indicator in the list of indicators. The numeration of indicators start with zero, i. e. the first indicator in the list has the 0 index. To obtain the number of indicators in the list use the [IndicatorsTotal\(\)](#) function.

Return Value

The short name of the indicator which is set in the [INDICATOR_SHORTNAME](#) property with the [IndicatorSetString\(\)](#) function. To get [error](#) details use the [GetLastError\(\)](#) function.

Note

Do not confuse the indicator short name and the file name that is specified when creating an indicator using functions [iCustom\(\)](#) and [IndicatorCreate\(\)](#). If the short name of an indicator is not set explicitly, then the name of the file containing the source code of the indicator will be specified during compilation.

The indicator's short name should be formed correctly. It will be written to the [INDICATOR_SHORTNAME](#) property using the [IndicatorSetString\(\)](#) function. It is recommended that the short name should contain values of all the input parameters of the indicator, because the indicator to be deleted from the chart by the [IndicatorDelete\(\)](#) function is identified by the short name.

See also

[IndicatorAdd\(\)](#), [IndicatorDelete](#), [IndicatorsTotal](#), [iCustom\(\)](#), [IndicatorCreate\(\)](#), [IndicatorSetString\(\)](#).

Navigate

Navigates the chart.

```
bool  Navigate(  
    ENUM_CHART_POSITION  position,      // Position  
    int                  shift=0        // Shift  
)
```

Parameters

position

[in] Value of [ENUM_CHART_POSITION](#) enumeration.

shift=0

[in] Number of bars to shift.

Returned value

true if successful, false if chart hasn't navigated.

Symbol

Gets symbol of the chart.

```
string Symbol() const
```

Returned value

Symbol of the chart, assigned to the class instance. If there is no chart assigned, it returns 0.

Period

Gets period of the chart.

```
ENUM_TIMEFRAMES Period() const
```

Returned value

Period of the chart, assigned to the class instance. If there is no chart assigned, it returns 0.

Redraw

Redraws chart, assigned to the class instance.

```
void Redraw()
```

GetInteger

The function returns the value of the corresponding object property. The object property must be of the integer type. There are 2 variants of the function.

1. Immediately returns the property value.

```
long GetInteger (
    ENUM_CHART_PROPERTY_INTEGER prop_id,          // property identifier
    int sub_window=0                             // subwindow number
) const
```

2. If successful, puts the value of property to the specified variable of integer type, passed by reference as last parameter.

```
bool GetInteger (
    ENUM_CHART_PROPERTY_INTEGER prop_id,          // property identifier
    int sub_window,                             // subwindow number
    long& value                                   // here we get the property value
) const
```

Parameters

prop_id

[in] Property identifier ([ENUM_CHART_PROPERTY_INTEGER](#) enumeration) .

sub_window

[in] Chart subwindow number.

value

[in] Variable of the integer type that received the value of the requested property.

Return Value

Value of property of the chart, assigned to the class instance. If there isn't any chart assigned, it returns -1.

For the second variant the function returns true, if this property is maintained and the value has been placed into the value variable, otherwise returns false. To read more about the [error](#) call [GetLastError\(\)](#).

SetInteger

Sets new value for the property of the integer type.

```
bool SetInteger(  
    ENUM_CHART_PROPERTY_INTEGER prop_id,    // property identifier  
    long value                               // new value  
)
```

Parameters

prop_id

[in] Property identifier ([ENUM_CHART_PROPERTY_INTEGER](#) enumeration) .

value

[in] New value of the property.

Returned value

true if successful, false if property of the integer type hasn't changed.

GetDouble

The function returns the value of the corresponding object property. The object property must be of the double type. There are 2 variants of the function.

1. Immediately returns the property value.

```
double  GetDouble (
    ENUM_CHART_PROPERTY_DOUBLE prop_id,           // property identifier
    int      sub_window=0                        // subwindow number
) const
```

2. If successful, puts the value of property to the specified variable of double type, passed by reference as last parameter.

```
bool  GetDouble (
    ENUM_CHART_PROPERTY_DOUBLE prop_id,           // property identifier
    int      sub_window,                          // subwindow number
    double&   value                               // here we get the property value
) const
```

Parameters

prop_id

[in] Property identifier ([ENUM_CHART_PROPERTY_DOUBLE](#) enumeration) .

sub_window

[in] Chart subwindow number.

value

[in] Variable of the double type that received the value of the requested property.

Return Value

Value of property of the chart, assigned to the class instance. If there isn't any chart assigned, it returns EMPTY_VALUE.

For the second variant the function returns true, if this property is maintained and the value has been placed into the value variable, otherwise returns false. To read more about the [error](#) call [GetLastError\(\)](#).

SetDouble

Sets new value for the property of the double type.

```
bool SetDouble(  
    ENUM_CHART_PROPERTY_DOUBLE prop_id,    // property identifier  
    double value                            // new value  
)
```

Parameters

prop_id

[in] Property identifier ([ENUM_CHART_PROPERTY_DOUBLE](#) enumeration) .

value

[in] New value for the property.

Returned value

true if successful, false if property of the double type hasn't changed.

GetString

The function returns the value of the corresponding object property. The object property must be of the string type. There are 2 variants of the function.

1. Immediately returns the property value.

```
string GetString(  
    ENUM_CHART_PROPERTY_STRING prop_id    // property identifier  
) const
```

2. If successful, puts the value of property to the specified variable of string type, passed by reference as last parameter.

```
bool GetString(  
    ENUM_CHART_PROPERTY_STRING prop_id,    // property identifier  
    string& value                        // here we get the property value  
) const
```

Parameters

prop_id

[in] Property identifier ([ENUM_CHART_PROPERTY_STRING](#) enumeration) .

sub_window

[in] Chart subwindow number.

value

[in] Variable of the string type that received the value of the requested property.

Return Value

Value of property of the chart, assigned to the class instance. If there isn't any chart assigned, it returns "".

For the second variant the function returns true, if this property is maintained and the value has been placed into the value variable, otherwise returns false. To read more about the [error](#) call [GetLastError\(\)](#).

SetString

Sets new value for the property of the string type.

```
bool SetString(  
    ENUM_CHART_PROPERTY_STRING prop_id,    // property identifier  
    string value                            // new property value  
)
```

Parameters

prop_id

[in] Property identifier ([ENUM_CHART_PROPERTY_STRING](#) enumeration) .

value

[in] New value for the property.

Returned value

true if successful, false if property of the string type hasn't changed.

SetSymbolPeriod

Changes symbol and period of the chart, assigned to the class instance.

```
bool SetSymbolPeriod(  
    const string      symbol_name,      // Symbol  
    ENUM_TIMEFRAMES  timeframe         // Period  
)
```

Parameters

symbol_name

[in] New symbol name. [NULL](#) means the symbol of the current chart (to which expert attached) .

timeframe

[in] New chart timeframe ([ENUM_TIMEFRAMES](#) enumeration) . 0 means the current timeframe.

Returned value

true if successful, false if property hasn't changed.

ApplyTemplate

Applies specified template to the chart.

```
bool ApplyTemplate(  
    const string filename    // template file name  
)
```

Parameters

filename

[in] File name of the template.

Returned value

true if successful, false if template hasn't applied.

ScreenShot

Creates screenshot of the specified chart and saves it to .gif file.

```
bool ScreenShot(  
    string      filename,           // File name  
    int         width,              // Width  
    int         height,            // Height  
    ENUM_ALIGN_MODE align_mode=ALIGN_RIGHT // Align type  
) const
```

Parameters

filename

[in] File name for screenshot.

width

[in] Screenshot width in pixels.

height

[in] Screenshot height in pixels.

align_mode=ALIGN_RIGHT

[in] Align mode, if screenshot is narrow.

Returned value

true if successful, false if error.

WindowOnDropped

Gets chart subwindow number corresponding to the object (expert or script) drop point.

```
int WindowOnDropped() const
```

Returned value

Chart subwindow number of the object drop point. 0 means main chart window.

PriceOnDropped

Gets price coordinate corresponding to the object (expert or script) drop point.

```
double PriceOnDropped() const
```

Returned value

Price coordinate of the object drop point.

TimeOnDropped

Gets time coordinate corresponding to the object (expert or script) drop point.

```
datetime TimeOnDropped() const
```

Returned value

Time coordinate of the object drop point.

XOnDropped

Gets X coordinate corresponding to the object (expert or script) drop point.

```
int XOnDropped() const
```

Returned value

X coordinate of the object drop point.

YOnDropped

Gets Y coordinate corresponding to the object (expert or script) drop point.

```
int YOnDropped() const
```

Returned value

Y coordinate of the object drop point.

Save

Saves object parameters to file.

```
virtual bool Save(  
    int file_handle    // File handle  
)
```

Parameters

file_handle

[in] handle of the binary file already opened by [FileOpen](#)(...) function.

Returned value

true if successful, false if error.

Load

Loads object parameters from file.

```
virtual bool Load(  
    int file_handle    // File handle  
)
```

Parameters

file_handle

[in] handle of the binary file already opened by [FileOpen](#)(...) function.

Returned value

true if successful, false if error.

Type

Returns graphic object type identifier.

```
virtual int Type() const
```

Returned value

Object type identifier (0x1111 for CChart) .

File Operations

This section contains the technical details of the file operations classes and descriptions of the corresponding components of the standard MQL5 library.

The file operations classes use will save time in developing applications which uses file input/output operations.

The MQL5 Standard Library is placed in the working directory of the terminal in the Include\Files folder.

Class	Description
<u>CFile</u>	Base file operations class
<u>CFileBin</u>	Binary file operations class
<u>CFileTxt</u>	Text file operations class

CFile

CFile is a base class for CFileBin and CFileTxt classes.

Description

Class CFile provides the simplified access for all of its descendants to MQL5 API file and folder functions.

Declaration

```
class CFile: public CObject
```

Title

```
#include <Files\File.mqh>
```

Class Methods

Attributes	
Handle	Gets file handle
Filename	Gets file name
Flags	Gets file flags
SetUnicode	Sets/Clears the FILE__UNICODE flag
SetCommon	Sets/Clears the FILE__COMMON flag
General methods for files	
Open	Opens file
Close	Closes file
Delete	Deletes file
IsExist	Checks file for existence
Copy	Copies file
Move	Renames/moves file
Size	Gets file size
Tell	Gets current file position
Seek	Sets current file position
Flush	Flushes data on disk
IsEnding	Checks file for end
IsLineEnding	Checks line for end
General methods for	

folders	
FolderCreate	Creates folder
FolderDelete	Deletes folder
FolderClean	Clears folder
Search methods	
FileFindFirst	Begin file search
FileFindNext	Continue file search
FileFindClose	Close search handle

Derived classes:

- [CFileBin](#)
- [CFileTxt](#)

Handle

Gets file handle of the opened file.

```
int Handle()
```

Returned value

Handle of the opened file, assigned to the class instance. If there is no file assigned, it returns -1.

FileName

Gets file name of the opened file.

```
string FileName()
```

Returned value

File name of the opened file, assigned to the class instance. If there is no file assigned, it returns "".

Flags

Gets flags of the opened file.

```
int Flags ()
```

Returned value

Flags of the opened file, assigned to the class instance.

SetUnicode

Sets/Clears the FILE__UNICODE flag.

```
void SetUnicode(  
    bool unicode    // New flag value  
)
```

Parameters

unicode

[in] New value for FILE__UNICODE flag.

Note

The result of string operations is dependent on the FILE__UNICODE flag. If it's false, the ANSI codes are used (one byte symbols) . If it set, the UNICODE codes are used (two byte symbols) . If the file has already opened, the flag cannot be changed.

SetCommon

Sets/Clears the FILE__COMMON flag.

```
void SetCommon(  
    bool common    // New flag value  
)
```

Parameters

common

[in] New value for FILE__COMMON flag.

Note

The FILE__UNICODE flag determines the current work folder. If it's false, the local terminal folder used as the current work folder. If it's true, the general folder used as the current work folder. If the file has already opened, the flag cannot be changed.

Open

Open the specified file and if it successful, assigns it to the class instance.

```
int Open(  
    const string file_name,      // File name  
    int flags,                  // Flags  
    short delimiter=9           // Separator  
)
```

Parameters

file_name

[in] File name to open.

flags

[in] File open flags.

delimiter=9

[in] CSV file separator.

Returned value

Handle of the opened file.

Note

The work folder is dependent on the FILE__COMMON flag, defined by SetCommon() method.

Close

Closes file, assigned to the class instance.

```
void Close()
```

Delete

Deletes the file, assigned to the file instance.

```
void Delete()
```

Delete

Deletes the specified file.

```
void Delete(  
    const string file_name    // File name  
)
```

Parameters

file_name

[in] File name of the file to delete.

Note

The work folder is dependent on FILE__COMMON flag, defined by SetCommon() method.

IsExist

Checks file for existence

```
bool IsExist(  
    const string file_name    // File name  
)
```

Parameters

file_name

[in] Name of the file to check.

Returned value

true, if file exists.

Copy

Copies a file.

```
bool Copy(  
    const string src_name,      // Source file name  
    int src_flag,              // Flag  
    const string dst_name,      // Destination file name  
    int dst_flags               // Flags  
)
```

Parameters

src_name

[in] File name of the file to copy.

src_flag

[in] Flags of the file to copy (only FILE_COMMON is used) .

dst_name

[in] File name of the destination file.

dst_flags

[in] Flags of the destination file (only FILE_REWRITE and FILE_COMMON are used) .

Returned value

true if successful, false if it hasn't been copied.

Move

Renames/moves file.

```
bool Move(  
    const string src_name,      // Source file name  
    int src_flag,              // Flag  
    const string dst_name,      // Destination file name  
    int dst_flags               // Flags  
)
```

Parameters

src_name

[in] File name of the file to move.

src_flag

[in] Flags of the file to copy (only FILE_COMMON is used) .

dst_name

[in] File name of the destination file.

dst_flags

[in] Flags of the destination file (only FILE_REWRITE and FILE_COMMON are used) .

Returned value

true if successful, false if it hasn't been moved.

Size

Gets file size in bytes.

```
ulong Size()
```

Returned value

File size in bytes. If there isn't any file assigned, it returns ULONG_MAX.

Tell

Gets the current file position.

```
ulong Tell()
```

Returned value

The current file position. If there isn't any file assigned, it returns ULONG_MAX.

Seek

Sets current file position.

```
void Seek(  
    long          offset,      // Offset  
    ENUM_FILE_POSITION origin  // Origin  
)
```

Parameters

offset

[in] File offset in bytes (can be negative) .

origin

[in] Origin of the offset.

Returned value

true if successful, false if file position hasn't been changed.

Flush

Flushes all of the file input/output buffer data on disk.

```
void Flush()
```

IsEnding

Checks file for end. It's used during the file read operations.

```
bool IsEnding()
```

Returned value

true if end of file has been achieved after read or seek operation.

IsLineEnding

Checks file for end of line. It's used during the file read operations.

```
bool IsLineEnding()
```

Returned value

true if end of line has been achieved after the txt or csv file read operation (CR-LF chars) .

FolderCreate

Creates new folder.

```
bool FolderCreate(  
    const string folder_name    // Folder name  
)
```

Parameters

folder_name

[in] Name of the folder to create. It contains path to the folder relative to the folder defined by FILE__COMMON flag.

Returned value

true if successful, and false if the folder hasn't been created.

Note

The work folder is dependent on FILE__COMMON flag, defined by SetCommon() method.

FolderDelete

Deletes specified folder.

```
bool FolderDelete(  
    const string folder_name    // Folder name  
)
```

Parameters

folder_name

[in] Name of the folder to delete. It contains path to the folder relative to the folder defined by FILE_COMMON flag.

Returned value

true if successful, and false if the folder hasn't been deleted.

Note

The work folder is dependent on FILE_COMMON flag, defined by SetCommon() method.

FolderClean

Cleans specified folder.

```
bool FolderClean(  
    const string folder_name    // Folder name  
)
```

Parameters

folder_name

[in] Name of the folder to delete. It contains path to the folder relative to the folder defined by FILE_COMMON flag.

Returned value

true if successful, and false if the folder hasn't been cleaned.

Note

The work folder is dependent on FILE_COMMON flag, defined by SetCommon() method.

FileFindFirst

It began file search using the filter specified.

```
int FileFindFirst(  
    const string filter,           // Search Filter  
    string&      file_name        // Reference to string  
)
```

Parameters

filter

[in] Search filter.

file_name

[out] The reference to string for the first file found.

Returned value

If successful, it returns handle, that can be used for further file search using FileFindNext, or INVALID_HANDLE if there isn't any file corresponding to the filter specified.

Note

The work folder is dependent on FILE_COMMON flag, defined by SetCommon() method.

FileFindNext

It continue search, started by function FileFindFirst() .

```
bool FileFindNext(  
    int      search_handle,      // Search handle  
    string&  file_name          // Reference to string for the next file found  
)
```

Parameters

search_handle

[in] Search handle, returned by FileFindFirst() method.

file_name

[in] The reference to string for the name of the file found if successful.

Returned value

true if successful, false if there isn't any file, corresponding to the filter specified.

FileFindClose

Closes search handle.

```
void FileFindClose(  
    int search_handle    // Search handle  
)
```

Parameters

search_handle

[in] Search handle, returned by FileFindFirst() method.

CFileBin

CFileBin is a class for simplified access to binary files.

Description

Class CFileBin provides an access to binary files.

Declaration

```
class CFileBin: public CFile
```

Title

```
#include <Files\FileBin.mqh>
```

Class Methods

Open methods	
Open	Opens a binary file
Write methods	
WriteChar	Writes char or uchar type variable
WriteShort	Writes short or ushort type variable
WriteInteger	Writes int or uint type variable
WriteLong	Writes long or ulong type variable
WriteFloat	Writes float type variable
WriteDouble	Writes double type variable
WriteString	Writes string type variable
WriteCharArray	Writes an array of char or uchar type variables
WriteShortArray	Writes an array of short or ushort type variables
WriteIntegerArray	Writes an array of int or uint type variables
WriteLongArray	Writes an array of long or ulong type variables
WriteFloatArray	Writes an array of float variables
WriteDoubleArray	Writes an array of double type variables
WriteObject	Writes data of the CObject class inheritor instance
Read methods	
ReadChar	Reads char or uchar type variable
ReadShort	Reads short or ushort type variable
ReadInteger	Reads int or uint type variable

ReadLong	Reads long or ulong type variable
ReadFloat	Reads float type variable
ReadDouble	Reads double type variable
ReadString	Reads string type variable
ReadCharArray	Reads an array of char or uchar type variables
ReadShortArray	Reads an array of short or ushort type variables
ReadIntegerArray	Reads an array of int or uint type variables
ReadLongArray	Reads an array of long or ulong type variables
ReadFloatArray	Reads an array of float type variables
ReadDoubleArray	Reads an array of double type variables
ReadObject	Reads data of the CObject class inheritor instance

Open

Open the specified binary file and if it successful, assigns it to the class instance.

```
int Open(  
    const string file_name,    // File name  
    int flags                  // Flags  
)
```

Parameters

file_name

[in] File name of the file to open.

flags

[in] File open flags (there is a forced set of the FILE__BIN flag) .

Returned value

Handle of the opened file.

WriteChar

Writes char or uchar type variable to file.

```
uint WriteChar(  
    char value    // Value  
)
```

Parameters

value

[in] Variable to write.

Returned value

Number of bytes written.

WriteShort

Writes short or ushort type variable to file.

```
uint WriteShort(  
    short value    // Value  
)
```

Parameters

value

[in] Variable to write.

Returned value

Number of bytes written.

WriteInteger

Writes int or uint type variable to file.

```
uint WriteInteger(  
    int value    // Value  
)
```

Parameters

value

[in] Variable to write.

Returned value

Number of bytes written.

WriteLong

Writes long or ulong type variable to file.

```
uint WriteLong(  
    long value    // Value  
)
```

Parameters

value

[in] Variable to write.

Returned value

Number of bytes written.

WriteFloat

Writes float type variable to file.

```
uint WriteFloat(  
    float value    // Value  
)
```

Parameters

value

[in] Variable to write.

Returned value

Number of bytes written.

WriteDouble

Writes double type variable to file.

```
uint WriteDouble(  
    double value    // Value  
)
```

Parameters

value

[in] Variable to write.

Returned value

Number of bytes written.

WriteString

Writes string type variable to file.

```
uint WriteString(  
    const string value    // Value  
)
```

Parameters

value

[in] String to write.

Returned value

Number of bytes written.

WriteString

Writes string type variable to file.

```
uint WriteString(  
    const string value,    // Value  
    int size              // Size  
)
```

Parameters

value

[in] String to write.

size

[in] Number of bytes to write.

Returned value

Number of bytes written.

WriteCharArray

Writes an array of char or uchar type variables to file.

```
uint WriteCharArray(  
    char& array[],           // Array reference  
    int start_item=0,        // Start element  
    int items_count=-1       // Number of elements  
)
```

Parameters

array[]

[in] Array to write.

start_item=0

[in] Start element to write from.

items_count=-1

[in] Number of elements to write (-1 - for whole array) .

Returned value

Number of bytes written.

WriteShortArray

Writes an array of short or ushort type variables to file.

```
uint WriteShortArray(  
    short& array[],           // Array to write  
    int start_item=0,         // Start element  
    int items_count=-1        // Number of elements to write  
)
```

Parameters

array[]

[in] Array to write.

start_item=0

[in] Start element to write from.

items_count=-1

[in] Number of elements to write (-1 - for whole array) .

Returned value

Number of bytes written.

WriteIntegerArray

Writes an array of int or uint type variables to file.

```
uint WriteIntegerArray(  
    int& array[],           // Array to write  
    int start_item=0,       // Start element  
    int items_count=-1     // Number of elements to write  
)
```

Parameters

array[]

[in] Array to write.

start_item=0

[in] Start element to write from.

items_count=-1

[in] Number of elements to write (-1 - for whole array) .

Returned value

Number of bytes written.

WriteLongArray

Writes an array of long or ulong type variables to file.

```
uint WriteLongArray(  
    long& array[],           // Array to write  
    int start_item=0,        // Start element  
    int items_count=-1       // Number of elements to write  
)
```

Parameters

array[]

[in] Array to write.

start_item=0

[in] Start element to write from.

items_count=-1

[in] Number of elements to write (-1 - for whole array) .

Returned value

Number of bytes written.

WriteFloatArray

Writes an array of float type variables to file.

```
uint WriteFloatArray(  
    float& array[],           // Array to write  
    int    start_item=0,      // Start element  
    int    items_count=-1     // Number of elements to write  
)
```

Parameters

array[]

[in] Array to write.

start_item=0

[in] Start element to write from.

items_count=-1

[in] Number of elements to write (-1 - for whole array) .

Returned value

Number of bytes written.

WriteDoubleArray

Writes an array of double type variables to file.

```
uint WriteDoubleArray(  
    double& array[],           // Array to write  
    int start_item=0,          // Start element  
    int items_count=-1         // Number of elements to write  
)
```

Parameters

array[]

[in] Array to write.

start_item=0

[in] Start element to write from.

items_count=-1

[in] Number of elements to write (-1 - for whole array) .

Returned value

Number of bytes written.

WriteObject

Writes data of the CObject class inheritor instance to file.

```
bool WriteObject(  
    CObject* object    // Reference to the object  
)
```

Parameters

object

[in] Reference to the CObject class inheritor instance to write.

Returned value

true if successful, false if data hasn't been written.

ReadChar

Reads char or uchar type variable from file.

```
bool ReadChar(  
    char& value    // Target variable  
)
```

Parameters

value

[in] Target variable of type char.

Returned value

true if successful, false if data hasn't been read.

ReadShort

Reads short or ushort type variable from file.

```
bool ReadShort(  
    short& value  
)
```

Parameters

value

[in] Target variable of type short or ushort.

Returned value

true if successful, false if data hasn't been read.

ReadInteger

Reads int or uint type variable from file.

```
bool ReadInteger(  
    int& value    // Target variable  
)
```

Parameters

value

[in] Target variable of type int or uint.

Returned value

true if successful, false if data hasn't been read.

ReadLong

Reads long or ulong type variable from file.

```
bool ReadLong(  
    long& value  
)
```

Parameters

value

[in] Target variable of type long or ulong.

Returned value

true if successful, false if data hasn't been read.

ReadFloat

Reads float type variable from file.

```
bool ReadFloat(  
    float& value    // Target variable  
)
```

Parameters

value

[in] Target variable of type float.

Returned value

true if successful, false if data hasn't been read.

ReadDouble

Reads double type variable from file.

```
bool ReadDouble(  
    double& value  
)
```

Parameters

value

[in] Target variable of type double.

Returned value

true if successful, false if data hasn't been read.

ReadString

Reads string type variable from file.

```
bool ReadString(  
    string& value      // Target string  
)
```

Parameters

value

[in] Target variable of type string.

Returned value

true if successful, false if data hasn't been read.

ReadString

Reads string type variable from file.

```
bool ReadString(  
    string& value  
)
```

Parameters

value

[in] Target variable of type string.

Returned value

true if successful, false if data hasn't been read.

ReadCharArray

Reads an array of char or uchar type variables from file.

```
bool ReadCharArray(  
    char& array[],           // Target array  
    int start_item=0,        // Start element  
    int items_count=-1       // Number of elements to read  
)
```

Parameters

array[]

[in] Reference to the target array of type char or uchar.

start_item=0

[in] Start element to read from.

items_count=-1

[in] Number of elements to read (-1 - read to the end of file) .

Returned value

true if successful, false if data hasn't been read.

ReadShortArray

Reads an array of short or ushort type variables from file.

```
bool ReadShortArray(  
    short& array[],           // Target array  
    int start_item=0,         // Start element  
    int items_count=-1        // Number of elements to read  
)
```

Parameters

array[]

[in] Reference to the target array of type short or ushort.

start_item=0

[in] Start element to read from.

items_count=-1

[in] Number of elements to read (-1 - read to the end of file) .

Returned value

true if successful, false if data hasn't been read.

ReadIntegerArray

Reads an array of int or uint type variables from file.

```
bool ReadIntegerArray(  
    int& array[],           // Target array  
    int start_item=0,       // Start element  
    int items_count=-1      // Number of elements to read  
)
```

Parameters

array[]

[in] Reference to the target array of type int or uint.

start_item=0

[in] Start element to read from.

items_count=-1

[in] Number of elements to read (-1 - read to the end of file) .

Returned value

true if successful, false if data hasn't been read.

ReadLongArray

Reads an array of long or ulong type variables from file.

```
bool ReadLongArray(  
    long& array[],           // Target array  
    int start_item=0,       // Start element  
    int items_count=-1      // Number of elements to read  
)
```

Parameters

array[]

[in] Reference to the target array of type long or ulong.

start_item=0

[in] Start element to read from.

items_count=-1

[in] Number of elements to read (-1 - read to the end of file) .

Returned value

true if successful, false if data hasn't been read.

ReadFloatArray

Reads an array of float type variables from file.

```
bool ReadFloatArray(  
    float& array[],           // Target array  
    int start_item=0,        // Start element  
    int items_count=-1       // Number of elements to read  
)
```

Parameters

array[]

[in] Reference to the target array of type float.

start_item=0

[in] Start element to read from.

items_count=-1

[in] Number of elements to read (-1 - read to the end of file) .

Returned value

true if successful, false if data hasn't been read.

ReadDoubleArray

Reads an array of double type variables from file.

```
bool ReadDoubleArray(  
    double& array[],           // Target array  
    int start_item=0,         // Start element  
    int items_count=-1        // Number of elements to read  
)
```

Parameters

array[]

[in] Reference to the target array of type double.

start_item=0

[in] Start element to read from.

items_count=-1

[in] Number of elements to read (-1 - read to the end of file) .

Returned value

true if successful, false if data hasn't been read.

ReadObject

Reads data of the CObject class inheritor instance from file.

```
bool ReadObject(  
    CObject* object    // Reference to the object  
)
```

Parameters

object

[in] Reference to the target CObject class inheritor instance for read to.

Returned value

true if successful, false if data hasn't been read.

CFileTxt

CFileTxt is a class for simplified access to text files.

Description

Class CFileTxt provides an access to text files.

Declaration

```
class CFileTxt: public CFile
```

Title

```
#include <Files\FileTxt.mqh>
```

Class Methods

Open methods	
Open	Open a text file
Write methods	
WriteString	Writes string type variable to file
Read methods	
ReadString	Reads string type variable from file

Open

Open the specified text file and if it successful, assigns it to the class instance.

```
int Open(  
    const string file_name,    // file name  
    int flags                 // flags  
)
```

Parameters

file_name

[in] File name to open.

flags

[in] File open flags (there is a forced set of the FILE__TXT flag) .

Returned value

Opened file handle.

WriteString

Writes string type variable to file.

```
uint WriteString(  
    const string value    // String to write  
)
```

Parameters

value

[in] String to write.

Returned value

Number of bytes written.

ReadString

Reads string type variable from file.

```
string ReadString()
```

Returned value

String which has been read.

String operations

This section contains the technical details of the string operations classes and descriptions of the corresponding components of the standard MQL5 library.

The use of string operations classes will save time in developing applications which uses text processing operations.

The MQL5 Standard Library is placed in the working directory of the terminal in the Include\Strings folder.

Class	Description
<u>CString</u>	Class for string operations

CString

CString is a class for simplified access to the variables of string type.

Description

Class CFile provides the simplified access for all of its descendants to MQL5 API string functions.

Declaration

```
class CString: public CObject
```

Title

```
#include <Strings\String.mqh>
```

Class Methods

Data access methods	
Str	Gets a string
Len	Gets length of a string
Copy	Copies a string
Fill methods	
Fill	Fills a string with specified char
Assign	Assigns a string
Append	Appends a string
Insert	Inserts a string
Compare methods	
Compare	Compares a string
CompareNoCase	Compares a strings case insensitive
Substring methods	
Left	Gets a specified number of characters from the left side of a string
Right	Gets a specified number of characters from the right side of a string
Mid	Gets a specified number of characters from a string
Trim/delete methods	
Trim	Removes all leading and trailing occurrences of a set of specified characters from a string
TrimLeft	Removes all leading occurrences of a set of specified characters from a string

TrimRight	Removes all trailing occurrences of a set of specified characters from a string
Clear	Clears a string
Convert methods	
ToUpper	Converts a string to uppercase.
ToLower	Converts a string to lowercase.
Reverse	Reverses a string
Search methods	
Find	Searches for the first match of a substring
FindRev	Searches for the last match of a substring
Remove	Deletes a substring from a string
Replace	Replaces a substring

Str

Gets a string.

```
string Str() const;
```

Returned value

Copy of a string.

Len

Gets length of a string.

```
uint Len() const;
```

Returned value

Length of a string.

Copy

Copies a string by reference.

```
void Copy(  
    string& copy      // Reference  
    ) const;
```

Parameters

copy

[in] Reference to a string to copy.

Copy

Copies a string to the CString class instance.

```
void Copy(  
    CString* copy      // Object descriptor  
    ) const;
```

Parameters

copy

[in] CString class object descriptor.

Fill

Fills a string with specified char.

```
bool Fill(  
    short character    // Character  
)
```

Parameters

character

[in] Character for filling.

Returned value

true if successful, false if a string hasn't been filled.

Assign

Assigns a string.

```
void Assign(  
    const string str      // String to assign  
)
```

Parameters

str

[in] String to assign.

Assign

Assigns a string to the CString class instance.

```
void Assign(  
    CString* str          // Object descriptor  
)
```

Parameters

str

[in] CString class object descriptor to assign.

Append

Appends a string.

```
void Append(  
    const string str      // String to append  
)
```

Parameters

str

[in] String to append.

Append

Appends a string to the CString class instance.

```
void Append(  
    CString* string      // Object descriptor  
)
```

Parameters

string

[in] CString class object descriptor to append.

Insert

Inserts a string to the specified position.

```
uint Insert(  
    uint      pos,      // Position  
    const string str     // String to insert  
)
```

Parameters

pos

[in] Insert position.

str

[in] String to insert.

Returned value

Resulted string length.

Insert

Inserts a string to the specified position to the CString class instance.

```
uint Insert(  
    uint      pos,      // Position  
    CString*  str       // Object descriptor  
)
```

Parameters

pos

[in] Insert position.

str

[in] CString class object descriptor to insert.

Returned value

Resulted string length.

Compare

Compares a string.

```
int Compare(  
    const string str    // String to compare  
    ) const;
```

Parameters

str

[in] String to compare.

Returned value

It returns 0 if strings are equal, -1 if a string of the class is lower than a string to compare, -1 if the class string is greater than a string to compare.

Compare

Compares a string with a string of the CString class instance.

```
int Compare(  
    CString* str        // Object descriptor  
    ) const;
```

Parameters

str

[in] CString class object descriptor to compare.

Returned value

It returns 0 if strings are equal, -1 if a string of the class is lower than a string to compare, -1 if the class string is greater than a string to compare.

CompareNoCase

Compares a strings case insensitive.

```
int CompareNoCase(  
    const string str      // String to compare  
    ) const;
```

Parameters

str

[in] String to compare.

Returned value

It returns 0 if a strings are equal, -1 if a string of the class is lower than a string to compare, -1 if the class string greater than a string to compare.

CompareNoCase

Compares a string (case insensitive) with a string of the CString class instance.

```
int CompareNoCase(  
    CString* str          // Object descriptor  
    ) const;
```

Parameters

str

[in] CString class object descriptor to compare.

Returned value

It returns 0 if a strings are equal, -1 if a string of the class is lower than a string to compare, -1 if the class string greater than a string to compare.

Left

Gets a specified number of characters from the left side of a string.

```
string Left(  
    uint count    // Number of characters  
)
```

Parameters

count

[in] Number of characters.

Returned value

Resulted substring.

Right

Gets a specified number of characters from the right side of a string.

```
string Right(  
    uint count    // Number of characters  
)
```

Parameters

count

[in] Number of characters.

Returned value

Resulted substring.

Mid

Gets a specified number of characters from a string.

```
string Mid(  
    uint pos,           // Position  
    uint count          // Number of characters  
)
```

Parameters

pos

[in] Position of a string.

count

[in] Number of characters.

Returned value

Resulted substring.

Trim

Removes all leading and trailing occurrences of a set of specified characters (and ' ', '\t', '\r', '\n') from a string.

```
int Trim(  
    const string targets    // Set of characters to remove  
)
```

Parameters

targets

[in] Set of characters to remove.

Returned value

Number of characters removed.

Example:

```
//--- example for CString::Trim  
#include <Strings\String.mqh>  
//---  
void OnStart()  
{  
    CString str;  
    //---  
    str.Assign("    \t\tABCD\r\n");  
    printf("Source string '%s'", str.Str());  
    //---  
    str.Trim("DA-DA-DA");  
    printf("Result string '%s'", str.Str());  
}
```

TrimLeft

Removes all leading occurrences of a set of specified characters (and '\t','r','n') from a string.

```
int TrimLeft(  
    const string targets    // Set of characters to remove  
)
```

Parameters

targets

[in] Set of characters to remove.

Returned value

Number of characters removed.

TrimRight

Removes all trailing occurrences of a set of specified characters (and ' ', '\t', '\r', '\n') from a string.

```
int TrimRight(  
    const string targets    // Set of characters to remove  
)
```

Parameters

targets

[in] Set of characters to remove.

Returned value

Number of characters removed.

Clear

Clears a string.

```
bool Clear()
```

Returned value

true if successful, false if a string hasn't been cleared.

ToUpper

Converts a string to uppercase.

```
bool ToUpper ()
```

Returned value

true if successful, false if a string hasn't been converted.

ToLower

Converts a string to lowercase.

```
bool ToLower()
```

Returned value

true if successful, false if a string hasn't been converted.

Reverse

Reverses of a string.

```
void Reverse()
```


Find

Searches for the first match of a substring.

```
int Find(  
    uint      start,          // Position  
    const string substring    // Substring to search for  
) const;
```

Parameters

start

[in] The index of the character in the string to begin the search with, or 0 to start from the beginning.

substring

[in] Substring to search for.

Returned value

The index of the first character that matches the requested substring; -1 if the substring is not found.

FindRev

Searches for the last match of a substring.

```
int FindRev(  
    const string  substring    // Substring  
    ) const;
```

Parameters

substring

[in] A substring to search for.

Returned value

The index of the last character that matches the requested substring; -1 if the substring is not found.

Remove

Deletes a substring from a string.

```
uint Remove(  
    const string substring // Substring to remove  
)
```

Parameters

substring

[in] A substring to search for.

Returned value

Number of substrings deleted.

Replace

Replaces a substring from a string.

```
uint Replace(  
    const string  substring,    // Substring to replace  
    const string  newstring     // New substring  
)
```

Parameters

substring

[in] A substring to search for.

newstring

[in] A substring to replace for.

Returned value

Number of substrings replaced.

Technical Indicators and Timeseries

This section contains the technical details of the technical indicator and timeseries classes and description of the corresponding components of the standard MQL5 library.

The use of the technical indicator and timeseries classes will save time in developing applications (scripts, Expert Advisors) .

The MQL5 Standard Library is placed in the working directory of the terminal in the Include\Indicators folder.

Class/group	Description
Base classes	Group of base and auxiliary classes
Timeseries classes	Group of timeseries classes
Trend Indicators	Group of Trend indicator classes
Oscillators	Group of Oscillator indicator classes
Volume Indicators	Group of Volume indicator classes
Bill Williams Indicators	Bill Williams indicator classes
Custom indicators	Custom indicator class

Base and Auxiliary Technical Indicator and Timeseries Classes

This section contains the technical details of base and auxiliary technical indicator and timeseries classes and description of the corresponding components of the Standard MQL5 library.

Class/group	Description
<u>CSpreadBuffer</u>	Historical spread buffer class
<u>CTimeBuffer</u>	Historical opening prices buffer class
<u>CTickVolumeBuffer</u>	Historical tick volumes buffer class
<u>CRealVolumeBuffer</u>	Historical real volumes buffer class
<u>CDoubleBuffer</u>	Base class of double type data buffer
<u>COpenBuffer</u>	Opening bar prices buffer class
<u>CHighBuffer</u>	High bar prices buffer class
<u>CLowBuffer</u>	Low bar prices buffer class
<u>CCloseBuffer</u>	Closing bar prices buffer class
<u>CIndicatorBuffer</u>	Technical indicator buffer class
<u>CSeries</u>	Base class for access to timeseries data
<u>CPriceSeries</u>	Base class for access to price data
<u>CIndicator</u>	Base class of technical indicator
<u>CIndicators</u>	Technical indicator and series collection

CSpreadBuffer

CSpreadBuffer is a class for simplified access to spreads of the bars in the history.

Description

The CSpreadBuffer class provides an access to spreads of the bars in the history.

Declaration

```
class CSpreadBuffer: public CArrayInt
```

Title

```
#include <Indicators\TimeSeries.mqh>
```

Class Methods

Attributes	
Size	Sets buffer size
Settings	
SetSymbolPeriod	Sets symbol and period
Data Access Methods	
At	Gets the buffer element by index
Data Update Methods	
virtual Refresh	Updates the buffer
virtual RefreshCurrent	Updates the current value

Size

Sets buffer size.

```
void Size(  
    int size    // new size  
)
```

Parameters

size

[in] New buffer size.

SetSymbolPeriod

Sets symbol and period.

```
void SetSymbolPeriod(  
    string          symbol,      // symbol  
    ENUM_TIMEFRAMES period      // period  
)
```

Parameters

symbol

[in] New symbol.

period

[in] New period ([ENUM_TIMEFRAMES](#) enumeration) .

At

Gets the buffer element by index.

```
long At(  
    int index    // index  
)
```

Parameters

index

[in] Index of buffer element.

Returned value

Buffer element with the specified index.

Refresh

Updates the buffer.

```
virtual bool Refresh()
```

Returned value

true - if successful, false - if the buffer hasn't been updated.

RefreshCurrent

Updates the current (zeroth) element of the buffer.

```
virtual bool RefreshCurrent()
```

Returned value

true - if successful, false - if the buffer hasn't been updated.

CTimeBuffer

CTimeBuffer is a class for simplified access to opening times of the bars in the history.

Description

The CTimeBuffer class provides an access to opening times of the bars in the history.

Declaration

```
class CTimeBuffer: public CArrayLong
```

Title

```
#include <Indicators\TimeSeries.mqh>
```

Class Methods

Attributes	
Size	Sets buffer size
Settings	
SetSymbolPeriod	Sets symbol and period
Data Access Methods	
At	Gets the buffer element by index
Data Update Methods	
virtual Refresh	Updates the buffer
virtual RefreshCurrent	Updates the current value

Size

Sets buffer size.

```
void Size(  
    int size    // new size  
)
```

Parameters

size

[in] New buffer size.

SetSymbolPeriod

Sets symbol and period.

```
void SetSymbolPeriod(  
    string      symbol,      // symbol  
    ENUM_TIMEFRAMES period  // period  
)
```

Parameters

symbol

[in] New symbol.

period

[in] New period ([ENUM_TIMEFRAMES](#) enumeration) .

At

Gets the buffer element by index.

```
long At(  
    int index    // index  
)
```

Parameters

index

[in] Index of buffer element.

Returned value

Buffer element with the specified index.

Refresh

Updates the buffer.

```
virtual bool Refresh()
```

Returned value

true - if successful, false - if the buffer hasn't been updated.

RefreshCurrent

Updates the current (zeroth) element of the buffer.

```
virtual bool RefreshCurrent()
```

Returned value

true - if successful, false - if the buffer hasn't been updated.

CTickVolumeBuffer

CTickVolumeBuffer is a class for simplified access to tick volumes of bars in the history.

Description

The CTickVolumeBuffer class provides an access to tick volumes of bars in the history.

Declaration

```
class CTickVolumeBuffer: public CArrayLong
```

Title

```
#include <Indicators\TimeSeries.mqh>
```

Class Methods

Attributes	
Size	Sets buffer size
Settings	
SetSymbolPeriod	Sets symbol and period
Data Access Methods	
At	Gets the buffer element by index
Data Update Methods	
virtual Refresh	Updates the buffer
virtual RefreshCurrent	Updates the current value

Size

Sets buffer size.

```
void Size(  
    int size    // new size  
)
```

Parameters

size

[in] New buffer size.

SetSymbolPeriod

Sets symbol and period.

```
void SetSymbolPeriod(  
    string          symbol,      // symbol  
    ENUM_TIMEFRAMES period      // period  
)
```

Parameters

symbol

[in] New symbol.

period

[in] New period ([ENUM_TIMEFRAMES](#) enumeration) .

At

Gets the buffer element by index.

```
long At(  
    int index    // index  
)
```

Parameters

index

[in] Index of buffer element.

Returned value

Buffer element with the specified index.

Refresh

Updates the buffer.

```
virtual bool Refresh()
```

Returned value

true - if successful, false - if the buffer hasn't been updated.

RefreshCurrent

Updates the current (zeroth) element of the buffer.

```
virtual bool RefreshCurrent()
```

Returned value

true - if successful, false - if the buffer hasn't been updated.

CRealVolumeBuffer

CRealVolumeBuffer is a class for simplified access to real volumes of bars in the history.

Description

The CRealVolumeBuffer class provides an access to real volumes of bars in the history.

Declaration

```
class CRealVolumeBuffer: public CArrayLong
```

Title

```
#include <Indicators\TimeSeries.mqh>
```

Class Methods

Attributes	
Size	Sets buffer size
Settings	
SetSymbolPeriod	Sets symbol and period
Data Access Methods	
At	Gets the buffer element by index
Data Update Methods	
virtual Refresh	Updates the buffer
virtual RefreshCurrent	Updates the current value

Size

Sets buffer size.

```
void Size(  
    int size    // new size  
)
```

Parameters

size

[in] New buffer size.

SetSymbolPeriod

Sets symbol and period.

```
void SetSymbolPeriod(  
    string          symbol,      // symbol  
    ENUM_TIMEFRAMES period      // period  
)
```

Parameters

symbol

[in] New symbol.

period

[in] New period ([ENUM_TIMEFRAMES](#) enumeration) .

At

Gets the buffer element by index.

```
long At(  
    int index    // index  
)
```

Parameters

index

[in] Index of buffer element.

Returned value

Buffer element with the specified index.

Refresh

Updates the buffer.

```
virtual bool Refresh()
```

Returned value

true - if successful, false - if the buffer hasn't been updated.

RefreshCurrent

Updates the current (zeroth) element of the buffer.

```
virtual bool RefreshCurrent()
```

Returned value

true - if successful, false - if the buffer hasn't been updated.

CDoubleBuffer

CDoubleBuffer is a base class for simplified access to data buffers of double type.

Description

The CDoubleBuffer class provides an access to the data of the buffer of double type.

Declaration

```
class CDoubleBuffer: public CArrayDouble
```

Title

```
#include <Indicators\TimeSeries.mqh>
```

Class Methods

Attributes	
Size	Sets buffer size
Settings	
SetSymbolPeriod	Sets symbol and period
Data Access Methods	
At	Gets the buffer element
Data Update Methods	
virtual Refresh	Updates the buffer
virtual RefreshCurrent	Updates the current value
Data Search Methods	
Minimum	Gets index of the lowest element of the buffer in the specified range
Maximum	Gets index of the highest element of the buffer in the specified range

Size

Sets buffer size.

```
void Size(  
    int size    // new size  
)
```

Parameters

size

[in] New buffer size.

SetSymbolPeriod

Sets symbol and period.

```
void SetSymbolPeriod(  
    string          symbol,      // symbol  
    ENUM_TIMEFRAMES period      // period  
)
```

Parameters

symbol

[in] New symbol.

period

[in] New period ([ENUM_TIMEFRAMES](#) enumeration) .

At

Gets the buffer element by index.

```
long At(  
    int index    // index  
)
```

Parameters

index

[in] Index of buffer element.

Returned value

Buffer element with the specified index.

Refresh

Updates the buffer.

```
virtual bool Refresh()
```

Returned value

true - if successful, false - if the buffer hasn't been updated.

RefreshCurrent

Updates the current (zeroth) element of the buffer.

```
virtual bool RefreshCurrent()
```

Returned value

true - if successful, false - if the buffer hasn't been updated.

Minimum

Gets index of the lowest element of the buffer in the specified range.

```
int Minimum(  
    int start,      // starting index  
    int count       // number of elements to proceed  
) const
```

Parameters

start

[in] Starting index of the array.

count

[in] Number of elements to proceed.

Returned value

Index of the lowest element of the buffer in the specified range.

Maximum

Gets index of the highest element of the buffer in the specified range.

```
int Maximum(  
    int start,      // starting index  
    int count       // number of elements to proceed  
) const
```

Parameters

start

[in] Starting index of the array.

count

[in] Number of elements to proceed.

Returned value

Index of the highest element of the buffer in the specified range.

COpenBuffer

COpenBuffer is a class for simplified access to open prices of bars in the history.

Description

The COpenBuffer class provides an access to open prices of bars in the history.

Declaration

```
class COpenBuffer: public CDoubleBuffer
```

Title

```
#include <Indicators\TimeSeries.mqh>
```

Class Methods

Data Update Methods	
virtual Refresh	Updates the buffer
virtual RefreshCurrent	Updates the current value

Refresh

Updates the buffer.

```
virtual bool Refresh()
```

Returned value

true - if successful, false - if the buffer hasn't been updated.

RefreshCurrent

Updates the current (zeroth) element of the buffer.

```
virtual bool RefreshCurrent()
```

Returned value

true - if successful, false - if the buffer hasn't been updated.

CHighBuffer

CHighBuffer is a class for simplified access to high prices of bars in the history.

Description

The CHighBuffer class provides an access to high prices of bars in the history.

Declaration

```
class CHighBuffer: public CDoubleBuffer
```

Title

```
#include <Indicators\TimeSeries.mqh>
```

Class Methods

Data Update Methods	
virtual Refresh	Updates the buffer
virtual RefreshCurrent	Updates the current value

Refresh

Updates the buffer.

```
virtual bool Refresh()
```

Returned value

true - if successful, false - if the buffer hasn't been updated.

RefreshCurrent

Updates the current (zeroth) element of the buffer.

```
virtual bool RefreshCurrent()
```

Returned value

true - if successful, false - if the buffer hasn't been updated.

CLowBuffer

CLowBuffer is a class for simplified access to low prices of bars in the history.

Description

The CLowBuffer class provides an access to low prices of bars in the history.

Declaration

```
class CLowBuffer: public CDoubleBuffer
```

Title

```
#include <Indicators\TimeSeries.mqh>
```

Class Methods

Data Update Methods	
virtual Refresh	Updates the buffer
virtual RefreshCurrent	Updates the current value

Refresh

Updates the buffer.

```
virtual bool Refresh()
```

Returned value

true - if successful, false - if the buffer hasn't been updated.

RefreshCurrent

Updates the current (zeroth) element of the buffer.

```
virtual bool RefreshCurrent()
```

Returned value

true - if successful, false - if the buffer hasn't been updated.

CCloseBuffer

CCloseBuffer is a class for simplified access to close prices of bars in the history.

Description

The CCloseBuffer class provides an access to close prices of bars in the history.

Declaration

```
class CCloseBuffer: public CDoubleBuffer
```

Title

```
#include <Indicators\TimeSeries.mqh>
```

Class Methods

Data Update Methods	
virtual Refresh	Updates the buffer
virtual RefreshCurrent	Updates the current value

Refresh

Updates the buffer.

```
virtual bool Refresh()
```

Returned value

true - if successful, false - if the buffer hasn't been updated.

RefreshCurrent

Updates the current (zeroth) element of the buffer.

```
virtual bool RefreshCurrent()
```

Returned value

true - if successful, false - if the buffer hasn't been updated.

CIndicatorBuffer

CIndicatorBuffer is a class for simplified access to the data of the indicator's buffer.

Description

The CIndicatorBuffer class provides the simplified access to the data buffer of technical indicator.

Declaration

```
class CIndicatorBuffer: public CDoubleBuffer
```

Title

```
#include <Indicators\Indicator.mqh>
```

Class Methods

Attributes	
Offset	Gets/Sets offset of the buffer
Name	Gets/Sets buffer name
Data Access Methods	
At	Gets buffer's element
Data Update Methods	
Refresh	Updates the buffer
RefreshCurrent	Updates only current value

Offset

Gets offset of the buffer.

```
int Offset() const
```

Returned value

Buffer offset.

Offset ()

Sets offset of the buffer.

```
void Offset(  
    int offset    // offset  
)
```

Parameters

offset

[in] New buffer offset.

Name

Gets the name of the buffer.

```
string Name() const
```

Returned value

Name of the buffer.

Name

Sets the name of the buffer.

```
void Name(  
    string name    // name  
)
```

Parameters

name

[in] New name of the buffer.

At

Gets buffer element by index.

```
long At(  
    int index    // index  
)
```

Parameters

index

[in] Index of buffer element.

Returned value

Buffer element with the specified index.

Refresh

Updates the whole buffer.

```
bool Refresh(  
    int handle,      // handle  
    int num          // buffer number  
)
```

Parameters

handle

[in] Handle of the indicator.

num

[in] Buffer index of the indicator.

Returned value

true - if successful, false - if the buffer hasn't been updated.

RefreshCurrent

Updates the current (zeroth) buffer element.

```
bool RefreshCurrent(  
    int handle,      // handle of the indicator  
    int num          // buffer number  
)
```

Parameters

handle

[in] Handle of the indicator.

num

[in] Buffer number.

Returned value

true - if successful, false - if the buffer hasn't been updated.

CSeries

CSeries is a base class for an access to the timeseries data of the Standard Library.

Description

The CSeries class provides the simplified access to MQL5 timeseries functions to all its descendants.

Declaration

```
class CSeries: public CArrayObj
```

Title

```
#include <Indicators\Series.mqh>
```

Class Methods

Attributes	
<u>Name</u>	Gets the name of timeseries or indicator
<u>BuffersTotal</u>	Gets the number of buffers of timeseries or indicator
<u>Timeframe</u>	Gets the timeframe flag of timeseries or indicator
<u>Symbol</u>	Gets the symbol of timeseries or indicator
<u>Period</u>	Gets the period of timeseries or indicator
<u>RefreshCurrent</u>	Gets/Sets the flag of updating the current data
Data Access Methods	
virtual <u>BufferResize</u>	Sets buffer size of timeseries or indicator
Data Update Methods	
virtual <u>Refresh</u>	Update the data of timeseries or indicator
<u>PeriodDescription</u>	Gets the period as a string

Name

Gets the name of timeseries or indicator

```
string Name() const
```

Returned value

The name of timeseries or indicator.

BuffersTotal

Gets the number of buffers of timeseries or indicator.

```
int BuffersTotal() const
```

Returned value

The number of buffers of timeseries or indicator.

Note

The timeseries has an only one buffer.

Timeframe

Gets the timeframe flag of timeseries or indicator.

```
int Timeframe() const
```

Returned value

The timeframe flag of timeseries or indicator.

Note

It's the visibility flag of some timeframes.

Symbol

Gets the symbol of timeseries or indicator.

```
string Symbol() const
```

Returned value

The symbol of timeseries or indicator.

Period

Gets the period of timeseries or indicator.

```
ENUM_TIMEFRAMES Period() const
```

Returned value

The period (value of [ENUM_TIMEFRAMES](#) enumeration) of timeseries or indicator.

RefreshCurrent

Sets a flag to update the current values of timeseries or indicator.

```
string RefreshCurrent(  
    bool flag    // new flag  
)
```

Parameters

flag

[in] New flag.

Returned value

None.

BufferResize

Sets buffer size of timeseries or indicator.

```
virtual void BufferResize(  
    int size    // new size  
)
```

Parameters

size

[in] New size of the buffers.

Note

All the buffers have the same buffer size.

Refresh

Updates the data of timeseries or indicator.

```
virtual void Refresh(  
    int flags    // flags  
)
```

Parameters

flags

[in] Timeframes to update (flag) .

PeriodDescription

Gets the string representation of the specified [ENUM_TIMEFRAMES](#) enumeration.

```
string PeriodDescription(  
    int val==0      // value  
) const
```

Parameters

val==0

[in] Value to convert.

Returned value

The string representation of the specified [ENUM_TIMEFRAMES](#) enumeration.

Note

If the value isn't specified or equal to zero, it returns the timeframe of timeseries or indicator.

CPriceSeries

CPriceSeries is a base class for access to the price data.

Description

The CSeries class provides the simplified access to MQL5 functions for working with price data to all its descendants.

Declaration

```
class CPriceSeries: public CSeries
```

Title

```
#include <Indicators\TimeSeries.mqh>
```

Class Methods

Create Methods	
virtual BufferResize	Sets size of the buffer
Data Access Methods	
virtual GetData	Gets the specified buffer element by index
Data Update Methods	
virtual Refresh	Updates timeseries data
Data Search Methods	
virtual MinIndex	Gets the index of minimal element in the specified range
virtual MinValue	Gets the value and index of minimal element in the specified range
virtual MaxIndex	Gets the index of maximal element in the specified range
virtual MaxValue	Gets the value and index of maximal element in the specified range

BufferResize

Sets new size of the buffer.

```
virtual void BufferResize(  
    int size    // new size  
)
```

Parameters

size

[in] New buffer size.

GetData

Gets the specified buffer element by index.

```
virtual double GetData(  
    int index    // index  
) const
```

Parameters

index

[in] Index of buffer element.

Returned value

The buffer element with the specified index or [EMPTY_VALUE](#).

Refresh

Updates the timeseries data

```
virtual void Refresh(  
    int flags    // timeframe flags  
)
```

Parameters

flags

[in] Timeframes to update (flag) .

MinIndex

Gets the index of minimal element in the specified range.

```
virtual int MinIndex(  
    int start,      // starting index  
    int count       // number of elements to scan  
) const
```

Parameters

start

[in] Starting index.

count

[in] Number of elements to proceed.

Returned value

The index of minimal element in the specified range, or -1 in the case of error.

MinValue

Gets the value and index of minimal element in the specified range.

```
virtual double MinValue(  
    int    start,      // starting index  
    int    count,      // number of elements to scan  
    int&   index       // reference to the variable for index  
) const
```

Parameters

start

[in] Starting index.

count

[in] Number of elements to proceed.

index

[out] Reference to the variable of integer type.

Returned value

The value of minimal element of the buffer in the specified range, or [EMPTY_VALUE](#) if error.

Note

The index of minimal element is stored in the variable index.

MaxIndex

Gets the index of maximal element in the specified range.

```
virtual int MaxIndex(  
    int start,      // starting index  
    int count       // number of elements to scan  
) const
```

Parameters

start

[in] Starting index.

count

[in] Number of elements to proceed.

Returned value

The index of the maximal element in the specified range, or -1 in the case of error.

MaxValue

Gets the value and index of maximal element in the specified range.

```
virtual double MaxValue(  
    int    start,      // starting index  
    int    count,      // number of elements to scan  
    int&   index       // reference to the variable for index  
) const
```

Parameters

start

[in] Starting index.

count

[in] Number of elements to proceed.

index

[out] Reference to the variable of integer type.

Returned value

The value of maximal element of the buffer in the specified range, or [EMPTY_VALUE](#) if error.

Note

The index of maximal element is stored in the variable index.

CIndicator

CIndicator is a base class for technical indicator classes of the standard MQL Library.

Description

The CIndicator class provides the simplified access for all of its descendants to MQL5 API technical indicator functions.

Declaration

```
class CIndicator: public CSeries
```

Title

```
#include <Indicators\Indicator.mqh>
```

Class Methods

Attributes	
Handle	Gets the indicator's handle.
Status	Gets the status of the indicator.
FullRelease	Sets a flag to release the handle of the indicator.
Creation	
Create	Creates the indicators
BufferResize	Sets new buffer size
Data Access Methods	
GetData	Copying the data from the indicator's buffer
Data Update Methods	
Refresh	Updates the indicator's data
Finding Min/Max Values	
Minimum	Gets the index of minimal element of the specified buffer in a specified range.
MinValue	Gets the value and index of minimal element of the specified buffer in a specified range.
Maximum	Gets the index of maximal element of the specified buffer in a specified range.
MaxValue	Gets the value and index of maximal element of the specified buffer in a specified range.
Conversion of Enumerations	

MethodDescription	Gets the value of ENUM__MA__METHOD enumeration as string
PriceDescription	Gets the value of ENUM__APPLIED__PRICE enumeration as string
VolumeDescription	Gets the value of ENUM__APPLIED__VOLUME enumeration as string
Working with chart	
AddToChart	Adds the indicator to the chart
DeleteFromChart	Deletes the indicator from the chart

Derived classes:

- [CiAC](#)
- [CiAD](#)
- [CiADX](#)
- [CiADXWilder](#)
- [CiAlligator](#)
- [CiAMA](#)
- [CiAO](#)
- [CiATR](#)
- [CiBands](#)
- [CiBearsPower](#)
- [CiBullsPower](#)
- [CiBWMFI](#)
- [CiCCI](#)
- [CiChaikin](#)
- [CiDEMA](#)
- [CiDeMarker](#)
- [CiEnvelopes](#)
- [CiForce](#)
- [CiFractals](#)
- [CiFrAMA](#)
- [CiGator](#)
- [CiIchimoku](#)
- [CiMA](#)
- [CiMACD](#)
- [CiMFI](#)
- [CiMomentum](#)
- [CiOBV](#)
- [CiOsMA](#)

- [CiRSI](#)
- [CiRVI](#)
- [CiSAR](#)
- [CiStdDev](#)
- [CiStochastic](#)
- [CiTEMA](#)
- [CiTriX](#)
- [CiVIDyA](#)
- [CiVolumes](#)
- [CiWPR](#)

Handle

Gets the indicator's handle.

```
int Handle() const
```

Returned value

Handle of the indicator.

Status

Gets the status of the indicator.

```
string Status() const
```

Returned value

The status of indicator creation.

FullRelease

Sets a flag to release the handle of the indicator.

```
void FullRelease(  
    bool flag    // flag  
)
```

Parameters

flag

[in] New value of the handle release flag.

Create

Creates the indicator with specified parameters.

```
bool Create(  
    string          symbol,          // symbol  
    ENUM_TIMEFRAMES period,          // period  
    ENUM_INDICATOR  type,            // type  
    int             num_params,      // number of parameters  
    MqlParam&       params           // reference to the parameters array  
)
```

Parameters

symbol

[in] Symbol name.

period

[in] Period ([ENUM_TIMEFRAMES](#) enumeration) .

type

[in] Indicator's type ([ENUM_INDICATOR](#) enumeration) .

num_params

[in] Number of indicator's parameters.

params

[in] Reference to the parameters array for the indicator.

Returned value

true if successful, false if indicator hasn't been created.

BufferResize

Sets the size of the indicator's buffer.

```
void BufferResize(  
    int size    // new size  
)
```

Parameters

size

[in] New buffer size.

Note

All the indicator's buffers have the same size.

GetData

Gets the specified element from the specified buffer of the indicator.

```
double  GetData (
    int   buffer_num,      // buffer number
    int   index            // element index
) const
```

Parameters

buffer_num
[in] Buffer number.

index
[in] Element index.

Returned value

If successful, it returns the numerical value of element, or [EMPTY_VALUE](#) in the case of error.

GetData

Gets the data from the indicator's buffer by starting position and number of necessary data.

```
int  GetData (
    int      start_pos,      // position
    int      count,          // number of elements needed
    int      buffer_num,     // buffer number
    double&  buffer          // target array for data
) const
```

Parameters

start_pos
[in] Starting position of the indicator's buffer.

count
[in] Number of elements needed.

buffer_num
[in] Number of the indicator's buffer.

buffer
[in] Referencet to the target array for the data.

Returned value

If successful, it returns the number of elements, received from the specified indicator buffer, otherwise -1.

GetData

Gets the data from the indicator's buffer by start time and number of necessary data.

```
int  GetData(
    datetime  start_time,      // starting time
    int       count,          // number of elements needed
    int       buffer_num,     // buffer number
    double&   buffer          // target array for data
) const
```

Parameters

start_time
[in] Starting time.

count
[in] Number of elements needed.

buffer_num
[in] Number of the indicator's buffer.

buffer
[in] Reference to the target array.

Returned value

If successful, it returns the number of elements, received from the specified indicator buffer, otherwise -1.

GetData

Gets the data from the indicator's buffer by start and stop time and number of necessary data.

```
int  GetData(
    datetime  start_time,      // start time
    datetime  stop_time,      // stop time
    int       buffer_num,     // number of buffer
    double&   buffer          // target array for data
) const
```

Parameters

start_time
[in] Starting time.

stop_time
[in] Stop time.

buffer_num
[in] Number of the indicator's buffer.

buffer
[in] Reference to the target array.

Returned value

If successful, it returns the number of elements, received from the specified indicator buffer, otherwise -1.

Refresh

Updates the indicator's data.

```
void Refresh(  
    int flags    // flags  
)
```

Parameters

flags

[in] Timeframe update flags.

Minimum

Returns the index of minimal element of the specified buffer in a specified range.

```
int Minimum(  
    int  buffer_num,      // buffer number  
    int  start,           // starting index  
    int  count            // number of elements to proceed  
) const
```

Parameters

buffer_num

[in] Buffer number to search the value in.

start

[in] Starting index of the search.

count

[in] Number of elements to search.

Returned value

Index of the minimal element of the specified buffer in a specified range.

MinValue

Returns the value and index of minimal element of the specified buffer in a specified range.

```
double MinValue(  
    int    buffer_num,      // buffer number  
    int    start,           // starting index  
    int    count,           // number of elements to proceed  
    int&   index            // reference  
) const
```

Parameters

buffer_num

[in] Buffer number to search the value in.

start

[in] Starting index.

count

[in] Number of elements to proceed.

index

[out] Reference to the variable of int type for the minimal element index.

Returned value

The value of the minimal element of the specified buffer in a specified range.

Note

The index of minimal buffer element is stored into the variable *index*, passed by reference.

Maximum

Returns the index of maximal element of the specified buffer in a specified range.

```
int Maximum(  
    int  buffer_num,      // buffer number  
    int  start,           // starting index  
    int  count            // number of elements to proceed  
) const
```

Parameters

buffer_num

[in] Buffer number to search the value in.

start

[in] Starting index of the search.

count

[in] Number of elements to search.

Returned value

Index of the maximal element of the specified buffer in a specified range.

MaxValue

Returns the value and index of maximal element of the specified buffer in a specified range.

```
double MaxValue(  
    int    buffer_num,      // buffer number  
    int    start,           // starting index  
    int    count,           // number of elements to proceed  
    int&   index            // reference  
) const
```

Parameters

buffer_num

[in] Buffer number to search the value in.

start

[in] Starting index.

count

[in] Number of elements to proceed.

index

[out] Reference to the variable of int type for the maximal element index.

Returned value

The value of the maximal element of the specified buffer in a specified range.

Note

The index of maximal buffer element is stored into the variable index, passed by reference.

MethodDescription

The function returns the value of [ENUM_MA_METHOD](#) enumeration as a string.

```
string MethodDescription(  
    int val==0      // value  
) const
```

Parameters

val==0

[in] Value of [ENUM_MA_METHOD](#) enumeration.

Returned value

The value of [ENUM_MA_METHOD](#) enumeration as a string.

PriceDescription

The method returns the value of [ENUM_APPLIED_PRICE](#) enumeration as a string.

```
string PriceDescription(  
    int val==0      // value  
) const
```

Parameters

val==0

[in] Value of [ENUM_APPLIED_PRICE](#) enumeration.

Returned value

The value of [ENUM_APPLIED_PRICE](#) enumeration as a string.

VolumeDescription

The method returns the value of [ENUM_APPLIED_VOLUME](#) enumeration as a string.

```
string VolumeDescription(  
    int val==0      // value  
) const
```

Parameters

val==0

[in] Value of [ENUM_APPLIED_VOLUME](#) enumeration.

Returned value

The value of [ENUM_APPLIED_VOLUME](#) enumeration as a string.

AddToChart

Adds the indicator to the chart.

```
bool AddToChart(  
    long chart,      // chart ID  
    int subwin       // chart subwindow  
)
```

Parameters

chart

[in] Chart ID.

subwin

[in] Chart subwindow.

Returned value

true - if successful, false if error.

DeleteFromChart

Deletes the indicator from the chart.

```
bool DeleteFromChart(  
    long chart,      // chart ID  
    int subwin       // chart subwindow  
)
```

Parameters

chart

[in] Chart ID.

subwin

[in] Chart subwindow.

Returned value

true - if successful, false if error.

CIndicators

The CIndicators is a class for collecting instances of timeseries and technical indicators classes.

Description

The CIndicators class provides creation of the technical indicators class instances, their storage and management (data synchronization, handle and memory management) .

Declaration

```
class CIndicators: public CArrayObj
```

Title

```
#include <Indicators\Indicators.mqh>
```

Class Methods

Create Methods	
Create	Creates technical indicator
Data Update Methods	
Refresh	Updates data for all technical indicators in the collection

Create

It creates the indicator with the specified parameters.

```
CIndicator* Create(  
    string          symbol,      // Symbol name  
    ENUM_TIMEFRAMES period,      // Period  
    ENUM_INDICATOR  type,        // Indicator's type  
    int             count,        // Number of parameters  
    MqlParam&       params       // Parameters array reference  
)
```

Parameters

symbol

[in] Symbol name.

period

[in] Period ([ENUM_TIMEFRAMES](#) enumeration) .

type

[in] Indicator's type ([ENUM_INDICATOR](#)) .

count

[in] Number of parameters for the indicator.

params

[in] Reference to the parameters array for the indicator.

Returned value

If successful, it returns the reference to the created indicator, and NULL if the indicator hasn't been created.

Refresh

Updates data for all technical indicators in the collection.

```
int Refresh()
```

Returned value

It returns the updated timeframe flags (formed as an object visibility flags) .

Timeseries classes

This group of chapters contains technical details of timeseries classes of the MQL5 Standard Library and descriptions of all its key components.

Class	Description
<u>CiSpread</u>	Provides an access to spread historical data
<u>CiTime</u>	Provides an access to open times of the bars in the history
<u>CiTickVolume</u>	Provides an access to tick volumes of the bars in the history
<u>CiRealVolume</u>	Provides an access to real volumes of the bars in the history
<u>CiOpen</u>	Provides an access to open prices of the bars in the history
<u>CiHigh</u>	Provides an access to high prices of the bars in the history
<u>CiLow</u>	Provides an access to low prices of the bars in the history
<u>CiClose</u>	Provides an access to close prices of the bars in the history

CiSpread

CiSpread is a class designed for access to spreads of the bars in the history.

Description

The CiSpread class provides an access to spread historical data.

Declaration

```
class CiSpread: public CSeries
```

Title

```
#include <Indicators\TimeSeries.mqh>
```

Class Methods

Create Methods	
Create	Creates a timeseries
BufferResize	Sets buffer size
Data Access Methods	
GetData	Gets the data
Data Update Methods	
Refresh	Updates the data

Create

Creates a timeseries with the specified parameters for access to the spreads history.

```
bool Create(  
    string      symbol,      // symbol  
    ENUM_TIMEFRAMES period    // period  
)
```

Parameters

symbol

[in] Symbol.

period

[in] Timeframe ([ENUM_TIMEFRAMES](#) enumeration) .

Returned value

true if successful, false if timeseries hasn't been created.

BufferResize

Sets new size of the series.

```
virtual void BufferResize(  
    int size    // new size  
)
```

Parameters

size

[in] New buffer size.

GetData

Gets the element of timeseries by index.

```
int  GetData(  
    int  index      // index  
    ) const
```

Parameters

index

[in] Index of the element needed.

Returned value

The timeseries buffer element, or 0.

GetData

Gets the element of timeseries by starting position and number of elements.

```
int  GetData(  
    int  start_pos,      // starting position  
    int  count,          // number of elements to get  
    int& buffer          // target array  
    ) const
```

Parameters

start_pos

[in] Starting position of timeseries.

count

[in] Number of elements needed.

buffer

[in] Reference to the target array for the data.

Returned value

>=0 if successful, -1 in the case of error.

GetData

Gets the element of timeseries by starting time and number of elements.

```
int  GetData(  
    datetime start_time, // starting time  
    int      count,      // number of elements  
    int&     buffer      // target array  
    ) const
```

Parameters

start_time

[in] Starting time.

count

[in] Number of elements needed.

buffer

[in] Reference to the target array for data.

Returned value

>=0 if successful, -1 in the case of error.

GetData

Gets the element of timeseries by starting and stop times.

```
int  GetData(  
    datetime  start_time,      // starting time  
    datetime  stop_time,       // stop time  
    int&      buffer           // target array  
) const
```

Parameters

start_time

[in] Starting time.

stop_time

[in] Stop time.

buffer

[in] Reference to the target array for data

Returned value

>=0 if successful, -1 in the case of error.

Refresh

Updates the data of timeseries.

```
virtual void Refresh(  
    int flags    // flags  
)
```

Parameters

flags

[in] Timeframe flags.

CiTime

CiTime is a class designed for access to open times of the bars in the history.

Description

The CiTime class provides an access to open times of the bars in the history.

Declaration

```
class CiTime: public CSeries
```

Title

```
#include <Indicators\TimeSeries.mqh>
```

Class Methods

Create Methods	
Create	Creates a timeseries
BufferResize	Sets buffer size
Data Access Methods	
GetData	Gets the data
Data Update Methods	
Refresh	Updates the data

Create

Creates a timeseries with the specified parameters for access to the opening times of the bars in the history.

```
bool Create(  
    string      symbol,      // symbol  
    ENUM_TIMEFRAMES period    // period  
)
```

Parameters

symbol

[in] Symbol.

period

[in] Timeframe ([ENUM_TIMEFRAMES](#) enumeration) .

Returned value

true if successful, false if timeseries hasn't been created.

BufferResize

Sets new size of the series.

```
virtual void BufferResize(  
    int size    // new size  
)
```

Parameters

size

[in] New buffer size.

GetData

Gets the element of timeseries by index.

```
datetime GetData(  
    int index      // index  
    ) const
```

Parameters

index

[in] Index of the element needed.

Returned value

The timeseries buffer element, or 0.

GetData

Gets the element of timeseries by starting position and number of elements.

```
int GetData(  
    int start_pos,    // starting position  
    int count,        // number of elements to get  
    long& buffer      // target array  
    ) const
```

Parameters

start_pos

[in] Starting position of timeseries.

count

[in] Number of elements needed.

buffer

[in] Reference to the target array for the data.

Returned value

>=0 if successful, -1 in the case of error.

GetData

Gets the element of timeseries by starting time and number of elements.

```
int GetData(  
    datetime start_time,    // starting time  
    int count,              // number of elements  
    long& buffer            // target array  
    ) const
```

Parameters

start_time

[in] Starting time.

count

[in] Number of elements needed.

buffer

[in] Reference to the target array for data.

Returned value

>=0 if successful, -1 in the case of error.

GetData

Gets the element of timeseries by starting and stop times.

```
int  GetData(  
    datetime  start_time,      // starting time  
    datetime  stop_time,       // stop time  
    long&     buffer           // target array  
) const
```

Parameters

start_time

[in] Starting time.

stop_time

[in] Stop time.

buffer

[in] Reference to the target array for data

Returned value

>=0 if successful, -1 in the case of error.

Refresh

Updates the data of timeseries.

```
virtual void Refresh(  
    int flags    // flags  
)
```

Parameters

flags

[in] Timeframe flags.

CiTickVolume

CiTickVolume is a class designed for access to tick volumes of the bars in the history.

Description

The CiTickVolume class provides an access to tick volumes of the bars in the history.

Declaration

```
class CiTickVolume: public CSeries
```

Title

```
#include <Indicators\TimeSeries.mqh>
```

Class Methods

Create Methods	
Create	Creates a timeseries
BufferResize	Sets buffer size
Data Access Methods	
GetData	Gets the data
Data Update Methods	
Refresh	Updates the data

Create

Creates a timeseries with the specified parameters for access to the tick volumes of the bars in the history.

```
bool Create(  
    string      symbol,      // symbol  
    ENUM_TIMEFRAMES period   // period  
)
```

Parameters

symbol

[in] Symbol.

period

[in] Timeframe ([ENUM_TIMEFRAMES](#) enumeration) .

Returned value

true if successful, false if timeseries hasn't been created.

BufferResize

Sets new size of the series.

```
virtual void BufferResize(  
    int size    // new size  
)
```

Parameters

size

[in] New buffer size.

GetData

Gets the element of timeseries by index.

```
datetime GetData(  
    int index      // index  
) const
```

Parameters

index

[in] Index of the element needed.

Returned value

The timeseries buffer element, or 0.

GetData

Gets the element of timeseries by starting position and number of elements.

```
int GetData(  
    int start_pos,      // starting position  
    int count,          // number of elements to get  
    long& buffer        // target array  
) const
```

Parameters

start_pos

[in] Starting position of timeseries.

count

[in] Number of elements needed.

buffer

[in] Reference to the target array for the data.

Returned value

>=0 if successful, -1 in the case of error.

GetData

Gets the element of timeseries by starting time and number of elements.

```
int GetData(  
    datetime start_time, // starting time  
    int count,           // number of elements  
    long& buffer         // target array  
) const
```

Parameters

start_time

[in] Starting time.

count

[in] Number of elements needed.

buffer

[in] Reference to the target array for data.

Returned value

>=0 if successful, -1 in the case of error.

GetData

Gets the element of timeseries by starting and stop times.

```
int  GetData(  
    datetime  start_time,      // starting time  
    datetime  stop_time,       // stop time  
    long&     buffer           // target array  
) const
```

Parameters

start_time

[in] Starting time.

stop_time

[in] Stop time.

buffer

[in] Reference to the target array for data

Returned value

>=0 if successful, -1 in the case of error.

Refresh

Updates the data of timeseries.

```
virtual void Refresh(  
    int flags    // flags  
)
```

Parameters

flags

[in] Timeframe flags.

CiRealVolume

CiRealVolume is a class designed for access to real volumes of the bars in the history.

Description

The CiRealVolume class provides an access to real volumes of the bars in the history.

Declaration

```
class CiRealVolume: public CSeries
```

Title

```
#include <Indicators\TimeSeries.mqh>
```

Class Methods

Create Methods	
Create	Creates a timeseries
BufferResize	Sets buffer size
Data Access Methods	
GetData	Gets the data
Data Update Methods	
Refresh	Updates the data

Create

Creates a timeseries with the specified parameters for access to the real volumes of the bars in the history.

```
bool Create(  
    string      symbol,      // symbol  
    ENUM_TIMEFRAMES period    // period  
)
```

Parameters

symbol

[in] Symbol.

period

[in] Timeframe ([ENUM_TIMEFRAMES](#) enumeration) .

Returned value

true if successful, false if timeseries hasn't been created.

BufferResize

Sets new size of the series.

```
virtual void BufferResize(  
    int size    // new size  
)
```

Parameters

size

[in] New buffer size.

GetData

Gets the element of timeseries by index.

```
datetime GetData(  
    int index      // index  
) const
```

Parameters

index

[in] Index of the element needed.

Returned value

The timeseries buffer element, or 0.

GetData

Gets the element of timeseries by starting position and number of elements.

```
int GetData(  
    int start_pos,    // starting position  
    int count,        // number of elements to get  
    long& buffer      // target array  
) const
```

Parameters

start_pos

[in] Starting position of timeseries.

count

[in] Number of elements needed.

buffer

[in] Reference to the target array for the data.

Returned value

>=0 if successful, -1 in the case of error.

GetData

Gets the element of timeseries by starting time and number of elements.

```
int GetData(  
    datetime start_time,    // starting time  
    int count,              // number of elements  
    long& buffer            // target array  
) const
```

Parameters

start_time

[in] Starting time.

count

[in] Number of elements needed.

buffer

[in] Reference to the target array for data.

Returned value

>=0 if successful, -1 in the case of error.

GetData

Gets the element of timeseries by starting and stop times.

```
int  GetData(  
    datetime  start_time,      // starting time  
    datetime  stop_time,       // stop time  
    long&     buffer           // target array  
) const
```

Parameters

start_time

[in] Starting time.

stop_time

[in] Stop time.

buffer

[in] Reference to the target array for data

Returned value

>=0 if successful, -1 in the case of error.

Refresh

Updates the data of timeseries.

```
virtual void Refresh(  
    int flags    // flags  
)
```

Parameters

flags

[in] Timeframe flags.

CiOpen

CiOpen is a class designed for access to open prices of the bars in the history.

Description

The CiOpen class provides an access to open prices of the bars in the history.

Declaration

```
class CiOpen: public CSeries
```

Title

```
#include <Indicators\TimeSeries.mqh>
```

Class Methods

Create Methods	
Create	Creates a timeseries
Data Access Methods	
GetData	Gets the data

Create

Creates a timeseries with the specified parameters for access to the open prices of the bars in the history.

```
bool Create(  
    string      symbol,      // symbol  
    ENUM_TIMEFRAMES period   // period  
)
```

Parameters

symbol

[in] Symbol.

period

[in] Timeframe ([ENUM_TIMEFRAMES](#) enumeration) .

Returned value

true if successful, false if timeseries hasn't been created.

GetData

Gets the element of timeseries by starting position and number of elements.

```
int  GetData(  
    int    start_pos,      // starting position  
    int    count,          // number of elements to get  
    double& buffer         // target array  
) const
```

Parameters

start_pos

[in] Starting position of timeseries.

count

[in] Number of elements needed.

buffer

[in] Reference to the target array for the data.

Returned value

>=0 if successful, -1 in the case of error.

GetData

Gets the element of timeseries by starting time and number of elements.

```
int  GetData(  
    datetime start_time,   // starting time  
    int      count,        // number of elements  
    double&  buffer        // target array  
) const
```

Parameters

start_time

[in] Starting time.

count

[in] Number of elements needed.

buffer

[in] Reference to the target array for data.

Returned value

>=0 if successful, -1 in the case of error.

GetData

Gets the element of timeseries by starting and stop times.

```
int  GetData(  
    datetime  start_time,      // starting time  
    datetime  stop_time,       // stop time  
    double&    buffer          // target array  
) const
```

Parameters

start_time

[in] Starting time.

stop_time

[in] Stop time.

buffer

[in] Reference to the target array for data

Returned value

>=0 if successful, -1 in the case of error.

CiHigh

CiHigh is a class designed for access to high prices of the bars in the history.

Description

The CiHigh class provides an access to high prices of the bars in the history.

Declaration

```
class CiHigh: public CSeries
```

Title

```
#include <Indicators\TimeSeries.mqh>
```

Class Methods

Create Methods	
Create	Creates a timeseries
Data Access Methods	
GetData	Gets the data

Create

Creates a timeseries with the specified parameters for access to the high prices of the bars in the history.

```
bool Create(  
    string      symbol,      // symbol  
    ENUM_TIMEFRAMES period    // period  
)
```

Parameters

symbol

[in] Symbol.

period

[in] Timeframe ([ENUM_TIMEFRAMES](#) enumeration) .

Returned value

true if successful, false if timeseries hasn't been created.

GetData

Gets the element of timeseries by starting position and number of elements.

```
int  GetData(  
    int    start_pos,      // starting position  
    int    count,          // number of elements to get  
    double& buffer         // target array  
) const
```

Parameters

start_pos

[in] Starting position of timeseries.

count

[in] Number of elements needed.

buffer

[in] Reference to the target array for the data.

Returned value

>=0 if successful, -1 in the case of error.

GetData

Gets the element of timeseries by starting time and number of elements.

```
int  GetData(  
    datetime start_time,   // starting time  
    int      count,        // number of elements  
    double&  buffer        // target array  
) const
```

Parameters

start_time

[in] Starting time.

count

[in] Number of elements needed.

buffer

[in] Reference to the target array for data.

Returned value

>=0 if successful, -1 in the case of error.

GetData

Gets the element of timeseries by starting and stop times.

```
int  GetData(  
    datetime  start_time,      // starting time  
    datetime  stop_time,       // stop time  
    double&    buffer          // target array  
) const
```

Parameters

start_time

[in] Starting time.

stop_time

[in] Stop time.

buffer

[in] Reference to the target array for data

Returned value

>=0 if successful, -1 in the case of error.

CiLow

CiLow is a class designed for access to low prices of the bars in the history.

Description

The CiLow class provides an access to low prices of the bars in the history.

Declaration

```
class CiLow: public CSeries
```

Title

```
#include <Indicators\TimeSeries.mqh>
```

Class Methods

Create Methods	
Create	Creates a timeseries
Data Access Methods	
GetData	Gets the data

Create

Creates a timeseries with the specified parameters for access to the low prices of the bars in the history.

```
bool Create(  
    string      symbol,      // symbol  
    ENUM_TIMEFRAMES period    // period  
)
```

Parameters

symbol

[in] Symbol.

period

[in] Timeframe ([ENUM_TIMEFRAMES](#) enumeration) .

Returned value

true if successful, false if timeseries hasn't been created.

GetData

Gets the element of timeseries by starting position and number of elements.

```
int  GetData(  
    int    start_pos,      // starting position  
    int    count,          // number of elements to get  
    double& buffer         // target array  
) const
```

Parameters

start_pos

[in] Starting position of timeseries.

count

[in] Number of elements needed.

buffer

[in] Reference to the target array for the data.

Returned value

>=0 if successful, -1 in the case of error.

GetData

Gets the element of timeseries by starting time and number of elements.

```
int  GetData(  
    datetime start_time,   // starting time  
    int      count,        // number of elements  
    double&  buffer        // target array  
) const
```

Parameters

start_time

[in] Starting time.

count

[in] Number of elements needed.

buffer

[in] Reference to the target array for data.

Returned value

>=0 if successful, -1 in the case of error.

GetData

Gets the element of timeseries by starting and stop times.

```
int  GetData(  
    datetime  start_time,      // starting time  
    datetime  stop_time,       // stop time  
    double&    buffer          // target array  
) const
```

Parameters

start_time

[in] Starting time.

stop_time

[in] Stop time.

buffer

[in] Reference to the target array for data

Returned value

>=0 if successful, -1 in the case of error.

CiClose

CiClose is a class designed for access to close prices of the bars in the history.

Description

The CiClose class provides an access to close prices of the bars in the history.

Declaration

```
class CiClose: public CSeries
```

Title

```
#include <Indicators\TimeSeries.mqh>
```

Class Methods

Create Methods	
Create	Creates a timeseries
Data Access Methods	
GetData	Gets the data

Create

Creates a timeseries with the specified parameters for access to the closing prices of the bars in the history.

```
bool Create(  
    string          symbol,      // symbol  
    ENUM_TIMEFRAMES period      // period  
)
```

Parameters

symbol

[in] Symbol.

period

[in] Timeframe ([ENUM_TIMEFRAMES](#) enumeration) .

Returned value

true if successful, false if timeseries hasn't been created.

GetData

Gets the element of timeseries by starting position and number of elements.

```
int  GetData(  
    int    start_pos,      // starting position  
    int    count,          // number of elements to get  
    double& buffer         // target array  
) const
```

Parameters

start_pos

[in] Starting position of timeseries.

count

[in] Number of elements needed.

buffer

[in] Reference to the target array for the data.

Returned value

>=0 if successful, -1 in the case of error.

GetData

Gets the element of timeseries by starting time and number of elements.

```
int  GetData(  
    datetime start_time,   // starting time  
    int      count,        // number of elements  
    double&   buffer       // target array  
) const
```

Parameters

start_time

[in] Starting time.

count

[in] Number of elements needed.

buffer

[in] Reference to the target array for data.

Returned value

>=0 if successful, -1 in the case of error.

GetData

Gets the element of timeseries by starting and stop times.

```
int  GetData(  
    datetime  start_time,      // starting time  
    datetime  stop_time,       // stop time  
    double&    buffer          // target array  
) const
```

Parameters

start_time

[in] Starting time.

stop_time

[in] Stop time.

buffer

[in] Reference to the target array for data

Returned value

>=0 if successful, -1 in the case of error.

Trend Indicator Classes

This group of chapters contains technical details of Trend Indicator classes of the MQL5 Standard Library and descriptions of all its key components.

Class/group	Description
CiADX	Average Directional Index
CiADXWilder	Average Directional Index by Welles Wilder
CiBands	Bollinger Bands
CiEnvelopes	Envelopes
CiIchimoku	Ichimoku Kinko Hyo
CiMA	Moving Average
CiSAR	Parabolic Stop And Reverse System
CiStdDev	Standard Deviation
CiDEMA	Double Exponential Moving Average
CiTEMA	Triple Exponential Moving Average
CiFrAMA	Fractal Adaptive Moving Average
CiAMA	Adaptive Moving Average
CiVIDyA	Variable Index DYnamic Average

CiADX

CiADX is a class intended for using the Average Directional Index technical indicator.

Description

The CiADX class provides the creation and access to the data of the Average Directional Index indicator.

Declaration

```
class CiADX: public CIndicator
```

Title

```
#include <Indicators\Trend.mqh>
```

Class Methods

Attributes	
MaPeriod	Returns the averaging period
Create Methods	
Create	Creates the indicator
Data Access Methods	
Main	Returns the buffer element of the main line
Plus	Returns the buffer element of the +DI line
Minus	Returns the buffer element of the -DI line
Input/output	
virtual Type	Returns the object type identifier

MaPeriod

Returns the averaging period.

```
int MaPeriod() const
```

Returned value

Returns the averaging period, defined at the indicator creation.

Create

It creates the indicator with specified parameters.

```
bool Create(  
    string          symbol,          // Symbol  
    ENUM_TIMEFRAMES period,          // Period  
    int             ma_period        // Averaging period  
)
```

Parameters

symbol

[in] Symbol.

period

[in] Timeframe ([ENUM_TIMEFRAMES](#) enumeration) .

ma_period

[in] Averaging period.

Returned value

true if successful, false if indicator hasn't been created.

Main

Returns the buffer element by the specified index.

```
double Main(  
    int index    // Index  
)
```

Parameters

index

[in] Element index.

Returned value

Buffer element of the specified index if successful, or [EMPTY_VALUE](#) if there isn't any correct data.

Plus

Returns the buffer element of the +DI line by the specified index.

```
double Plus(  
    int index    // Index  
)
```

Parameters

index

[in] Element index.

Returned value

The buffer element of the +DI line of the specified index, or [EMPTY_VALUE](#) if there isn't any correct data.

Minus

Returns the buffer element of the -DI line by the specified index.

```
double Minus (  
    int index    // Index  
)
```

Parameters

index

[in] Element index.

Returned value

The buffer element of the -DI line of the specified index, or EMPTY_VALUE if there isn't any correct data.

Type

Returns the object type identifier.

```
virtual int Type() const
```

Returned value

Object type identifier ([IND_ADX](#) for CiADX) .

CiADXWilder

CiADXWilder is a class intended for using the Average Directional Index by Welles Wilder technical indicator.

Description

The CiADXWilder class provides the creation and access to the data of the Average Directional Index by Welles Wilder indicator.

Declaration

```
class CiADXWilder: public CIndicator
```

Title

```
#include <Indicators\Trend.mqh>
```

Class Methods

Attributes	
MaPeriod	Returns the averaging period
Create Methods	
Create	Creates the indicator
Data Access Methods	
Main	Returns the buffer element of the main line
Plus	Returns the buffer element of the +DI line
Minus	Returns the buffer element of the -DI line
Input/output	
virtual Type	Returns the object type identifier

MaPeriod

Returns the averaging period.

```
int MaPeriod() const
```

Returned value

Returns the averaging period, defined at the indicator creation.

Create

It creates the indicator with specified parameters.

```
bool Create(  
    string          symbol,          // Symbol  
    ENUM_TIMEFRAMES period,          // Period  
    int             ma_period        // Averaging period  
)
```

Parameters

symbol

[in] Symbol.

period

[in] Timeframe ([ENUM_TIMEFRAMES](#) enumeration) .

ma_period

[in] Averaging period.

Returned value

true if successful, false if indicator hasn't been created.

Main

Returns the buffer element by the specified index.

```
double Main(  
    int index    // Index  
)
```

Parameters

index

[in] Element index.

Returned value

Buffer element of the specified index if successful, or [EMPTY_VALUE](#) if there isn't any correct data.

Plus

Returns the buffer element of the +DI line by the specified index.

```
double Plus(  
    int index    // Index  
)
```

Parameters

index

[in] Element index.

Returned value

The buffer element of the +DI line of the specified index, or [EMPTY_VALUE](#) if there isn't any correct data.

Minus

Returns the buffer element of the -DI line by the specified index.

```
double Minus (  
    int index    // Index  
)
```

Parameters

index

[in] Element index.

Returned value

The buffer element of the -DI line of the specified index, or [EMPTY_VALUE](#) if there isn't any correct data.

Type

Returns the object type identifier.

```
virtual int Type() const
```

Returned value

Object type identifier ([IND_ADXW](#) for CiADXWilder) .

CiBands

CiBands is a class intended for using the Bollinger Bands technical indicator.

Description

The CiBands class provides the creation and access to the data of the Bollinger Bands indicator.

Declaration

```
class CiBands: public CIndicator
```

Title

```
#include <Indicators\Trend.mqh>
```

Class Methods

Attributes	
MaPeriod	Returns the averaging period
MaShift	Returns the horizontal shift
Deviation	Returns the deviation
Applied	Returns the price type or handle to apply
Create Methods	
Create	Creates the indicator
Data Access Methods	
Main	Returns the buffer element of the base line
Upper	Returns the buffer element of the upper line
Lower	Returns the buffer element of the lower line
Input/output	
virtual Type	Returns the object type identifier

MaPeriod

Returns the averaging period.

```
int MaPeriod() const
```

Returned value

Returns the averaging period, defined at the indicator creation.

MaShift

Returns the horizontal shift of the indicator.

```
int MaShift() const
```

Returned value

Returns the horizontal shift value, defined at the indicator creation.

Deviation

Returns the deviation.

```
double Deviation() const
```

Returned value

Returns the deviation, defined at the indicator creation.

Applied

Returns the price type or handle to apply.

```
int Applied() const
```

Returned value

Price type or handle to apply, defined at the indicator creation.

Create

It creates the indicator with specified parameters.

```
bool Create(  
    string          symbol,          // Symbol  
    ENUM_TIMEFRAMES period,          // Period  
    int             ma_period,       // Averaging period  
    int             ma_shift,        // Shift  
    double          deviation,       // Deviation  
    int             applied           // applied price, or handle  
)
```

Parameters

symbol

[in] Symbol.

period

[in] Timeframe ([ENUM_TIMEFRAMES](#) enumeration) .

ma_period

[in] Averaging period.

ma_shift

[in] Horizontal shift of the indicator.

deviation

[in] Deviation.

applied

[in] Volume type to apply.

Returned value

true if successful, false if indicator hasn't been created.

Base

Returns the buffer element of the base line by the specified index.

```
double Base(  
    int index    // Index  
)
```

Parameters

index

[in] Element index.

Returned value

The buffer element of the Base line of the specified index, or [EMPTY_VALUE](#) if there isn't any correct data.

Upper

Returns the buffer element of the upper line by the specified index.

```
double Upper (  
    int index    // Index  
)
```

Parameters

index

[in] Element index.

Returned value

The buffer element of the upper line of the specified index, or [EMPTY_VALUE](#) if there isn't any correct data.

Lower

Returns the buffer element of the lower line by the specified index.

```
double Lower(  
    int index    // Index  
)
```

Parameters

index

[in] Element index.

Returned value

The buffer element of the lower line of the specified index, or [EMPTY_VALUE](#) if there isn't any correct data.

Type

Returns the object type identifier.

```
virtual int Type() const
```

Returned value

Object type identifier ([IND_BANDS](#) for CiBands) .

CiEnvelopes

CiEnvelopes is a class intended for using the Envelopes technical indicator.

Description

The CiEnvelopes class provides the creation and access to the data of the Envelopes indicator.

Declaration

```
class CiEnvelopes: public CIndicator
```

Title

```
#include <Indicators\Trend.mqh>
```

Class Methods

Attributes	
MaPeriod	Returns the averaging period
MaShift	Returns the horizontal shift
MaMethod	Returns the averaging method
Deviation	Returns the deviation
Applied	Returns the price type or handle to apply
Create Methods	
Create	Creates the indicator
Data Access Methods	
Upper	Returns the buffer element of the upper line
Lower	Returns the buffer element of the lower line
Input/output	
virtual Type	Returns the object type identifier

MaPeriod

Returns the averaging period.

```
int MaPeriod() const
```

Returned value

Returns the averaging period, defined at the indicator creation.

MaShift

Returns the horizontal shift of the indicator.

```
int MaShift() const
```

Returned value

Returns the horizontal shift value, defined at the indicator creation.

MaMethod

Returns the averaging method.

```
ENUM_MA_METHOD MaMethod() const
```

Returned value

Returns the averaging method, defined at the indicator creation.

Deviation

Returns the value of deviation.

```
double Deviation() const
```

Returned value

Returns the value of deviation, defined at the indicator creation.

Applied

Returns the price type or handle to apply.

```
int Applied() const
```

Returned value

Price type or handle to apply, defined at the indicator creation.

Create

It creates the indicator with specified parameters.

```
bool Create(  
    string          symbol,          // Symbol  
    ENUM_TIMEFRAMES period,          // Period  
    int             ma_period,       // Averaging period  
    int             ma_shift,        // Horizontal shift  
    ENUM_MA_METHOD  ma_method,       // Averaging method  
    int             applied,         // Price type or handle to apply  
    double          deviation        // Deviation  
)
```

Parameters

symbol

[in] Symbol.

period

[in] Timeframe ([ENUM_TIMEFRAMES](#) enumeration) .

ma_period

[in] Averaging period.

ma_shift

[in] Horizontal shift.

ma_method

[in] Averaging method ([ENUM_MA_METHOD](#) enumeration) .

applied

[in] Price type of handle to apply.

deviation

[in] Deviation.

Returned value

true if successful, false if indicator hasn't been created.

Upper

Returns the buffer element of the upper line by the specified index.

```
double Upper (  
    int index    // Index  
)
```

Parameters

index

[in] Element index.

Returned value

The buffer element of the upper line of the specified index, or [EMPTY_VALUE](#) if there isn't any correct data.

Lower

Returns the buffer element of the lower line by the specified index.

```
double Lower (  
    int index    // Index  
)
```

Parameters

index

[in] Element index.

Returned value

The buffer element of the lower line of the specified index, or [EMPTY_VALUE](#) if there isn't any correct data.

Type

Returns the object type identifier.

```
virtual int Type() const
```

Returned value

Object type identifier ([IND_ENVELOPES](#) for CiEnvelopes) .

Cilchimoku

Cilchimoku is a class intended for using the Ichimoku Kinko Hyo technical indicator.

Description

The Cilchimoku class provides the creation, setup and access to the data of the Ichimoku Kinko Hyo indicator.

Declaration

```
class CiIchimoku: public CIndicator
```

Title

```
#include <Indicators\Trend.mqh>
```

Class Methods

Attributes	
TenkanSenPeriod	Returns the TenkanSen period
KijunSenPeriod	Returns the KijunSen period
SenkouSpanBPeriod	Returns the SenkouSpanB period
Create Methods	
Create	Creates the indicator
Data Access Methods	
TenkanSen	Returns the buffer element of the TenkanSen line
KijunSen	Returns the buffer element of the KijunSen line
SenkouSpanA	Returns the buffer element of the SenkouSpanA line
SenkouSpanB	Returns the buffer element of the SenkouSpanB line
ChinkouSpan	Returns the buffer element of the ChinkouSpan line
Input/output	
virtual Type	Returns the object type identifier

TenkanSenPeriod

Returns the TenkanSen period.

```
int TenkanSenPeriod() const
```

Returned value

Returns the TenkanSen period, defined at the indicator creation.

KijunSenPeriod

Returns the KijunSen period.

```
int KijunSenPeriod() const
```

Returned value

Returns the KijunSen period, defined at the indicator creation.

SenkouSpanBPeriod

Returns the SenkouSpanB period.

```
int SenkouSpanBPeriod() const
```

Returned value

Returns the SenkouSpanB period, defined at the indicator creation.

Create

It creates the indicator with specified parameters.

```
bool Create(  
    string          symbol,           // Symbol  
    ENUM_TIMEFRAMES period,          // Period  
    int             tenkan_sen,       // Period of TenkanSen  
    int             kijun_sen,        // Period of KijunSen  
    int             senkou_span_b     // Period of SenkouSpanB  
)
```

Parameters

symbol

[in] Symbol.

period

[in] Timeframe ([ENUM_TIMEFRAMES](#) enumeration) .

tenkan_sen

[in] Period of TenkanSen.

kijun_sen

[in] Period of KijunSen.

senkou_span_b

[in] Period of SenkouSpanB.

Returned value

true if successful, false if indicator hasn't been created.

TenkanSen

Returns the buffer element of the TenkanSen line by the specified index.

```
double TenkanSen(  
    int index    // Index  
)
```

Parameters

index

[in] Element index.

Returned value

The buffer element of the TenkanSen line of the specified index, or [EMPTY_VALUE](#) if there isn't any correct data.

KijunSen

Returns the buffer element of the KijunSen line by the specified index.

```
double KijunSen(  
    int index    // Index  
)
```

Parameters

index

[in] Element index.

Returned value

The buffer element of the KijunSen line of the specified index, or [EMPTY_VALUE](#) if there isn't any correct data.

SenkouSpanA

Returns the buffer element of the SenkouSpanA line by the specified index.

```
double SenkouSpanA(  
    int index    // Index  
)
```

Parameters

index

[in] Element index.

Returned value

The buffer element of the SenkouSpanA line of the specified index, or [EMPTY_VALUE](#) if there isn't any correct data.

SenkouSpanB

Returns the buffer element of the SenkouSpanB line by the specified index.

```
double SenkouSpanB(  
    int index    // Index  
)
```

Parameters

index

[in] Element index.

Returned value

The buffer element of the SenkouSpanB line of the specified index, or [EMPTY_VALUE](#) if there isn't any correct data.

ChinkouSpan

Returns the buffer element of the ChinkouSpan line by the specified index.

```
double ChinkouSpan(  
    int index    // Index  
)
```

Parameters

index

[in] Element index.

Returned value

The buffer element of the ChinkouSpan line of the specified index, or [EMPTY_VALUE](#) if there isn't any correct data.

Type

Returns the object type identifier.

```
virtual int Type() const
```

Returned value

Object type identifier ([IND_ICHIMOKU](#) for Cilchimoku) .

CiMA

CiMA is a class intended for using the Moving Average technical indicator.

Description

The CiMA class provides the creation, setup and access to the data of the Moving Average indicator.

Declaration

```
class CiMA: public CIndicator
```

Title

```
#include <Indicators\Trend.mqh>
```

Class Methods

Attributes	
MaPeriod	Returns the averaging period
MaShift	Returns the horizontal shift
MaMethod	Returns the averaging method
Applied	Returns the price type or handle to apply
Create Methods	
Create	Creates the indicator
Data Access Methods	
Main	Returns the buffer element
Input/output	
virtual Type	Returns the object type identifier

MaPeriod

Returns the averaging period.

```
int MaPeriod() const
```

Returned value

Returns the averaging period, defined at the indicator creation.

MaShift

Returns the horizontal shift of the indicator.

```
int MaShift() const
```

Returned value

Returns the horizontal shift value, defined at the indicator creation.

MaMethod

Returns the averaging method.

```
ENUM_MA_METHOD MaMethod() const
```

Returned value

Returns the averaging method (value of [ENUM_MA_METHOD](#) enumeration) , defined at the indicator creation.

Applied

Returns the price type or handle to apply.

```
int Applied() const
```

Returned value

Price type or handle to apply, defined at the indicator creation.

Create

It creates the indicator with specified parameters.

```
bool Create(  
    string          symbol,           // Symbol  
    ENUM_TIMEFRAMES period,          // Period  
    int             ma_period,        // Averaging period  
    int             ma_shift,         // Horizontal shift  
    ENUM_MA_METHOD  ma_method,        // Averaging method  
    int             applied            // Price type of handle to apply  
)
```

Parameters

string

[in] Symbol.

period

[in] Timeframe ([ENUM_TIMEFRAMES](#) enumeration) .

ma_period

[in] Averaging period.

ma_shift

[in] Horizontal shift.

ma_method

[in] Averaging method ([ENUM_MA_METHOD](#) enumeration) .

applied

[in] Price type or handle to apply.

Returned value

true if successful, false if indicator hasn't been created.

Main

Returns the buffer element by the specified index.

```
double Main(  
    int index    // Index  
)
```

Parameters

index

[in] Element index.

Returned value

Buffer element of the specified index if successful, or [EMPTY_VALUE](#) if there isn't any correct data.

Type

Returns the object type identifier.

```
virtual int Type() const
```

Returned value

Object type identifier ([IND_MA](#) for CiMA) .

CiSAR

CiSAR is a class intended for using the Parabolic Stop And Reverse System technical indicator.

Description

The CiSAR class provides the creation, setup and access to the data of the Parabolic Stop And Reverse System indicator.

Declaration

```
class CiSAR: public CIndicator
```

Title

```
#include <Indicators\Trend.mqh>
```

Class Methods

Attributes	
SarStep	Returns the step for the velocity increasing
Maximum	Returns the coefficient of price following
Create Methods	
Create	Creates the indicator
Data Access Methods	
Main	Returns the buffer element
Input/output	
virtual Type	Returns the object type identifier

SarStep

Returns the step for the velocity increasing (acceleration coefficient) .

```
double SarStep() const
```

Returned value

The step for the velocity increasing, defined at the indicator creation.

Maximum

Returns the coefficient of price following.

```
double Maximum() const
```

Returned value

The price following coefficient, defined at the indicator creation.

Create

It creates the indicator with specified parameters.

```
bool Create(  
    string          symbol,      // Symbol  
    ENUM_TIMEFRAMES period,      // Period  
    double          step,        // Step  
    double          maximum      // Coefficient  
)
```

Parameters

symbol

[in] Symbol.

period

[in] Timeframe ([ENUM_TIMEFRAMES](#) enumeration) .

step

[in] Step for the velocity increasing.

maximum

[in] Price following coefficient.

Returned value

true if successful, false if indicator hasn't been created.

Main

Returns the buffer element by the specified index.

```
double Main(  
    int index    // Index  
)
```

Parameters

index

[in] Element index.

Returned value

Buffer element of the specified index if successful, or [EMPTY_VALUE](#) if there isn't any correct data.

Type

Returns the object type identifier.

```
virtual int Type() const
```

Returned value

Object type identifier ([IND_SAR](#) for CiSAR) .

CiStdDev

CiStdDev is a class intended for using the Standard Deviation technical indicator.

Description

The CiStdDev class provides the creation, setup and access to the data of the Standard Deviation indicator.

Declaration

```
class CiStdDev: public CIndicator
```

Title

```
#include <Indicators\Trend.mqh>
```

Class Methods

Attributes	
MaPeriod	Returns the averaging period
MaShift	Returns the horizontal shift
MaMethod	Returns the averaging method
Applied	Returns the price type or handle to apply
Create Methods	
Create	Creates the indicator
Data Access Methods	
Main	Returns the buffer element
Input/output	
virtual Type	Returns the object type identifier

MaPeriod

Returns the averaging period.

```
int MaPeriod() const
```

Returned value

Returns the averaging period, defined at the indicator creation.

MaShift

Returns the horizontal shift of the indicator.

```
int MaShift() const
```

Returned value

Returns the horizontal shift value, defined at the indicator creation.

MaMethod

Returns the averaging method.

```
ENUM_MA_METHOD MaMethod() const
```

Returned value

Returns the averaging method (value of [ENUM_MA_METHOD](#) enumeration) , defined at the indicator creation.

Applied

Returns the price type or handle to apply.

```
int Applied() const
```

Returned value

Price type or handle to apply, defined at the indicator creation.

Create

It creates the indicator with specified parameters.

```
bool Create(  
    string          symbol,          // Symbol  
    ENUM_TIMEFRAMES period,          // Period  
    int             ma_period,       // Averaging period  
    int             ma_shift,        // Horizontal shift  
    ENUM_MA_METHOD  ma_method,       // Averaging method  
    int             applied           // Price type or handle to apply  
)
```

Parameters

symbol

[in] Symbol.

period

[in] Timeframe ([ENUM_TIMEFRAMES](#) enumeration) .

ma_period

[in] Averaging period.

ma_shift

[in] Horizontal shift.

ma_method

[in] Averaging method ([ENUM_MA_METHOD](#) enumeration) .

applied

[in] Price type or handle to apply.

Returned value

true if successful, false if indicator hasn't been created.

Main

Returns the buffer element by the specified index.

```
double Main(  
    int index    // Index  
)
```

Parameters

index

[in] Element index.

Returned value

Buffer element of the specified index if successful, or [EMPTY_VALUE](#) if there isn't any correct data.

Type

Returns the object type identifier.

```
virtual int Type() const
```

Returned value

Object type identifier ([IND_STDDEV](#) for CiStdDev) .

CiDEMA

CiDEMA is a class intended for using the Double Exponential Moving Average technical indicator.

Description

The CiDEMA class provides the creation, setup and access to the data of the Double Exponential Moving Average indicator.

Declaration

```
class CiDEMA: public CIndicator
```

Title

```
#include <Indicators\Trend.mqh>
```

Class Methods

Attributes	
MaPeriod	Returns the averaging period
IndShift	Returns the horizontal shift
Applied	Returns the price type or handle to apply
Create Methods	
Create	Creates the indicator
Data Access Methods	
Main	Returns the buffer element
Input/output	
virtual Type	Returns the object type identifier

MaPeriod

Returns the averaging period.

```
int MaPeriod() const
```

Returned value

Returns the averaging period, defined at the indicator creation.

IndShift

Returns the horizontal shift of the indicator.

```
int IndShift() const
```

Returned value

Returns the horizontal shift value, defined at the indicator creation.

Applied

Returns the price type or handle to apply.

```
int Applied() const
```

Returned value

Price type or handle to apply, defined at the indicator creation.

Create

It creates the indicator with specified parameters.

```
bool Create(  
    string          string,          // Symbol  
    ENUM_TIMEFRAMES period,          // Period  
    int             ma_period,        // Averaging period  
    int             ind_shift,        // Shift  
    int             applied           // Price type of handle to apply  
)
```

Parameters

string

[in] Symbol.

period

[in] Timeframe ([ENUM_TIMEFRAMES](#) enumeration) .

ma_period

[in] Averaging period.

ind_shift

[in] Horizontal shift.

applied

[in] Price type or handle to apply.

Returned value

true if successful, false if indicator hasn't been created.

Main

Returns the buffer element by the specified index.

```
double Main(  
    int index    // Index  
)
```

Parameters

index

[in] Element index.

Returned value

Buffer element of the specified index if successful, or [EMPTY_VALUE](#) if there isn't any correct data.

Type

Returns the object type identifier.

```
virtual int Type() const
```

Returned value

Object type identifier ([IND_DEMA](#) for CiDEMA) .

CiTEMA

CiTEMA is a class intended for using the Triple Exponential Moving Average technical indicator.

Description

The CiTEMA class provides the creation, setup and access to the data of the Triple Exponential Moving Average indicator.

Declaration

```
class CiTEMA: public CIndicator
```

Title

```
#include <Indicators\Trend.mqh>
```

Class Methods

Attributes	
MaPeriod	Returns the averaging period
IndShift	Returns the horizontal shift
Applied	Returns the price type or handle to apply
Create Methods	
Create	Creates the indicator
Data Access Methods	
Main	Returns the buffer element
Input/output	
virtual Type	Returns the object type identifier

MaPeriod

Returns the averaging period.

```
int MaPeriod() const
```

Returned value

Returns the averaging period, defined at the indicator creation.

IndShift

Returns the horizontal shift of the indicator.

```
int IndShift() const
```

Returned value

Returns the horizontal shift value, defined at the indicator creation.

Applied

Returns the price type or handle to apply.

```
int Applied() const
```

Returned value

Price type or handle to apply, defined at the indicator creation.

Create

It creates the indicator with specified parameters.

```
bool Create(  
    string          symbol,          // Symbol  
    ENUM_TIMEFRAMES period,          // Period  
    int             ma_period,       // Averaging period  
    int             ma_shift,        // Offset  
    int             applied          // Price type of handle to apply  
)
```

Parameters

symbol

[in] Symbol.

period

[in] Timeframe ([ENUM_TIMEFRAMES](#) enumeration) .

ma_period

[in] Averaging period.

ma_shift

[in] Horizontal shift.

applied

[in] Price type or handle to apply.

Returned value

true if successful, false if indicator hasn't been created.

Main

Returns the buffer element by the specified index.

```
double Main(  
    int index    // Index  
)
```

Parameters

index

[in] Element index.

Returned value

Buffer element of the specified index if successful, or [EMPTY_VALUE](#) if there isn't any correct data.

Type

Returns the object type identifier.

```
virtual int Type() const
```

Returned value

Object type identifier ([IND_TEMA](#) for CiTEMA) .

CiFrAMA

CiFrAMA is a class intended for using the Fractal Adaptive Moving Average technical indicator.

Description

The CiFrAMA class provides the creation, setup and access to the data of the Fractal Adaptive Moving Average indicator.

Declaration

```
class CiFrAMA: public CIndicator
```

Title

```
#include <Indicators\Trend.mqh>
```

Class Methods

Attributes	
MaPeriod	Returns the averaging period
IndShift	Returns the horizontal shift
Applied	Returns the price type or handle to apply
Create Methods	
Create	Creates the indicator
Data Access Methods	
Main	Returns the buffer element
Input/output	
virtual Type	Returns the object type identifier

MaPeriod

Returns the averaging period.

```
int MaPeriod() const
```

Returned value

Returns the averaging period, defined at the indicator creation.

IndShift

Returns the horizontal shift of the indicator.

```
int IndShift() const
```

Returned value

Returns the horizontal shift value, defined at the indicator creation.

Applied

Returns the price type or handle to apply.

```
int Applied() const
```

Returned value

Price type or handle to apply, defined at the indicator creation.

Create

It creates the indicator with specified parameters.

```
bool Create(  
    string          symbol,          // Symbol  
    ENUM_TIMEFRAMES period,          // Period  
    int             ma_period,       // Averaging period  
    int             ma_shift,        // Offset  
    int             applied           // Price type of handle to apply  
)
```

Parameters

symbol

[in] Symbol.

period

[in] Timeframe ([ENUM_TIMEFRAMES](#) enumeration) .

ma_period

[in] Averaging period.

ma_shift

[in] Horizontal shift.

applied

[in] Price type or handle to apply.

Returned value

true if successful, false if indicator hasn't been created.

Main

Returns the buffer element by the specified index.

```
double Main(  
    int index    // Index  
)
```

Parameters

index

[in] Element index.

Returned value

Buffer element of the specified index if successful, or [EMPTY_VALUE](#) if there isn't any correct data.

Type

Returns the object type identifier.

```
virtual int Type() const
```

Returned value

Object type identifier ([IND_FRAMA](#) for CiFrAMA) .

CiAMA

CiAMA is a class intended for using the Adaptive Moving Average technical indicator.

Description

The CiAMA class provides the creation, setup and access to the data of the Adaptive Moving Average indicator.

Declaration

```
class CiAMA: public CIndicator
```

Title

```
#include <Indicators\Trend.mqh>
```

Class Methods

Attributes	
MaPeriod	Returns the averaging period
FastEmaPeriod	Returns the averaging period for the fast EMA
SlowEmaPeriod	Returns the averaging period for the slow EMA
IndShift	Returns the horizontal shift
Applied	Returns the price type or handle to apply
Create Methods	
Create	Creates the indicator
Data Access Methods	
Main	Returns the buffer element
Input/output	
virtual Type	Returns the object type identifier

MaPeriod

Returns the averaging period.

```
int MaPeriod() const
```

Returned value

Returns the averaging period, defined at the indicator creation.

FastEmaPeriod

Returns the averaging period for the fast EMA.

```
int FastEmaPeriod() const
```

Returned value

Returns the averaging period for the fast EMA, defined at the indicator creation.

SlowEmaPeriod

Returns the averaging period for the slow EMA.

```
int SlowEmaPeriod() const
```

Returned value

Returns the averaging period for the slow EMA, defined at the indicator creation.

IndShift

Returns the horizontal shift of the indicator.

```
int IndShift() const
```

Returned value

Returns the horizontal shift value, defined at the indicator creation.

Applied

Returns the price type or handle to apply.

```
int Applied() const
```

Returned value

Price type or handle to apply, defined at the indicator creation.

Create

It creates the indicator with specified parameters.

```
bool Create(  
    string          symbol,           // Symbol  
    ENUM_TIMEFRAMES period,          // Period  
    int             ma_period,        // Averaging period  
    int             fast_ema_period,  // Fast EMA period  
    int             slow_ema_period,  // Slow EMA period  
    int             ind_shift,        // Shift  
    int             applied           // Price type or handle to apply  
)
```

Parameters

string

[in] Symbol.

period

[in] Timeframe ([ENUM_TIMEFRAMES](#) enumeration) .

ma_period

[in] Averaging period.

fast_ema_period

[in] Fast EMA averaging period.

slow_ema_period

[in] Slow EMA averaging period.

ind_shift

[in] Horizontal shift.

applied

[in] Price type or handle to apply.

Returned value

true if successful, false if indicator hasn't been created.

Main

Returns the buffer element by the specified index.

```
double Main(  
    int index    // Index  
)
```

Parameters

index

[in] Element index.

Returned value

Buffer element of the specified index if successful, or [EMPTY_VALUE](#) if there isn't any correct data.

Type

Returns the object type identifier.

```
virtual int Type() const
```

Returned value

Object type identifier ([IND_AMA](#) for CiAMA) .

CiVIDyA

CiVIDyA is a class intended for using the Variable Index DYnamic Average technical indicator.

Description

The CiVIDyA class provides the creation, setup and access to the data of the Variable Index DYnamic Average indicator.

Declaration

```
class CiVIDyA: public CIndicator
```

Title

```
#include <Indicators\Trend.mqh>
```

Class Methods

Attributes	
CmoPeriod	Returns the period for Momentum
EmaPeriod	Returns the averaging period for EMA
IndShift	Returns the horizontal shift
Applied	Returns the price type or handle to apply
Create Methods	
Create	Creates the indicator
Data Access Methods	
Main	Returns the buffer element
Input/output	
virtual Type	Returns the object type identifier

CmoPeriod

Returns the period for Momentum.

```
int CmoPeriod() const
```

Returned value

Returns the period for Momentum, defined at the indicator creation.

EmaPeriod

Returns the averaging period for EMA.

```
int EmaPeriod() const
```

Returned value

Returns the averaging period for EMA, defined at the indicator creation.

IndShift

Returns the horizontal shift of the indicator.

```
int IndShift() const
```

Returned value

Returns the horizontal shift value, defined at the indicator creation.

Applied

Returns the price type or handle to apply.

```
int Applied() const
```

Returned value

Price type or handle to apply, defined at the indicator creation.

Create

It creates the indicator with specified parameters.

```
bool Create(  
    string          symbol,          // Symbol  
    ENUM_TIMEFRAMES period,          // Period  
    int             cmo_period,      // Momentum period  
    int             ema_period,      // Averaging period  
    int             ind_shift,       // Shift  
    int             applied          // Price type or handle to apply  
)
```

Parameters

symbol

[in] Symbol.

period

[in] Timeframe ([ENUM_TIMEFRAMES](#) enumeration) .

cmo_period

[in] Momentum period.

ema_period

[in] Averaging period.

ind_shift

[in] Horizontal shift.

applied

[in] Price type or handle to apply.

Returned value

true if successful, false if indicator hasn't been created.

Main

Returns the buffer element by the specified index.

```
double Main(  
    int index    // Index  
)
```

Parameters

index

[in] Element index.

Returned value

Buffer element of the specified index if successful, or [EMPTY_VALUE](#) if there isn't any correct data.

Type

Returns the object type identifier.

```
virtual int Type() const
```

Returned value

Object type identifier ([IND__VIDYA](#) for CiVIDyA) .

Oscillators

This group of chapters contains the technical details of Oscillators classes of the MQL5 Standard Library and descriptions of all its key components.

Class/group	Description
<u>CiATR</u>	Average True Range
<u>CiBearsPower</u>	Bears Power
<u>CiBullsPower</u>	Bulls Power
<u>CiCCI</u>	Commodity Channel Index
<u>CiChaikin</u>	Chaikin Oscillator
<u>CiDeMarker</u>	DeMarker
<u>CiForce</u>	Force Index
<u>CiMACD</u>	Moving Averages Convergence-Divergence
<u>CiMomentum</u>	Momentum
<u>CiOsMA</u>	Moving Average of Oscillator (MACD histogram)
<u>CiRSI</u>	Relative Strength Index
<u>CiRVI</u>	Relative Vigor Index
<u>CiStochastic</u>	Stochastic Oscillator
<u>CiWPR</u>	Williams' Percent Range
<u>CiTriX</u>	Triple Exponential Moving Averages Oscillator

CiATR

CiATR is a class intended for using the Average True Range technical indicator.

Description

The CiATR class provides the creation, setup and access to the data of the Average True Range indicator.

Declaration

```
class CiATR: public CIndicator
```

Title

```
#include <Indicators\Oscillators.mqh>
```

Class Methods

Attributes	
MaPeriod	Returns the averaging period
Create Methods	
Create	Creates the indicator
Data Access Methods	
Main	Returns the buffer element
Input/output	
virtual Type	Returns the object type identifier

MaPeriod

Returns the averaging period.

```
int MaPeriod() const
```

Returned value

Returns the averaging period, defined at the indicator creation.

Create

It creates the indicator with specified parameters.

```
bool Create(  
    string          symbol,          // Symbol  
    ENUM_TIMEFRAMES period,          // Period  
    int             ma_period        // Averaging period  
)
```

Parameters

symbol

[in] Symbol.

period

[in] Timeframe ([ENUM_TIMEFRAMES](#) enumeration) .

ma_period

[in] Averaging period.

Returned value

true if successful, false if indicator hasn't been created.

Main

Returns the buffer element by the specified index.

```
double Main(  
    int index    // Index  
)
```

Parameters

index

[in] Element index.

Returned value

Buffer element of the specified index if successful, or [EMPTY_VALUE](#) if there isn't any correct data.

Type

Returns the object type identifier.

```
virtual int Type() const
```

Returned value

Object type identifier ([IND_ATR](#) for CiATR) .

CiBearsPower

CiBearsPower is a class intended for using the Bears Power technical indicator.

Description

The CiBearsPower class provides the creation, setup and access to the data of the Bears Power indicator.

Declaration

```
class CiBearsPower: public CIndicator
```

Title

```
#include <Indicators\Oscillators.mqh>
```

Class Methods

Attributes	
MaPeriod	Returns the averaging period
Create Methods	
Create	Creates the indicator
Data Access Methods	
Main	Returns the buffer element
Input/output	
virtual Type	Returns the object type identifier

MaPeriod

Returns the averaging period.

```
int MaPeriod() const
```

Returned value

Returns the averaging period, defined at the indicator creation.

Create

It creates the indicator with specified parameters.

```
bool Create(  
    string          symbol,          // Symbol  
    ENUM_TIMEFRAMES period,          // Period  
    int             ma_period        // Averaging period  
)
```

Parameters

symbol

[in] Symbol.

period

[in] Timeframe ([ENUM_TIMEFRAMES](#) enumeration) .

ma_period

[in] Averaging period.

Returned value

true if successful, false if indicator hasn't been created.

Main

Returns the buffer element by the specified index.

```
double Main(  
    int index    // Index  
)
```

Parameters

index

[in] Element index.

Returned value

Buffer element of the specified index if successful, or [EMPTY_VALUE](#) if there isn't any correct data.

Type

Returns the object type identifier.

```
virtual int Type() const
```

Returned value

Object type identifier ([IND_BEARS](#) for CiBearsPower) .

CiBullsPower

CiBullsPower is a class intended for using the Bulls Power technical indicator.

Description

The CiBullsPower class provides the creation, setup and access to the data of the Bulls Power indicator.

Declaration

```
class CiBullsPower: public CIndicator
```

Title

```
#include <Indicators\Oscillators.mqh>
```

Class Methods

Attributes	
MaPeriod	Returns the averaging period
Create Methods	
Create	Creates the indicator
Data Access Methods	
Main	Returns the buffer element
Input/output	
virtual Type	Returns the object type identifier

MaPeriod

Returns the averaging period.

```
int MaPeriod() const
```

Returned value

Returns the averaging period, defined at the indicator creation.

Create

It creates the indicator with specified parameters.

```
bool Create(  
    string          symbol,          // Symbol  
    ENUM_TIMEFRAMES period,          // Period  
    int             ma_period        // Averaging period  
)
```

Parameters

symbol

[in] Symbol.

period

[in] Timeframe ([ENUM_TIMEFRAMES](#) enumeration) .

ma_period

[in] Averaging period.

Returned value

true if successful, false if indicator hasn't been created.

Main

Returns the buffer element by the specified index.

```
double Main(  
    int index    // Index  
)
```

Parameters

index

[in] Element index.

Returned value

Buffer element of the specified index if successful, or [EMPTY_VALUE](#) if there isn't any correct data.

Type

Returns the object type identifier.

```
virtual int Type() const
```

Returned value

Object type identifier ([IND_BULLS](#) for CiBullsPower) .

CiCCI

CiCCI is a class intended for using the Commodity Channel Index technical indicator.

Description

The CiCCI class provides the creation, setup and access to the data of the Commodity Channel Index indicator.

Declaration

```
class CiCCI: public CIndicator
```

Title

```
#include <Indicators\Oscillators.mqh>
```

Class Methods

Attributes	
MaPeriod	Returns the averaging period
Applied	Returns the price type or handle to apply
Create Methods	
Create	Creates the indicator
Data Access Methods	
Main	Returns the buffer element
Input/output	
virtual Type	Returns the object type identifier

MaPeriod

Returns the averaging period.

```
int MaPeriod() const
```

Returned value

Returns the averaging period, defined at the indicator creation.

Applied

Returns the price type or handle to apply.

```
int Applied() const
```

Returned value

Price type or handle to apply, defined at the indicator creation.

Create

It creates the indicator with specified parameters.

```
bool Create(  
    string          symbol,          // Symbol  
    ENUM_TIMEFRAMES period,          // Period  
    int             ma_period,       // Averaging period  
    int             applied          // Price type or handle to apply  
)
```

Parameters

symbol

[in] Symbol.

period

[in] Timeframe ([ENUM_TIMEFRAMES](#) enumeration) .

ma_period

[in] Averaging period.

applied

[in] Price type or handle to apply.

Returned value

true if successful, false if indicator hasn't been created.

Main

Returns the buffer element by the specified index.

```
double Main(  
    int index    // Index  
)
```

Parameters

index

[in] Element index.

Returned value

Buffer element of the specified index if successful, or [EMPTY_VALUE](#) if there isn't any correct data.

Type

Returns the object type identifier.

```
virtual int Type() const
```

Returned value

Object type identifier ([IND_CCI](#) for CcCI) .

CiChaikin

CiChaikin is a class intended for using the Chaikin Oscillator technical indicator.

Description

The CiChaikin class provides the creation, setup and access to the data of the Chaikin Oscillator indicator.

Declaration

```
class CiChaikin: public CIndicator
```

Title

```
#include <Indicators\Oscillators.mqh>
```

Class Methods

Attributes	
FastMaPeriod	Returns the averaging period for the fast MA
SlowMaPeriod	Returns the averaging period for the slow MA
MaMethod	Returns the averaging method
Applied	Returns the price type or handle to apply
Create Methods	
Create	Creates the indicator
Data Access Methods	
Main	Returns the buffer element
Input/output	
virtual Type	Returns the object type identifier

FastMaPeriod

Returns the averaging period for the fast EMA.

```
int FastMaPeriod() const
```

Returned value

Returns the averaging period for the fast EMA, defined at the indicator creation.

SlowMaPeriod

Returns the averaging period for the slow EMA.

```
int SlowMaPeriod() const
```

Returned value

Returns the averaging period for the slow EMA, defined at the indicator creation.

MaMethod

Returns the averaging method.

```
ENUM_MA_METHOD MaMethod() const
```

Returned value

Returns the averaging method, defined at the indicator creation.

Applied

Returns the volume type to apply.

```
ENUM_APPLIED_VOLUME Applied() const
```

Returned value

Volume type to apply, defined at the indicator creation.

Create

It creates the indicator with specified parameters.

```
bool Create(  
    string          symbol,          // Symbol  
    ENUM_TIMEFRAMES period,          // Period  
    int             fast_ma_period,  // Fast EMA period  
    int             slow_ma_period,  // Slow EMA period  
    ENUM_MA_METHOD  ma_method,       // Averaging method  
    ENUM_APPLIED_VOLUME applied      // Volume type to apply  
)
```

Parameters

symbol

[in] Symbol.

period

[in] Timeframe ([ENUM_TIMEFRAMES](#) enumeration) .

fast_ma_period

[in] Period for fast EMA.

slow_ma_period

[in] Period for slow EMA.

ma_method

[in] Averaging method ([ENUM_MA_METHOD](#) enumeration) .

applied

[in] Volume type to apply ([ENUM_APPLIED_VOLUME](#) enumeration) .

Returned value

true if successful, false if indicator hasn't been created.

Main

Returns the buffer element by the specified index.

```
double Main(  
    int index    // Index  
)
```

Parameters

index

[in] Element index.

Returned value

Buffer element of the specified index if successful, or [EMPTY_VALUE](#) if there isn't any correct data.

Type

Returns the object type identifier.

```
virtual int Type() const
```

Returned value

Object type identifier ([IND_CHAIKIN](#) for CiChaikin) .

CiDeMarker

CiDeMarker is a class intended for using the DeMarker technical indicator.

Description

The CiDeMarker class provides the creation, setup and access to the data of the DeMarker indicator.

Declaration

```
class CiDeMarker: public CIndicator
```

Title

```
#include <Indicators\Oscillators.mqh>
```

Class Methods

Attributes	
MaPeriod	Returns the averaging period
Create Methods	
Create	Creates the indicator
Data Access Methods	
Main	Returns the buffer element
Input/output	
virtual Type	Returns the object type identifier

MaPeriod

Returns the averaging period.

```
int MaPeriod() const
```

Returned value

Returns the averaging period, defined at the indicator creation.

Create

It creates the indicator with specified parameters.

```
bool Create(  
    string          symbol,          // Symbol  
    ENUM_TIMEFRAMES period,          // Period  
    int             ma_period        // Averaging period  
)
```

Parameters

symbol

[in] Symbol.

period

[in] Timeframe ([ENUM_TIMEFRAMES](#) enumeration) .

ma_period

[in] Averaging period.

Returned value

true if successful, false if indicator hasn't been created.

Main

Returns the buffer element by the specified index.

```
double Main(  
    int index    // Index  
)
```

Parameters

index

[in] Element index.

Returned value

Buffer element of the specified index if successful, or [EMPTY_VALUE](#) if there isn't any correct data.

Type

Returns the object type identifier.

```
virtual int Type() const
```

Returned value

Object type identifier ([IND_DEMARKER](#) for CiDeMarker) .

CiForce

CiForce is a class intended for using the Force Index technical indicator.

Description

The CiForce class provides the creation, setup and access to the data of the Force Index indicator.

Declaration

```
class CiForce: public CIndicator
```

Title

```
#include <Indicators\Oscillators.mqh>
```

Class Methods

Attributes	
MaPeriod	Returns the averaging period
MaMethod	Returns the averaging method
Applied	Returns the price type or handle to apply
Create Methods	
Create	Creates the indicator
Data Access Methods	
Main	Returns the buffer element
Input/output	
virtual Type	Returns the object type identifier

MaPeriod

Returns the averaging period.

```
int MaPeriod() const
```

Returned value

Returns the averaging period, defined at the indicator creation.

MaMethod

Returns the averaging method.

```
ENUM_MA_METHOD MaMethod() const
```

Returned value

Returns the averaging method, defined at the indicator creation.

Applied

Returns the volume type to apply.

```
ENUM_APPLIED_VOLUME Applied() const
```

Returned value

Volume type to apply, defined at the indicator creation.

Create

It creates the indicator with specified parameters.

```
bool Create(  
    string          symbol,          // Symbol  
    ENUM_TIMEFRAMES period,          // Period  
    int             ma_period,        // Averaging period  
    ENUM_MA_METHOD  ma_method,        // Averaging method  
    ENUM_APPLIED_VOLUME applied       // Volume type to apply  
)
```

Parameters

symbol

[in] Symbol.

period

[in] Timeframe ([ENUM_TIMEFRAMES](#) enumeration) .

ma_period

[in] Averaging period.

ma_method

[in] Averaging method ([ENUM_MA_METHOD](#) enumeration) .

applied

[in] Volume type to apply ([ENUM_APPLIED_VOLUME](#) enumeration) .

Returned value

true if successful, false if indicator hasn't been created.

Main

Returns the buffer element by the specified index.

```
double Main(  
    int index    // Index  
)
```

Parameters

index

[in] Element index.

Returned value

Buffer element of the specified index if successful, or [EMPTY_VALUE](#) if there isn't any correct data.

Type

Returns the object type identifier.

```
virtual int Type() const
```

Returned value

Object type identifier ([IND_FORCE](#) for CiForce) .

CiMACD

CiMACD is a class intended for using the Moving Averages Convergence-Divergence technical indicator.

Description

The CiMACD class provides the creation, setup and access to the data of the Moving Averages Convergence-Divergence indicator.

Declaration

```
class CiMACD: public CIndicator
```

Title

```
#include <Indicators\Oscillators.mqh>
```

Class Methods

Attributes	
FastEmaPeriod	Returns the averaging period fofo for the fast EMA
SlowEmaPeriod	Returns the averaging period of the slow EMA
SignalPeriod	Returns the averaging period of the signal line
Applied	Returns the price type or handle to apply
Create Methods	
Create	Creates the indicator
Data Access Methods	
Main	Returns the buffer element of the main line
Signal	Returns the buffer element of the signal line
Input/output	
virtual Type	Returns the object type identifier

FastEmaPeriod

Returns the averaging period for the fast EMA.

```
int FastEmaPeriod() const
```

Returned value

Returns the averaging period for the fast EMA, defined at the indicator creation.

SlowEmaPeriod

Returns the averaging period for the slow EMA.

```
int SlowEmaPeriod() const
```

Returned value

Returns the averaging period for the slow EMA, defined at the indicator creation.

SignalPeriod

Returns the averaging period for the signal line.

```
int SignalPeriod() const
```

Returned value

Returns the averaging period for the signal line, defined at the indicator creation.

Applied

Returns the price type or handle to apply.

```
int Applied() const
```

Returned value

Price type or handle to apply, defined at the indicator creation.

Create

It creates the indicator with specified parameters.

```
bool Create(  
    string          symbol,           // Symbol  
    ENUM_TIMEFRAMES period,          // Period  
    int             fast_ema_period,  // Fast EMA period  
    int             slow_ema_period,  // Slow EMA period  
    int             signal_period,    // Signal period  
    int             applied           // Price type or handle to apply  
)
```

Parameters

symbol

[in] Symbol.

period

[in] Timeframe ([ENUM_TIMEFRAMES](#) enumeration) .

fast_ema_period

[in] Fast EMA period.

slow_ema_period

[in] Slow EMA period.

signal_period

[in] Signal line period.

applied

[in] Price type or handle to apply.

Returned value

true if successful, false if indicator hasn't been created.

Main

Returns the buffer element by the specified index.

```
double Main(  
    int index    // Index  
)
```

Parameters

index

[in] Element index.

Returned value

Buffer element of the specified index if successful, or [EMPTY_VALUE](#) if there isn't any correct data.

Signal

Returns the buffer element of the signal line by the specified index.

```
double Signal(  
    int index    // Index  
)
```

Parameters

index

[in] Element index.

Returned value

The buffer element of the signal line of the specified index, or [EMPTY_VALUE](#) if there isn't any correct data.

Type

Returns the object type identifier.

```
virtual int Type() const
```

Returned value

Object type identifier ([IND_MACD](#) for CiMACD) .

CiMomentum

CiMomentum is a class intended for using the Momentum technical indicator.

Description

The CiMomentum class provides the creation, setup and access to the data of the Momentum indicator.

Declaration

```
class CiMomentum: public CIndicator
```

Title

```
#include <Indicators\Oscillators.mqh>
```

Class Methods

Attributes	
MaPeriod	Returns the averaging period
Applied	Returns the price type or handle to apply
Create Methods	
Create	Creates the indicator
Data Access Methods	
Main	Returns the buffer element
Input/output	
virtual Type	Returns the object type identifier

MaPeriod

Returns the averaging period.

```
int MaPeriod() const
```

Returned value

Returns the averaging period, defined at the indicator creation.

Applied

Returns the price type or handle to apply.

```
int Applied() const
```

Returned value

Price type or handle to apply, defined at the indicator creation.

Create

It creates the indicator with specified parameters.

```
bool Create(  
    string          symbol,          // Symbol  
    ENUM_TIMEFRAMES period,          // Period  
    int             ma_period,       // Averaging period  
    int             applied          // Price type or handle to apply  
)
```

Parameters

symbol

[in] Symbol.

period

[in] Timeframe ([ENUM_TIMEFRAMES](#) enumeration) .

ma_period

[in] Averaging period.

applied

[in] Price type or handle to apply.

Returned value

true if successful, false if indicator hasn't been created.

Main

Returns the buffer element by the specified index.

```
double Main(  
    int index    // Index  
)
```

Parameters

index

[in] Element index.

Returned value

Buffer element of the specified index if successful, or [EMPTY_VALUE](#) if there isn't any correct data.

Type

Returns the object type identifier.

```
virtual int Type() const
```

Returned value

Object type identifier ([IND__MOMENTUM](#) for CiMomentum) .

CiOsMA

CiOsMA is a class intended for using the Moving Average of Oscillator (MACD histogram) technical indicator.

Description

The CiOsMA class provides the creation, setup and access to the data of the Moving Average of Oscillator (MACD histogram) indicator.

Declaration

```
class CiOsMA: public CIndicator
```

Title

```
#include <Indicators\Oscilators.mqh>
```

Class Methods

Attributes	
FastEmaPeriod	Returns the averaging period of the fast EMA
SlowEmaPeriod	Returns the averaging period of the slow EMA
SignalPeriod	Returns the averaging period of the signal line
Applied	Returns the price type or handle to apply
Create Methods	
Create	Creates the indicator
Data Access Methods	
Main	Returns the buffer element
Input/output	
virtual Type	Returns the object type identifier

FastEmaPeriod

Returns the averaging period for the fast EMA.

```
int FastEmaPeriod() const
```

Returned value

Returns the averaging period for the fast EMA, defined at the indicator creation.

SlowEmaPeriod

Returns the averaging period for the slow EMA.

```
int SlowEmaPeriod() const
```

Returned value

Returns the averaging period for the slow EMA, defined at the indicator creation.

SignalPeriod

Returns the averaging period for the signal line.

```
int SignalPeriod() const
```

Returned value

Returns the averaging period for the signal line, defined at the indicator creation.

Applied

Returns the price type or handle to apply.

```
int Applied() const
```

Returned value

Price type or handle to apply, defined at the indicator creation.

Create

It creates the indicator with specified parameters.

```
bool Create(  
    string          symbol,           // Symbol  
    ENUM_TIMEFRAMES period,          // Period  
    int             fast_ema_period,  // Fast EMA period  
    int             slow_ema_period,  // Slow EMA period  
    int             signal_period,    // Signal line period  
    int             applied           // Price type or handle to apply  
)
```

Parameters

symbol

[in] Symbol.

period

[in] Timeframe ([ENUM_TIMEFRAMES](#) enumeration) .

fast_ema_period

[in] Fast EMA period.

slow_ema_period

[in] Slow EMA period.

signal_period

[in] Signal line period.

applied

[in] Price type or handle to apply.

Returned value

true if successful, false if indicator hasn't been created.

Main

Returns the buffer element by the specified index.

```
double Main(  
    int index    // Index  
)
```

Parameters

index

[in] Element index.

Returned value

Buffer element of the specified index if successful, or [EMPTY_VALUE](#) if there isn't any correct data.

Type

Returns the object type identifier.

```
virtual int Type() const
```

Returned value

Object type identifier ([IND_OSMA](#) for CiOsMA) .

CiRSI

CiRSI is a class intended for using the Relative Strength Index technical indicator.

Description

The CiRSI class provides the creation, setup and access to the data of the Relative Strength Index indicator.

Declaration

```
class CiRSI: public CIndicator
```

Title

```
#include <Indicators\Oscillators.mqh>
```

Class Methods

Attributes	
MaPeriod	Returns the averaging period
Applied	Returns the price type or handle to apply
Create Methods	
Create	Creates the indicator
Data Access Methods	
Main	Returns the buffer element
Input/output	
virtual Type	Returns the object type identifier

MaPeriod

Returns the averaging period.

```
int MaPeriod() const
```

Returned value

Returns the averaging period, defined at the indicator creation.

Applied

Returns the price type or handle to apply.

```
int Applied() const
```

Returned value

Price type or handle to apply, defined at the indicator creation.

Create

It creates the indicator with specified parameters.

```
bool Create(  
    string          symbol,          // Symbol  
    ENUM_TIMEFRAMES period,          // Period  
    int             ma_period,       // Averaging period  
    int             applied           // Price type or handle to apply  
)
```

Parameters

symbol

[in] Symbol.

period

[in] Timeframe ([ENUM_TIMEFRAMES](#) enumeration) .

ma_period

[in] Averaging period.

applied

[in] Price type or handle to apply.

Returned value

true if successful, false if indicator hasn't been created.

Main

Returns the buffer element by the specified index.

```
double Main(  
    int index    // Index  
)
```

Parameters

index

[in] Element index.

Returned value

Buffer element of the specified index if successful, or [EMPTY_VALUE](#) if there isn't any correct data.

Type

Returns the object type identifier.

```
virtual int Type() const
```

Returned value

Object type identifier ([IND_RSI](#) for CIRSI) .

CiRVI

CiRVI is a class intended for using the Relative Vigor Index technical indicator.

Description

The CiRVI class provides the creation, setup and access to the data of the Relative Vigor Index indicator.

Declaration

```
class CiRVI: public CIndicator
```

Title

```
#include <Indicators\Oscillators.mqh>
```

Class Methods

Attributes	
MaPeriod	Returns the averaging period
Create Methods	
Create	Creates the indicator
Data Access Methods	
Main	Returns the buffer element of the base line
Signal	Returns the buffer element of the signal line
Input/output	
virtual Type	Returns the object type identifier

MaPeriod

Returns the averaging period.

```
int MaPeriod() const
```

Returned value

Returns the averaging period, defined at the indicator creation.

Create

It creates the indicator with specified parameters.

```
bool Create(  
    string          symbol,          // Symbol  
    ENUM_TIMEFRAMES period,          // Period  
    int             ma_period        // Averaging period  
)
```

Parameters

symbol

[in] Symbol.

period

[in] Timeframe ([ENUM_TIMEFRAMES](#) enumeration) .

ma_period

[in] Averaging period.

Returned value

true if successful, false if indicator hasn't been created.

Main

Returns the buffer element by the specified index.

```
double Main(  
    int index    // Index  
)
```

Parameters

index

[in] Element index.

Returned value

Buffer element of the specified index if successful, or [EMPTY_VALUE](#) if there isn't any correct data.

Signal

Returns the buffer element of the signal line by the specified index.

```
double Signal(  
    int index    // Index  
)
```

Parameters

index

[in] Element index.

Returned value

The buffer element of the signal line of the specified index, or [EMPTY_VALUE](#) if there isn't any correct data.

Type

Returns the object type identifier.

```
virtual int Type() const
```

Returned value

Object type identifier ([IND_RVI](#) for CIRVI) .

CiStochastic

CiStochastic is a class intended for using the Stochastic Oscillator technical indicator.

Description

The CiStochastic class provides the creation, setup and access to the data of the Stochastic Oscillator indicator.

Declaration

```
class CiStochastic: public CIndicator
```

Title

```
#include <Indicators\Oscilators.mqh>
```

Class Methods

Attributes	
<u>Kperiod</u>	Returns the averaging period for the %K line
<u>Dperiod</u>	Returns the averaging period for the %D line
<u>Slowing</u>	Returns the slowing period
<u>MaMethod</u>	Returns the averaging method
<u>PriceField</u>	Price type (Low/High ore Close/Close) to apply
Create Methods	
<u>Create</u>	Creates the indicator
Data Access Methods	
<u>Main</u>	Returns the buffer element of the base line
<u>Signal</u>	Returns the buffer element of the signal line
Input/output	
virtual <u>Type</u>	Returns the object type identifier

Kperiod

Returns the averaging period for the %K line.

```
int Kperiod() const
```

Returned value

Returns the averaging period for the %K line, defined at the indicator creation.

Dperiod

Returns the averaging period for the %D line.

```
int Dperiod() const
```

Returned value

Returns the averaging period for the %D line, defined at the indicator creation.

Slowing

Returns the period of slowing.

```
int Slowing() const
```

Returned value

Returns the period of slowing, defined at the indicator creation.

MaMethod

Returns the averaging method.

```
ENUM_MA_METHOD MaMethod() const
```

Returned value

Returns the averaging method, defined at the indicator creation.

PriceField

Returns the price type (Low/High or Close/Close) to apply.

```
ENUM_STO_PRICE PriceField() const
```

Returned value

The price type to apply, defined at the indicator creation.

Create

It creates the indicator with specified parameters.

```
bool Create(  
    string          symbol,          // Symbol  
    ENUM_TIMEFRAMES period,          // Period  
    int             Kperiod,         // Averaging period of %K  
    int             Dperiod,         // Averaging period of %D  
    int             slowing,         // Slowing period  
    ENUM_MA_METHOD  ma_method,       // Averaging method  
    ENUM_STO_PRICE  price_field      // Price type to apply  
)
```

Parameters

symbol

[in] Symbol.

period

[in] Timeframe ([ENUM_TIMEFRAMES](#) enumeration) .

Kperiod

[in] Averaging period of %K line.

Dperiod

[in] Averaging period of %D line.

slowing

[in] Slowing period.

ma_method

[in] Averaging method ([ENUM_MA_METHOD](#) enumeration) .

price_field

[in] Price type (Low/High or Close/Close) to apply ([ENUM_STO_PRICE](#) enumeration) .

Returned value

true if successful, false if indicator hasn't been created.

Main

Returns the buffer element by the specified index.

```
double Main(  
    int index    // Index  
)
```

Parameters

index

[in] Element index.

Returned value

Buffer element of the specified index if successful, or [EMPTY_VALUE](#) if there isn't any correct data.

Signal

Returns the buffer element of the signal line by the specified index.

```
double Signal(  
    int index    // Index  
)
```

Parameters

index

[in] Element index.

Returned value

Signal

Returns the buffer element of the signal line by the specified index.

```
double Signal(  
    int index    // Index  
)
```

Parameters

index

[in] Element index.

Returned value

The buffer element of the signal line of the specified index, or [EMPTY_VALUE](#) if there isn't any correct data.

Type

Returns the object type identifier.

```
virtual int Type() const
```

Returned value

Object type identifier ([IND_STOCHASTIC](#) for CiStochastic) .

CiTriX

CiTriX is a class intended for using the Triple Exponential Moving Averages Oscillator technical indicator.

Description

The CiTriX class provides the creation, setup and access to the data of the Triple Exponential Moving Averages Oscillator indicator.

Declaration

```
class CiTriX: public CIndicator
```

Title

```
#include <Indicators\Oscilators.mqh>
```

Class Methods

Attributes	
<u>MaPeriod</u>	Returns the averaging period
<u>Applied</u>	Returns the price type or handle to apply
Create Methods	
<u>Create</u>	Creates the indicator
Data Access Methods	
<u>Main</u>	Returns the buffer element
Input/output	
virtual <u>Type</u>	Returns the object type identifier

MaPeriod

Returns the averaging period.

```
int MaPeriod() const
```

Returned value

Returns the averaging period, defined at the indicator creation.

Applied

Returns the price type or handle to apply.

```
int Applied() const
```

Returned value

Price type or handle to apply, defined at the indicator creation.

Create

It creates the indicator with specified parameters.

```
bool Create(  
    string          symbol,          // Symbol  
    ENUM_TIMEFRAMES period,          // Period  
    int             ma_period,       // Averaging period  
    int             applied           // Price type or handle  
)
```

Parameters

symbol

[in] Symbol.

period

[in] Timeframe ([ENUM_TIMEFRAMES](#) enumeration) .

ma_period

[in] Averaging period.

applied

[in] Price type of handle to apply.

Returned value

true if successful, false if indicator hasn't been created.

Main

Returns the buffer element by the specified index.

```
double Main(  
    int index    // Index  
)
```

Parameters

index

[in] Element index.

Returned value

Buffer element of the specified index if successful, or [EMPTY_VALUE](#) if there isn't any correct data.

Type

Returns the object type identifier.

```
virtual int Type() const
```

Returned value

Object type identifier ([IND__TRIX](#) for CiTriX) .

CiWPR

CiWPR is a class intended for using the Williams' Percent Range technical indicator.

Description

The CiWPR class provides the creation, setup and access to the data of the Williams' Percent Range indicator.

Declaration

```
class CiWPR: public CIndicator
```

Title

```
#include <Indicators\Oscillators.mqh>
```

Class Methods

Attributes	
CalcPeriod	Returns the calculation period
Create Methods	
Create	Creates the indicator
Data Access Methods	
Main	Returns the buffer element
Input/output	
virtual Type	Returns the object type identifier

CalcPeriod

Returns the period for calculation.

```
int CalcPeriod() const
```

Returned value

Returns the the period for calculation, defined at the indicator creation.

Create

It creates the indicator with specified parameters.

```
bool Create(  
    string          symbol,          // Symbol  
    ENUM_TIMEFRAMES period,          // Period  
    int             calc_period      // Calculation period  
)
```

Parameters

symbol

[in] Symbol.

period

[in] Timeframe ([ENUM_TIMEFRAMES](#) enumeration) .

calc_period

[in] Period for calculation.

Returned value

true if successful, false if indicator hasn't been created.

Main

Returns the buffer element by the specified index.

```
double Main(  
    int index    // Index  
)
```

Parameters

index

[in] Element index.

Returned value

Buffer element of the specified index if successful, or [EMPTY_VALUE](#) if there isn't any correct data.

Type

Returns the object type identifier.

```
virtual int Type() const
```

Returned value

Object type identifier ([IND_WPR](#) for CiWPR) .

Volume Indicators

This group of chapters contains technical details of Volume Indicator classes of the MQL5 Standard Library and descriptions of all its key components.

Class/group	Description
CiAD	Accumulation/Distribution
CiMFI	Money Flow Index
CiOBV	On Balance Volume
CiVolumes	Volumes

CiAD

CiAD is a class intended for using the Accumulation/Distribution technical indicator.

Description

The CiAD class provides the creation, setup and access to the data of the Accumulation/Distribution indicator.

Declaration

```
class CiAD: public CIndicator
```

Title

```
#include <Indicators\Volumes.mqh>
```

Class Methods

Attributes	
Applied	Returns the volume type to apply
Create Methods	
Create	Creates the indicator
Data Access Methods	
Main	Returns the buffer element
Input/output	
virtual Type	Returns the object type identifier

Applied

Returns the volume type to apply.

```
ENUM_APPLIED_VOLUME Applied() const
```

Returned value

Volume type to apply, defined at the indicator creation.

Create

It creates the indicator with specified parameters.

```
bool Create(  
    string          symbol,      // Symbol  
    ENUM_TIMEFRAMES period,     // Period  
    ENUM_APPLIED_VOLUME applied // Volume type to apply  
)
```

Parameters

symbol

[in] Symbol.

period

[in] Timeframe ([ENUM_TIMEFRAMES](#) enumeration) .

applied

[in] Volume type to apply ([ENUM_APPLIED_VOLUME](#) enumeration) .

Returned value

true if successful, false if indicator hasn't been created.

Main

Returns the buffer element by the specified index.

```
double Main(  
    int index    // Index  
)
```

Parameters

index

[in] Element index.

Returned value

Buffer element of the specified index if successful, or [EMPTY_VALUE](#) if there isn't any correct data.

Type

Returns the object type identifier.

```
virtual int Type() const
```

Returned value

Object type identifier ([IND_AD](#) for CiAD) .

CiMFI

CiMFI is a class intended for using the Money Flow Index technical indicator.

Description

The CiMFI class provides the creation, setup and access to the data of the Money Flow Index indicator.

Declaration

```
class CiMFI: public CIndicator
```

Title

```
#include <Indicators\Volumes.mqh>
```

Class Methods

Attributes	
MaPeriod	Returns the averaging period
Applied	Returns the volume type to apply
Create Methods	
Create	Creates the indicator
Data Access Methods	
Main	Returns the buffer element
Input/output	
virtual Type	Returns the object type identifier

MaPeriod

Returns the averaging period.

```
int MaPeriod() const
```

Returned value

Returns the averaging period, defined at the indicator creation.

Applied

Returns the volume type to apply.

```
ENUM_APPLIED_VOLUME Applied() const
```

Returned value

Volume type to apply, defined at the indicator creation.

Create

It creates the indicator with specified parameters.

```
bool Create(  
    string          symbol,          // Symbol  
    ENUM_TIMEFRAMES period,          // Period  
    int             ma_period,        // Averaging period  
    ENUM_APPLIED_VOLUME applied      // Volume type to apply  
)
```

Parameters

symbol

[in] Symbol.

period

[in] Timeframe ([ENUM_TIMEFRAMES](#) enumeration) .

ma_period

[in] Averaging period.

applied

[in] Volume type to apply ([ENUM_APPLIED_VOLUME](#) enumeration) .

Returned value

true if successful, false if indicator hasn't been created.

Main

Returns the buffer element by the specified index.

```
double Main(  
    int index    // Index  
)
```

Parameters

index

[in] Element index.

Returned value

Buffer element of the specified index if successful, or [EMPTY_VALUE](#) if there isn't any correct data.

Type

Returns the object type identifier.

```
virtual int Type() const
```

Returned value

Object type identifier ([IND_MFI](#) for CiMFI) .

CiOBV

CiOBV is a class intended for using the On Balance Volume technical indicator.

Description

The CiOBV class provides the creation, setup and access to the data of the On Balance Volume indicator.

Declaration

```
class CiOBV: public CIndicator
```

Title

```
#include <Indicators\Volumes.mqh>
```

Class Methods

Attributes	
Applied	Returns the volume type to apply
Create Methods	
Create	Creates the indicator
Data Access Methods	
Main	Returns the buffer element
Input/output	
virtual Type	Returns the object type identifier

Applied

Returns the volume type to apply.

```
ENUM_APPLIED_VOLUME Applied() const
```

Returned value

Volume type to apply, defined at the indicator creation.

Create

It creates the indicator with specified parameters.

```
bool Create(  
    string          symbol,      // Symbol  
    ENUM_TIMEFRAMES period,     // Period  
    ENUM_APPLIED_VOLUME applied  // Volume type to apply  
)
```

Parameters

symbol

[in] Symbol.

period

[in] Timeframe ([ENUM_TIMEFRAMES](#) enumeration) .

applied

[in] Volume type to apply ([ENUM_APPLIED_VOLUME](#) enumeration) .

Returned value

true if successful, false if indicator hasn't been created.

Main

Returns the buffer element by the specified index.

```
double Main(  
    int index    // Index  
)
```

Parameters

index

[in] Element index.

Returned value

Buffer element of the specified index if successful, or [EMPTY_VALUE](#) if there isn't any correct data.

Type

Returns the object type identifier.

```
virtual int Type() const
```

Returned value

Object type identifier ([IND_OBV](#) for CiOBV) .

CiVolumes

CiVolumes is a class intended for using the Volumes technical indicator.

Description

The CiVolumes class provides the creation, setup and access to the data of the Volumes indicator.

Declaration

```
class CiVolumes: public CIndicator
```

Title

```
#include <Indicators\Volumes.mqh>
```

Class Methods

Attributes	
Applied	Returns the volume type to apply
Create Methods	
Create	Creates the indicator
Data Access Methods	
Main	Returns the buffer element
Input/output	
virtual Type	Returns the object type identifier

Applied

Returns the volume type to apply.

```
ENUM_APPLIED_VOLUME Applied() const
```

Returned value

Volume type to apply, defined at the indicator creation.

Create

It creates the indicator with specified parameters.

```
bool Create(  
    string          symbol,      // Symbol  
    ENUM_TIMEFRAMES period,     // Period  
    ENUM_APPLIED_VOLUME applied // Volume type to apply  
)
```

Parameters

symbol

[in] Symbol.

period

[in] Timeframe ([ENUM_TIMEFRAMES](#) enumeration) .

applied

[in] Volume type to apply ([ENUM_APPLIED_VOLUME](#) enumeration) .

Returned value

true if successful, false if indicator hasn't been created.

Main

Returns the buffer element by the specified index.

```
double Main(  
    int index    // Index  
)
```

Parameters

index

[in] Element index.

Returned value

Buffer element of the specified index if successful, or [EMPTY_VALUE](#) if there isn't any correct data.

Type

Returns the object type identifier.

```
virtual int Type() const
```

Returned value

Object type identifier ([IND_VOLUMES](#) for CiVolumes) .

Bill Williams Indicators

This group of chapters contains technical details of Bill Williams Indicator classes of the MQL5 Standard Library and descriptions of all its key components.

Class/group	Description
CiAC	Accelerator Oscillator
CiAlligator	Alligator
CiAO	Awesome Oscillator
CiFractals	Fractals
CiGator	Gator Oscillator
CiBWMFI	Market Facilitation Index

CiAC

CiAC is a class intended for using the Accelerator Oscillator technical indicator.

Description

The CiAC class provides the creation, setup and access to the data of the Accelerator Oscillator indicator.

Declaration

```
class CiAC: public CIndicator
```

Title

```
#include <Indicators\BillWilliams.mqh>
```

Class Methods

Create Methods	
Create	Creates the indicator
Data Access Methods	
Main	Returns the buffer element
Input/output	
virtual Type	Returns the object type identifier

Create

It creates the indicator with specified parameters.

```
bool Create(  
    string      symbol,      // Symbol  
    ENUM_TIMEFRAMES period    // Period  
)
```

Parameters

symbol

[in] Symbol.

period

[in] Timeframe ([ENUM_TIMEFRAMES](#) enumeration) .

Returned value

true if successful, false if indicator hasn't been created.

Main

Returns the buffer element by the specified index.

```
double Main(  
    int index    // Index  
)
```

Parameters

index

[in] Element index.

Returned value

Buffer element of the specified index if successful, or [EMPTY_VALUE](#) if there isn't any correct data.

Type

Returns the object type identifier.

```
virtual int Type() const
```

Returned value

Object type identifier ([IND_AC](#) for CiAC) .

CiAlligator

CiAlligator is a class intended for using the Alligator technical indicator.

Description

The CiAlligator class provides the creation, setup and access to the data of the Alligator indicator.

Declaration

```
class CiAlligator: public CIndicator
```

Title

```
#include <Indicators\BillWilliams.mqh>
```

Class Methods

Attributes	
<u>JawPeriod</u>	Returns the averaging period for the Jaws line
<u>JawShift</u>	Returns the horizontal shift of the Jaws line
<u>TeethPeriod</u>	Returns the averaging period for the Teeths line
<u>TeethShift</u>	Returns the horizontal shift of the Teeths line
<u>LipsPeriod</u>	Returns the averaging period for the Lips line
<u>LipsShift</u>	Returns the horizontal shift of the Lips line
<u>MaMethod</u>	Returns the averaging method
<u>Applied</u>	Returns the price type or handle to apply
Create Methods	
<u>Create</u>	Creates the indicator
Data Access Methods	
<u>Jaw</u>	Returns the buffer element of the Jaws line buffer
<u>Teeth</u>	Returns the buffer element of the Teeths line buffer
<u>Lips</u>	Returns the buffer element of the Lips line buffer
Input/output	
virtual <u>Type</u>	Returns the object type identifier

JawPeriod

Returns the averaging period for the Jaw line.

```
int JawPeriod() const
```

Returned value

Returns the averaging period for the Jaw line, defined at the indicator creation.

JawShift

Returns the horizontal shift of the Jaws line.

```
int JawShift() const
```

Returned value

Horizontal shift of the Jaws line, defined at the indicator creation.

TeethPeriod

Returns the averaging period for the Teeth line.

```
int TeethPeriod() const
```

Returned value

Returns the averaging period for the Teeth line, defined at the indicator creation.

TeethShift

Returns the horizontal shift of the Teeths line.

```
int TeethShift() const
```

Returned value

Horizontal shift of the Teeths line, defined at the indicator creation.

LipsPeriod

Returns the averaging period for the Lips line.

```
int LipsPeriod() const
```

Returned value

Returns the averaging period for the Lips line, defined at the indicator creation.

LipsShift

Returns the horizontal shift of the Lips line.

```
int LipsShift() const
```

Returned value

Horizontal shift of the Lips line, defined at the indicator creation.

MaMethod

Returns the averaging method.

```
ENUM_MA_METHOD MaMethod() const
```

Returned value

Returns the averaging method, defined at the indicator creation.

Applied

Returns the price type or handle to apply.

```
int Applied() const
```

Returned value

Price type or handle to apply, defined at the indicator creation.

Create

It creates the indicator with specified parameters.

```
bool Create(
    string          symbol,           // Symbol
    ENUM_TIMEFRAMES period,          // Period
    int             jaw_period,       // Jaws period
    int             jaw_shift,        // Jaws shift
    int             teeth_period,     // Teeths period
    int             teeth_shift,      // Teeths shift
    int             lips_period,      // Lips period
    int             lips_shift,       // Lips shift
    ENUM_MA_METHOD  ma_method,       // Averaging method
    int             applied           // Volume type to apply
)
```

Parameters

symbol

[in] Symbol.

period

[in] Timeframe ([ENUM_TIMEFRAMES](#) enumeration) .

jaw_period

[in] Jaws period.

jaw_shift

[in] Jaws shift.

teeth_period

[in] Teeths period.

teeth_shift

[in] Teeths shift.

lips_period

[in] Lips period.

lips_shift

[in] Lips shift.

ma_method

[in] Moving average method ([ENUM_MA_METHOD](#) enumeration) .

applied

[in] Volume type to apply.

Returned value

true if successful, false if indicator hasn't been created.

Jaw

Returns the buffer element of the Jaws line by the specified index.

```
double  Jaw(  
    int  index    // Index  
)
```

Parameters

index

[in] Element index.

Returned value

The buffer element of the Jaws line of the specified index, or [EMPTY_VALUE](#) if there isn't any correct data.

Teeth

Returns the buffer element of the Teeths line by the specified index.

```
double Teeth(  
    int index    // Index  
)
```

Parameters

index

[in] Element index.

Returned value

The buffer element of the Teeths line of the specified index, or [EMPTY_VALUE](#) if there isn't any correct data.

Lips

Returns the buffer element of the Lips line by the specified index.

```
double Lips(  
    int index    // Index  
)
```

Parameters

index

[in] Element index.

Returned value

The buffer element of the Lips line of the specified index, or [EMPTY_VALUE](#) if there isn't any correct data.

Type

Returns the object type identifier.

```
virtual int Type() const
```

Returned value

Object type identifier ([IND_ALLIGATOR](#) for CiAlligator) .

CiAO

CiAO is a class intended for using the Awesome Oscillator technical indicator.

Description

The CiAO class provides the creation, setup and access to the data of the Awesome Oscillator indicator.

Declaration

```
class CiAO: public CIndicator
```

Title

```
#include <Indicators\BillWilliams.mqh>
```

Class Methods

Create Methods	
Create	Creates the indicator
Data Access Methods	
Main	Returns the buffer element
Input/output	
virtual Type	Returns the object type identifier

Create

It creates the indicator with specified parameters.

```
bool Create(  
    string      symbol,      // Symbol  
    ENUM_TIMEFRAMES period    // Period  
)
```

Parameters

symbol

[in] Symbol.

period

[in] Timeframe ([ENUM_TIMEFRAMES](#) enumeration) .

Returned value

true if successful, false if indicator hasn't been created.

Main

Returns the buffer element by the specified index.

```
double Main(  
    int index    // Index  
)
```

Parameters

index

[in] Element index.

Returned value

Buffer element of the specified index if successful, or [EMPTY_VALUE](#) if there isn't any correct data.

Type

Returns the object type identifier.

```
virtual int Type() const
```

Returned value

Object type identifier ([IND_AO](#) for CiAO) .

CiFractals

CiFractals is a class intended for using the Fractals technical indicator.

Description

The CiFractals class provides the creation, setup and access to the data of the Fractals indicator.

Declaration

```
class CiFractals: public CIndicator
```

Title

```
#include <Indicators\BillWilliams.mqh>
```

Class Methods

Create Methods	
Create	Creates the indicator
Data Access Methods	
Upper	Returns the buffer element of the upper buffer
Lower	Returns the buffer element of the lower buffer
Input/output	
virtual Type	Returns the object type identifier

Create

It creates the indicator with specified parameters.

```
bool Create(  
    string      symbol,      // Symbol  
    ENUM_TIMEFRAMES period    // Period  
)
```

Parameters

symbol

[in] Symbol.

period

[in] Timeframe ([ENUM_TIMEFRAMES](#) enumeration) .

Returned value

true if successful, false if indicator hasn't been created.

Upper

Returns the buffer element of the upper buffer by the specified index.

```
double Upper (  
    int index    // Index  
)
```

Parameters

index

[in] Element index.

Returned value

The buffer element of the upper buffer of the specified index, or [EMPTY_VALUE](#) if there isn't any correct data.

Lower

Returns the buffer element of the lower buffer by the specified index.

```
double Lower(  
    int index    // Index  
)
```

Parameters

index

[in] Element index.

Returned value

The buffer element of the lower buffer of the specified index, or [EMPTY_VALUE](#) if there isn't any correct data.

Type

Returns the object type identifier.

```
virtual int Type() const
```

Returned value

Object type identifier ([IND_FRACTALS](#) for CiFractals) .

CiGator

CiGator is a class intended for using the Gator Oscillator technical indicator.

Description

The CiGator class provides the creation, setup and access to the data of the Gator Oscillator indicator.

Declaration

```
class CiGator: public CIndicator
```

Title

```
#include <Indicators\BillWilliams.mqh>
```

Class Methods

Attributes	
<u>JawPeriod</u>	Returns the averaging period for the Jaws line
<u>JawShift</u>	Returns the horizontal shift of the Jaws line
<u>TeethPeriod</u>	Returns the averaging period for the Teeths line
<u>TeethShift</u>	Returns the horizontal shift of the Teeths line
<u>LipsPeriod</u>	Returns the averaging period for the Lips line
<u>LipsShift</u>	Returns the horizontal shift of the Lips line
<u>MaMethod</u>	Returns the averaging method
<u>Applied</u>	Returns the price type or handle to apply
Create Methods	
<u>Create</u>	Creates the indicator
Data Access Methods	
<u>Upper</u>	Returns the buffer element of the upper buffer
<u>Lower</u>	Returns the buffer element of the lower buffer
Input/output	
virtual <u>Type</u>	Returns the object type identifier

JawPeriod

Returns the averaging period for the Jaws line.

```
int JawPeriod() const
```

Returned value

Returns the averaging period for the Jaws line, defined at the indicator creation.

JawShift

Returns the horizontal shift of the Jaws line.

```
int JawShift() const
```

Returned value

Horizontal shift of the Jaws line, defined at the indicator creation.

TeethPeriod

Returns the averaging period for the Teeth line.

```
int TeethPeriod() const
```

Returned value

Returns the averaging period for the Teeth line, defined at the indicator creation.

TeethShift

Returns the horizontal shift of the Teeths line.

```
int TeethShift() const
```

Returned value

Horizontal shift of the Teeths line, defined at the indicator creation.

LipsPeriod

Returns the averaging period for the Lips line.

```
int LipsPeriod() const
```

Returned value

Returns the averaging period for the Lips line, defined at the indicator creation.

LipsShift

Returns the horizontal shift of the Lips line.

```
int LipsShift() const
```

Returned value

Horizontal shift of the Lips line, defined at the indicator creation.

MaMethod

Returns the averaging method.

```
ENUM_MA_METHOD MaMethod() const
```

Returned value

Returns the averaging method, defined at the indicator creation.

Applied

Returns the price type or handle to apply.

```
int Applied() const
```

Returned value

Price type or handle to apply, defined at the indicator creation.

Create

It creates the indicator with specified parameters.

```
bool Create(
    string          symbol,           // Symbol
    ENUM_TIMEFRAMES period,          // Period
    int             jaw_period,       // Jaws period
    int             jaw_shift,        // Jaws shift
    int             teeth_period,     // Teeths period
    int             teeth_shift,      // Teeths shift
    int             lips_period,      // Lips period
    int             lips_shift,       // Lips shift
    ENUM_MA_METHOD  ma_method,       // Averaging method
    int             applied           // Volume type to apply
)
```

Parameters

symbol

[in] Symbol.

period

[in] Timeframe ([ENUM_TIMEFRAMES](#) enumeration) .

jaw_period

[in] Jaws period.

jaw_shift

[in] Jaws shift.

teeth_period

[in] Teeths period.

teeth_shift

[in] Teeths shift.

lips_period

[in] Lips period.

lips_shift

[in] Lips shift.

ma_method

[in] Averaging method ([ENUM_MA_METHOD](#) enumeration) .

applied

[in] Volume type to apply.

Returned value

true if successful, false if indicator hasn't been created.

Upper

Returns the buffer element of the upper buffer by the specified index.

```
double Upper (  
    int index    // Index  
)
```

Parameters

index

[in] Element index.

Returned value

The buffer element of the upper buffer of the specified index, or [EMPTY_VALUE](#) if there isn't any correct data.

Lower

Returns the buffer element of the lower buffer by the specified index.

```
double Lower (
    int index    // Index
)
```

Parameters

index

[in] Element index.

Returned value

The buffer element of the lower buffer of the specified index, or [EMPTY_VALUE](#) if there isn't any correct data.

Type

Returns the object type identifier.

```
virtual int Type() const
```

Returned value

Object type identifier ([IND_GATOR](#) for CiGator) .

CiBWMFI

CiBWMFI is a class intended for using the Market Facilitation Index by Bill Williams technical indicator.

Description

The CiBWMFI class provides the creation, setup and access to the data of the Market Facilitation Index by Bill Williams indicator.

Declaration

```
class CiBWMFI: public CIndicator
```

Title

```
#include <Indicators\BillWilliams.mqh>
```

Class Methods

Attributes	
Applied	Returns the volume type to apply
Create Methods	
Create	Creates the indicator
Data Access Methods	
Main	Returns the buffer element
Input/output	
virtual Type	Returns the object type identifier

Applied

Returns the volume type to apply.

```
ENUM_APPLIED_VOLUME Applied() const
```

Returned value

Volume type to apply, defined at the indicator creation.

Create

It creates the indicator with specified parameters.

```
bool Create(  
    string          symbol,      // Symbol  
    ENUM_TIMEFRAMES period,      // Period  
    ENUM_APPLIED_VOLUME applied  // Volume type to apply  
)
```

Parameters

symbol

[in] Symbol.

period

[in] Timeframe ([ENUM_TIMEFRAMES](#) enumeration) .

applied

[in] Volume type to apply ([ENUM_APPLIED_VOLUME](#) enumeration) .

Returned value

true if successful, false if indicator hasn't been created.

Main

Returns the buffer element by the specified index.

```
double Main(  
    int index    // Index  
)
```

Parameters

index

[in] Element index.

Returned value

Buffer element of the specified index if successful, or [EMPTY_VALUE](#) if there isn't any correct data.

Type

Returns the object type identifier.

```
virtual int Type() const
```

Returned value

Object type identifier ([IND_BWMFI](#) for CiBWMFI) .

CiCustom

CiCustom is a class intended for using the custom technical indicators.

Description

The CiCustom class provides the creation, setup and access to the data of the custom indicator.

Declaration

```
class CiCustom: public CIndicator
```

Title

```
#include <Indicators\Custom.mqh>
```

Class Methods

Attributes	
NumBuffers	Sets the number of buffers
NumParams	Gets the number of parameters
ParamType	Gets the type of the specified parameter
ParamLong	Gets the value of the specified parameter of integer type
ParamDouble	Gets the value of the specified parameter of double type
ParamString	Gets the value of the specified parameter of string type
Input/output	
virtual Type	Gets the object type identifier

NumBuffers

Sets the number of buffers.

```
bool NumBuffers(  
    int buffers    // number of buffers  
)
```

Returned value

true if successful, false if buffers haven't been set.

NumParams

Gets the number of parameters.

```
int NumParams() const
```

Returned value

Number of parameters, used in creation of the indicator.

ParamType

Gets a type of the parameter with specified index.

```
ENUM_DATATYPE ParamType (  
    int index    // parameter index  
) const
```

Parameters

index

[in] Parameter index.

Returned value

Returns the data type (value of [ENUM_DATATYPE](#) enumeration) of the parameter with specified index, used in indicator creation.

Note

If parameter index is invalid, it returns [WRONG_VALUE](#).

ParamLong

Gets the value of specified parameter of long type.

```
long ParamLong(  
    int index    // index  
) const
```

Parameters

index

[in] Parameter index.

Returned value

The value of specified parameter of long type, used in creation of the indicator.

Note

If the number is invalid, it returns 0.

ParamDouble

Gets the value of specified parameter of double type.

```
double ParamDouble(  
    int index    // index  
) const
```

Parameters

index

[in] Parameter index.

Returned value

The value of specified parameter of double type, used in creation of the indicator.

Note

If the number is invalid, it returns [EMPTY_VALUE](#).

ParamString

Gets the value of specified parameter of string type.

```
string ParamString(  
    int index    // index  
    ) const
```

Parameters

index

[in] Parameter index.

Returned value

The value of specified string parameter, used in creation of the indicator.

Note

If the number is invalid, it returns an empty string.

Type

Returns the object type identifier.

```
virtual int Type() const
```

Returned value

Object type identifier ([IND_CUSTOM](#) for CiCustom) .

Trade Classes

This section contains technical details of working with trade classes and description of the relevant components of the MQL5 standard library.

Using trade classes will save time when creating custom programs (experts) .

MQL5 Standard Library (in terms of data sets) is placed in the terminal working directory, in the Include\Arrays folder.

Class/Group	Description
<u>CAccountInfo</u>	Class for working with trade account properties
<u>CSymbolInfo</u>	Class for working with trade instrument properties
<u>COrderInfo</u>	Class for working with pending order properties
<u>CHistoryOrderInfo</u>	Class for working with history order properties
<u>CPositionInfo</u>	Class for working with open position properties
<u>CDealInfo</u>	Class for working with history deal properties
<u>CTrade</u>	Class for trade operations execution

CAccountInfo

CAccountInfo is a class for easy access to the currently opened trade account properties.

Description

CAccountInfo class provides easy access to the currently opened trade account properties.

Declaration

```
class CAccountInfo : public CObject
```

Title

```
#include <Trade\AccountInfo.mqh>
```

Class methods by groups

Access to integer type properties	
Login	Gets the account number
TradeMode	Gets the trade mode
TradeModeDescription	Gets the trade mode as a string
Leverage	Gets the amount of given leverage
MarginMode	Gets the mode of account stop out
MarginModeDescription	Gets the description of account stop out mode
TradeAllowed	Gets the flag of trade allowance
TradeExpert	Gets the flag of automated trade allowance
LimitOrders	Gets the maximal number of allowed pending orders
Access to double type properties	
Balance	Gets the balance of account
Credit	Gets the amount of given credit
Profit	Gets the amount of current profit on account
Equity	Gets the amount of current equity on account
Margin	Gets the amount of reserved margin
FreeMargin	Gets the amount of free margin
MarginLevel	Gets the level of margin
MarginCall	Gets the level of margin for deposit
MarginStopOut	Gets the level of margin for Stop Out

Access to text properties	
Name	Gets the client name
Server	Gets the trade server name
Currency	Gets the deposit currency name
Company	Gets the company name, that serves an account
Access to MQL5 API functions	
Integer	Gets the value of specified integer type property
Double	Gets the value of specified double type property
String	Gets value of specified string type property
Additional methods	
OrderProfitCheck	Gets the evaluated profit, based on the parameters passed
MarginCheck	Gets the amount of margin, required to execute trade operation
FreeMarginCheck	Gets the amount of free margin, left after execution of trade operation
MaxLotCheck	Gets the maximal possible volume of trade operation

Login

Gets the account number.

```
long Login() const
```

Returned value

Account number.

TradeMode

Gets the trade mode.

```
ENUM_ACCOUNT_TRADE_MODE TradeMode() const
```

Returned value

Trade mode (value of [ENUM_ACCOUNT_TRADE_MODE](#) enumeration) .

TradeModeDescription

Gets the trade mode as a string.

```
string TradeModeDescription() const
```

Returned value

Trade mode as a string.

Leverage

Gets the amount of given leverage.

```
long Leverage() const
```

Returned value

Amount of given leverage.

MarginMode

Gets the mode of account Stop Out.

```
ENUM_ACCOUNT_STOPOUT_MODE MarginMode() const
```

Returned value

Account Stop Out mode (value of [ENUM_ACCOUNT_STOPOUT_MODE](#) enumeration) .

MarginModeDescription

Gets the mode of setting minimal margin level as a string.

```
string MarginModeDescription() const
```

Returned value

Mode of setting minimal margin level as a string.

TradeAllowed

Gets the flag of trade allowance.

```
bool TradeAllowed() const
```

Returned value

Flag of trade allowance.

TradeExpert

Gets the flag of automated trade allowance.

```
bool TradeExpert() const
```

Returned value

Flag of automated trade allowance.

LimitOrders

Gets the maximal number of allowed pending orders

```
int LimitOrders() const
```

Returned value

The maximal number of allowed pending orders.

Note

0 - no limits.

Balance

Gets the balance of account.

```
double Balance() const
```

Returned value

The balance of account (in deposit currency) .

Credit

Gets the amount of given credit.

```
double Credit() const
```

Returned value

Amount of given credit (in deposit currency) .

Profit

Gets the amount of current profit on account.

```
double Profit() const
```

Returned value

Amount of current profit on account (in deposit currency) .

Equity

Gets the amount of current equity on account.

```
double Equity() const
```

Returned value

Amount of current equity on account (in deposit currency) .

Margin

Gets the amount of reserved margin.

```
double Margin() const
```

Returned value

Amount of reserved margin (in deposit currency) .

FreeMargin

Gets the amount of free margin.

```
double FreeMargin() const
```

Returned value

Amount of free margin (in deposit currency) .

MarginLevel

Gets the level of margin.

```
double MarginLevel() const
```

Returned value

Level of margin.

MarginCall

Gets the level of margin for a deposit.

```
double MarginCall() const
```

Returned value

Level of margin for a deposit.

MarginStopOut

Gets the level of margin for Stop Out.

```
double MarginStopOut() const
```

Returned value

Level of margin for Stop Out.

Name

Gets the client name.

```
string Name() const
```

Returned value

Client name.

Server

Gets the trade server name.

```
string Server() const
```

Returned value

Trade server name.

Currency

Gets the deposit currency name.

```
string Currency() const
```

Returned value

Deposit currency name.

Company

Gets the company name, that serves an account.

```
string Company() const
```

Returned value

Company name, that serves an account.

InfoInteger

Gets the value of specified integer type property.

```
long InfoInteger(  
    ENUM_ACCOUNT_INFO_INTEGER prop_id // property ID  
) const
```

Parameters

prop_id

[in] Identifier of the property. The value can be one of the values of [ENUM_ACCOUNT_INFO_INTEGER](#) enumeration.

Returned value

Value of [long](#) type.

InfoDouble

Gets the value of specified double type property.

```
double InfoDouble(  
    ENUM_ACCOUNT_INFO_DOUBLE prop_id // property ID  
) const
```

Parameters

prop_id

[in] Identifier of the property. The value can be one of the values of [ENUM_ACCOUNT_INFO_DOUBLE](#) enumeration.

Returned value

Value of [double](#) type.

InfoString

Gets the value of specified string type property.

```
string InfoString(  
    ENUM_ACCOUNT_INFO_STRING prop_id    // property ID  
) const
```

Parameters

prop_id

[in] Identifier of the property. The value can be one of the values of [ENUM_ACCOUNT_INFO_STRING](#) enumeration.

Returned value

Value of [string](#) type.

OrderProfitCheck

The function calculates the profit for the current account, based on the parameters passed. The function is used for pre-evaluation of the result of a trade operation. The value is returned in the account currency.

```
double OrderProfitCheck(  
    const string      symbol,           // symbol  
    ENUM_ORDER_TYPE   trade_operation,  // operation type (ORDER_TYPE_BUY or ORDER_TYPE_SELL)  
    double            volume,           // volume  
    double            price_open,        // open price  
    double            price_close       // close price  
) const
```

Parameters

symbol

[in] Symbol for trade operation.

trade_operation

[in] Type of trade operation ([ENUM_ORDER_TYPE](#) enumeration) .

volume

[in] Volume of trade operation.

price_open

[in] Open price.

price_close

[in] Close price.

Returned value

If successful, it returns amount of profit or [EMPTY_VALUE](#) in the case of error.

MarginCheck

Gets the amount of margin, required for trade operation.

```
double MarginCheck(  
    const string      symbol,           // symbol  
    ENUM_ORDER_TYPE   trade_operation, // operation  
    double            volume,          // volume  
    double            price            // price  
) const
```

Parameters

symbol

[in] Symbol for trade operation.

trade_operation

[in] Type of trade operation ([ENUM_ORDER_TYPE](#) enumeration) .

volume

[in] Volume of trade operation.

price

[in] Price of trade operation.

Returned value

Amount of margin, required for trade operation.

FreeMarginCheck

Gets the amount of free margin, left after trade operation.

```
double FreeMarginCheck(  
    const string      symbol,           // symbol  
    ENUM_ORDER_TYPE   trade_operation, // operation  
    double            volume,          // volume  
    double            price            // price  
) const
```

Parameters

symbol

[in] Symbol for trade operation.

trade_operation

[in] Type of trade operation ([ENUM_ORDER_TYPE](#) enumeration) .

volume

[in] Volume of trade operation.

price

[in] Price of trade operation.

Returned value

Amount of free margin, left after trade operation.

MaxLotCheck

Gets the maximal possible volume of trade operation.

```
double MaxLotCheck(  
    const string      symbol,           // symbol  
    ENUM_ORDER_TYPE   trade_operation,  // operation  
    double            price             // price  
) const
```

Parameters

symbol

[in] Symbol for trade operation.

trade_operation

[in] Type of trade operation ([ENUM_ORDER_TYPE](#) enumeration) .

price

[in] Price of trade operation.

Returned value

Maximal possible volume of trade operation.

CSymbolInfo

CSymbolInfo is a class for easy access to the symbol properties.

Description

CSymbolInfo class provides access to the symbol properties.

Declaration

```
class CSymbolInfo : public CObject
```

Title

```
#include <Trade\SymbolInfo.mqh>
```

Class methods by groups

Controlling	
Refresh	Refreshes the symbol data
RefreshRates	Refreshes the symbol quotes
Properties	
Name	Gets/sets symbol name
Select	Gets/sets the "Market Watch" symbol flag
IsSynchronized	Checks the symbol synchronization with server
Volumes	
Volume	Gets the volume of last deal
VolumeHigh	Gets the maximal volume for a day
VolumeLow	Gets the minimal volume for a day
VolumeBid	Gets the volume in the current Bid
VolumeAsk	Gets the volume in the current Ask
Miscellaneous	
Time	Gets the time of last quote
Spread	Gets the amount of spread (in points)
SpreadFloat	Gets the flag of floating spread
TickBookDepth	Gets the depth of ticks saving
Levels	
StopsLevel	Gets the minimal indent for orders (in points)
FreezeLevel	Gets the distance of freezing trade operations (in points)

Bid prices	
Bid	Gets the current Bid price
BidHigh	Gets the maximal Bid price for a day
BidLow	Gets the minimal Bid price for a day
Ask prices	
Ask	Gets the current Ask price
AskHigh	Gets the maximal Ask price for a day
AskLow	Gets the minimal Ask price for a day
Prices	
Last	Gets the current Last price
LastHigh	Gets the maximal Last price for a day
LastLow	Gets the minimal Last price for a day
Trade modes	
TradeCalcMode	Gets the mode of contract cost calculation
TradeCalcModeDescription	Gets the mode of contract cost calculation as a string
TradeMode	Gets the type of order execution
TradeModeDescription	Gets the type of order execution as a string
TradeExecution	Gets the closing of deals mode
TradeExecutionDescription	Gets the closing of deals mode as a string
Swaps	
SwapMode	Gets the swap calculation model
SwapModeDescription	Gets the swap calculation model as a string
SwapRollover3days	Gets the day of triple swap charge
SwapRollover3daysDescription	Gets the day of triple swap charge as a string
Margins and flags	
MarginInitial	Gets the value of initial margin
MarginMaintenance	Gets the value of maintenance margin
MarginLong	Gets the rate of margin charging for long positions
MarginShort	Gets the rate of margin charging for short positions
MarginLimit	Gets the rate of margin charging for Limit orders
MarginStop	Gets the rate of margin charging for Stop orders
MarginStopLimit	Gets the rate of margin charging for StopLimit orders

TradeTimeFlags	Gets the flags of the order expiration allowed modes
TradeFillFlags	Gets the flags of the order filling allowed modes
Quantization	
Digits	Gets the number of digits after period
Point	Gets the value of one point
TickValue	Gets the cost of tick (minimal change of price)
TickValueProfit	Gets the calculated tick price for a profitable position
TickValueLoss	Gets the calculated tick price for a losing position
TickSize	Gets the minimal change of price
Contracts sizes	
ContractSize	Gets the amount of trade contract
LotsMin	Gets the minimal volume to close a deal
LotsMax	Gets the maximal volume to close a deal
LotsStep	Gets the minimal step of volume change to close a deal
LotsLimit	Gets the maximal allowed volume of opened position and pending orders (direction insensitive) for one symbol
Swaps sizes	
SwapLong	Gets the value of long position swap
SwapShort	Gets the value of short position swap
Text properties	
CurrencyBase	Gets the name of symbol base currency
CurrencyProfit	Gets the profit currency name
CurrencyMargin	Gets the margin currency name
Bank	Gets the name of current quote source
Description	Gets the string description of symbol
Path	Gets the path in symbols tree
Access to MQL5 API functions	
InfoInteger	Gets the value of specified integer type property
InfoDouble	Gets the value of specified double type property
InfoString	Gets the value of specified string type property
Service functions	
NormalizePrice	Returns the value of price, normalized using the symbol properties

Refresh

Refreshes the symbol data.

```
void Refresh()
```

Returned value

None.

Note

The symbol should be selected by [Name](#) method.

RefreshRates

Refreshes the symbol quotes data.

```
bool RefreshRates ()
```

Returned value

true - in case of success, false - if unable to refresh quotes.

Note

The symbol should be selected by [Name](#) method.

Name

Gets symbol name.

```
string Name() const
```

Returned value

Symbol name.

Name

Sets symbol name.

```
void Name(string name)
```

Returned value

None.

Select

Gets the "Market Watch" symbol flag.

```
bool Select() const
```

Returned value

Gets the "Market Watch" symbol flag.

Select

Sets the "Market Watch" symbol flag.

```
bool Select()
```

Returned value

true - in case of success, false - if unable to change flag.

IsSynchronized

Checks the symbol synchronization with server.

```
bool IsSynchronized() const
```

Returned value

true - if the symbol is synchronized with server, false - if not.

Note

The symbol should be selected by [Name](#) method.

Volume

Gets the volume of last deal.

```
long Volume() const
```

Returned value

Volume of last deal.

Note

The symbol should be selected by [Name](#) method.

VolumeHigh

Gets the maximal volume of the day.

```
long VolumeHigh() const
```

Returned value

Maximal volume of the day.

Note

The symbol should be selected by [Name](#) method.

VolumeLow

Gets the minimal volume of the day.

```
long VolumeLow() const
```

Returned value

Minimal volume of the day.

Note

The symbol should be selected by [Name](#) method.

VolumeBid

Gets the volume of the current Bid.

```
long VolumeBid() const
```

Returned value

Volume of the current Bid.

Note

The symbol should be selected by [Name](#) method.

VolumeAsk

Gets the volume of the current Ask.

```
long VolumeAsk() const
```

Returned value

Volume of the current Ask.

Note

The symbol should be selected by [Name](#) method.

Time

Gets the time of last quote.

```
datetime Time() const
```

Returned value

Time of last quote.

Note

The symbol should be selected by [Name](#) method.

Spread

Gets the amount of spread (in points) .

```
int Spread() const
```

Returned value

Gets the amount of spread (in points) .

Note

The symbol should be selected by [Name](#) method.

SpreadFloat

Gets the flag of floating spread.

```
bool SpreadFloat() const
```

Returned value

Flag of floating spread.

Note

The symbol should be selected by [Name](#) method.

TickBookDepth

Gets the depth of ticks saving.

```
int TickBookDepth() const
```

Returned value

Depth of ticks saving.

Note

The symbol should be selected by [Name](#) method.

StopsLevel

Gets the minimal stop level for orders (in points) .

```
int StopsLevel() const
```

Returned value

Minimal stop level for orders (in points) .

Note

The symbol should be selected by [Name](#) method.

FreezeLevel

Gets the freeze level (in points) .

```
int FreezeLevel() const
```

Returned value

Distance of freeze level (in points) .

Note

The symbol should be selected by [Name](#) method.

Bid

Gets the current Bid price.

```
double Bid() const
```

Returned value

Current Bid price.

Note

The symbol should be selected by [Name](#) method.

BidHigh

Gets the maximal Bid price of the day.

```
double BidHigh() const
```

Returned value

Maximal Bid price of the day.

Note

The symbol should be selected by [Name](#) method.

BidLow

Gets the minimal Bid price of the day.

```
double BidLow() const
```

Returned value

Minimal Bid price of the day.

Note

The symbol should be selected by [Name](#) method.

Ask

Gets the current Ask price.

```
double Ask() const
```

Returned value

Current Ask price.

Note

The symbol should be selected by [Name](#) method.

AskHigh

Gets the maximal Ask price for a day.

```
double AskHigh() const
```

Returned value

Maximal Ask price of the day.

Note

The symbol should be selected by [Name](#) method.

AskLow

Gets the minimal Ask price for a day.

```
double AskLow() const
```

Returned value

Minimal Ask price of the day.

Note

The symbol should be selected by [Name](#) method.

Last

Gets the current Last price.

```
double Last() const
```

Returned value

Current Last price.

LastHigh

Gets the maximal Last price of the day.

```
double LastHigh() const
```

Returned value

Maximal Last price of the day.

Note

The symbol should be selected by [Name](#) method.

LastLow

Gets the minimal Last price of the day.

```
double LastLow() const
```

Returned value

Minimal Last price of the day.

Note

The symbol should be selected by [Name](#) method.

TradeCalcMode

Gets the mode of contract cost calculation.

```
ENUM_SYMBOL_CALC_MODE TradeCalcMode() const
```

Returned value

Mode of contract cost calculation (value of [ENUM_SYMBOL_CALC_MODE](#) enumeration) .

Note

The symbol should be selected by [Name](#) method.

TradeCalcModeDescription

Gets the mode of contract cost calculation as a string.

```
string TradeCalcModeDescription() const
```

Returned value

Mode of contract cost calculation as a string.

Note

The symbol should be selected by [Name](#) method.

TradeMode

Gets the order execution type.

```
ENUM_SYMBOL_TRADE_MODE TradeMode() const
```

Returned value

Order execution type (value of [ENUM_SYMBOL_TRADE_MODE](#) enumeration) .

Note

The symbol should be selected by [Name](#) method.

TradeModeDescription

Gets the trade mode as a string.

```
string TradeModeDescription() const
```

Returned value

Trade mode as a string.

Note

The symbol should be selected by [Name](#) method.

TradeExecution

Gets the trade execution mode.

```
ENUM_SYMBOL_TRADE_EXECUTION TradeExecution() const
```

Returned value

Trade execution mode (value of [ENUM_SYMBOL_TRADE_EXECUTION](#) enumeration) .

Note

The symbol should be selected by [Name](#) method.

TradeExecutionDescription

Gets the description of trade execution mode as a string.

```
string TradeExecutionDescription() const
```

Returned value

Trade execution mode as a string.

Note

The symbol should be selected by [Name](#) method.

SwapMode

Gets the swap calculation mode.

```
ENUM_SYMBOL_SWAP_MODE SwapMode() const
```

Returned value

Swap calculation mode (value of [ENUM_SYMBOL_SWAP_MODE](#) enumeration) .

Note

The symbol should be selected by [Name](#) method.

SwapModeDescription

Gets the swap mode description as a string.

```
string SwapModeDescription() const
```

Returned value

Swap mode description as a string.

Note

The symbol should be selected by [Name](#) method.

SwapRollover3days

Gets the swap rollover day.

```
ENUM_DAY_OF_WEEK SwapRollover3days () const
```

Returned value

Swap rollover day (value of [ENUM_DAY_OF_WEEK](#) enumeration) .

Note

The symbol should be selected by [Name](#) method.

SwapRollover3daysDescription

Gets the swap rollover day as a string.

```
string SwapRollover3daysDescription() const
```

Returned value

Swap rollover day as a string.

Note

The symbol should be selected by [Name](#) method.

MarginInitial

Gets the value of initial margin.

```
double MarginInitial ()
```

Returned value

Value of initial margin.

Note

It points the amount of margin (in margin currency of instrument) , that is charged from one lot.
Used to check client's equity, when he enters the market.

The symbol should be selected by [Name](#) method.

MarginMaintenance

Gets the value of maintenance margin.

```
double MarginMaintenance()
```

Returned value

Value of maintenance margin.

Note

It points the amount of margin (in margin currency of instrument) , that is charged from one lot.
Used to check client's equity, when the account state is changed. If the maintenance margin is equal to 0, then the initial margin is used.

The symbol should be selected by [Name](#) method.

MarginLong

Gets the rate of margin charging on long positons.

```
double MarginLong() const
```

Returned value

Rate of margin charging on long positons.

Note

The symbol should be selected by [Name](#) method.

MarginShort

Gets the rate of margin charging on short positons.

```
double MarginShort() const
```

Returned value

Rate of margin charging on short positons.

Note

The symbol should be selected by [Name](#) method.

MarginLimit

Gets the rate of margin charging on Limit orders.

```
double MarginLimit() const
```

Returned value

Rate of margin charging on Limit orders.

Note

The symbol should be selected by [Name](#) method.

MarginStop

Gets the rate of margin charging on Stop orders.

```
double MarginStop() const
```

Returned value

Rate of margin charging on Stop orders.

Note

The symbol should be selected by [Name](#) method.

MarginStopLimit

Gets the rate of margin charging on Stop Limit orders.

```
double MarginStopLimit() const
```

Returned value

Rate of margin charging on Stop Limit orders.

Note

The symbol should be selected by [Name](#) method.

TradeTimeFlags

Gets the flags of the order expiration allowed modes.

```
int TradeTimeFlags() const
```

Returned value

Flags of the order expiration allowed modes.

Note

The symbol should be selected by [Name](#) method.

TradeFillFlags

Gets the flags of the order filling allowed modes.

```
int TradeFillFlags() const
```

Returned value

Flags of the order filling allowed modes.

Note

The symbol should be selected by [Name](#) method.

Digits

Gets the number of digits after period.

```
int Digits() const
```

Returned value

Gets the number of digits after period.

Note

The symbol should be selected by [Name](#) method.

Point

Gets the value of one point.

```
double Point() const
```

Returned value

Value of one point.

Note

The symbol should be selected by [Name](#) method.

TickValue

Gets the cost of tick (minimal change of price) .

```
double TickValue() const
```

Returned value

Cost of tick (minimal change of price) .

Note

The symbol should be selected by [Name](#) method.

TickValueProfit

Gets the calculated tick price for a profitable position.

```
double TickValueProfit() const
```

Returned value

The calculated tick price for a profitable position.

Note

The symbol should be selected by [Name](#) method.

TickValueLoss

Gets the calculated tick price for a losing position.

```
double TickValueLoss() const
```

Returned value

The calculated tick price for a losing position.

Note

The symbol should be selected by [Name](#) method.

TickSize

Gets the minimal change of price.

```
double TickSize() const
```

Returned value

Minimal change of price.

Note

The symbol should be selected by [Name](#) method.

ContractSize

Gets the amount of trade contract.

```
double ContractSize() const
```

Returned value

Amount of trade contract.

Note

The symbol should be selected by [Name](#) method.

LotsMin

Gets the minimal volume to close a deal.

```
double LotsMin() const
```

Returned value

Minimal volume to close a deal.

Note

The symbol should be selected by [Name](#) method.

LotsMax

Gets the maximal volume to close a deal.

```
double LotsMax() const
```

Returned value

Maximal volume to close a deal.

Note

The symbol should be selected by [Name](#) method.

LotsStep

Gets the minimal step of volume change to close a deal.

```
double LotsStep() const
```

Returned value

Minimal step of volume change to close a deal.

Note

The symbol should be selected by [Name](#) method.

LotsLimit

Gets the maximal allowed volume of opened position and pending orders (direction insensitive) for one symbol.

```
double LotsLimit() const
```

Returned value

The maximal allowed volume of opened position and pending orders (direction insensitive) for one symbol.

Note

The symbol should be selected by [Name](#) method.

SwapLong

Gets the value of long position swap.

```
double SwapLong() const
```

Returned value

Value of long position swap.

Note

The symbol should be selected by [Name](#) method.

SwapShort

Gets the value of short position swap.

```
double SwapShort() const
```

Returned value

Value of short position swap.

Note

The symbol should be selected by [Name](#) method.

CurrencyBase

Gets the name of symbol base currency.

```
string CurrencyBase() const
```

Returned value

Name of symbol base currency.

Note

The symbol should be selected by [Name](#) method.

CurrencyProfit

Gets the profit currency name.

```
string CurrencyProfit() const
```

Returned value

Profit currency name.

Note

The symbol should be selected by [Name](#) method.

CurrencyMargin

Gets the margin currency name.

```
string CurrencyMargin() const
```

Returned value

Margin currency name.

Note

The symbol should be selected by [Name](#) method.

Bank

Gets the name of current quote source.

```
string Bank() const
```

Returned value

Name of current quote source.

Note

The symbol should be selected by [Name](#) method.

Description

Gets the string description of symbol.

```
string Description() const
```

Returned value

String description of symbol.

Note

The symbol should be selected by [Name](#) method.

Path

Gets the path in symbols tree.

```
string Path() const
```

Returned value

Gets the path in symbols tree.

Note

The symbol should be selected by [Name](#) method.

InfoInteger

Gets the value of specified integer type property.

```
bool InfoInteger (
    ENUM_SYMBOL_INFO_INTEGER prop_id,    // property ID
    long& var                            // reference to variable
) const
```

Parameters

prop_id

[in] ID of integer type property (value of [ENUM_SYMBOL_INFO_INTEGER](#) enumeration) .

var

[out] Reference to long type variable to place result.

Returned value

true - in case of success, false - if unable to get property value.

Note

The symbol should be selected by [Name](#) method.

InfoDouble

Gets the value of specified double type property.

```
bool InfoDouble(  
    ENUM_SYMBOL_INFO_DOUBLE prop_id,    // property ID  
    double& var                        // reference to variable  
) const
```

Parameters

prop_id

[in] ID of double type property (value of [ENUM_SYMBOL_INFO_DOUBLE](#) enumeration) .

var

[out] Reference to double type variable to place result.

Returned value

true - in case of success, false - if unable to get property value.

Note

The symbol should be selected by [Name](#) method.

InfoString

Gets the value of specified string type property.

```
bool InfoString(  
    ENUM_SYMBOL_INFO_STRING prop_id,    // property ID  
    string& var                        // reference to variable  
) const
```

Parameters

prop_id

[in] ID of text property.

var

[out] Reference to string type variable to place result.

Returned value

true - in case of success, false - if unable to get property value.

Note

The symbol should be selected by [Name](#) method.

NormalizePrice

Returns the value of price, normalized using the symbol properties.

```
double NormalizePrice(  
    double price // price  
) const
```

Parameters

price
[in] Price.

Returned value

Normalized price.

Note

The symbol should be selected by [Name](#) method.

COrderInfo

COrderInfo is a class for easy access to the pending order properties.

Description

COrderInfo class provides access to the pending order properties.

Declaration

```
class COrderInfo : public CObject
```

Title

```
#include <Trade\OrderInfo.mqh>
```

Class methods by groups

Access to integer type properties	
Ticket	Gets the ticket of an order, previously selected for access
TimeSetup	Gets the time of order placement
OrderType	Gets the order type
OrderTypeDescription	Gets the order type as a string
State	Gets the order state
StateDescription	Gets the order state as a string
TimeExpiration	Gets the time of order expiration
TimeDone	Gets the time of order execution or cancellation
TypeFilling	Gets the type of order execution by remainder
TypeFillingDescription	Gets the type of order execution by remainder as a string
TypeTime	Gets the type of order at the time of the expiration
TypeTimeDescription	Gets the order type by expiration time as a string
Magic	Gets the ID of expert, that placed the order
PositionId	Gets the ID of position
Access to double type properties	
VolumeInitial	Gets the initial volume of order
VolumeCurrent	Gets the unfilled volume of order
PriceOpen	Gets the order price
StopLoss	Gets the order's Stop Loss

<u>TakeProfit</u>	Gets the order's Take Profit
<u>PriceCurrent</u>	Gets the current price by order symbol
<u>PriceStopLimit</u>	Gets the price of setting limit order
Access to text properties	
<u>Symbol</u>	Gets the name of order symbol
<u>Comment</u>	Gets the order comment
Access to MQL5 API functions	
<u>InfoInteger</u>	Gets the value of specified integer type property
<u>InfoDouble</u>	Gets the value of specified double type property
<u>InfoString</u>	Gets value of specified string type property
State	
<u>StoreState</u>	Saves the order parameters
<u>CheckState</u>	Checks the current parameters against the saved parameters
Selection	
<u>Select</u>	Selects an order by ticket for further access to its properties
<u>SelectByIndex</u>	Selects an order by index for further access to its properties

Ticket

Gets the ticket of an order, previously selected for access using the [Select](#) method.

```
ulong Ticket() const
```

Returned value

Order ticket if successful, otherwise - [ULONG_MAX](#).

Note

The order should be selected using the [Select](#) (by ticket) or [SelectByIndex](#) (by index) methods.

TimeSetup

Gets the time of order placement.

```
datetime TimeSetup() const
```

Returned value

Time of order placement.

Note

The order should be selected using the [Select](#) (by ticket) or [SelectByIndex](#) (by index) methods.

OrderType

Gets the order type.

```
ENUM_ORDER_TYPE OrderType()
```

Returned value

Order type (value of [ENUM_ORDER_TYPE](#) enumeration) .

Note

The order should be selected using the [Select](#) (by ticket) or [SelectByIndex](#) (by index) methods.

TypeDescription

Gets the order type as a string.

```
string TypeDescription() const
```

Returned value

Order type as a string.

Note

The order should be selected using the [Select](#) (by ticket) or [SelectByIndex](#) (by index) methods.

State

Gets the order state.

```
ENUM_ORDER_STATE State() const
```

Returned value

Order state (value of [ENUM_ORDER_STATE](#) enumeration) .

Note

The order should be selected using the [Select](#) (by ticket) or [SelectByIndex](#) (by index) methods.

StateDescription

Gets the order state as a string.

```
string StateDescription() const
```

Returned value

Order state as a string.

Note

The order should be selected using the [Select](#) (by ticket) or [SelectByIndex](#) (by index) methods.

TimeExpiration

Gets the order expiration time.

```
datetime TimeExpiration() const
```

Returned value

Order expiration time, set on its placement.

Note

The order should be selected using the [Select](#) (by ticket) or [SelectByIndex](#) (by index) methods.

TimeDone

Gets the time of order execution or cancellation.

```
datetime TimeDone() const
```

Returned value

Time of order execution or cancellation.

Note

The order should be selected using the [Select](#) (by ticket) or [SelectByIndex](#) (by index) methods.

TypeFilling

Gets the order filling type.

```
ENUM_ORDER_TYPE_FILLING TypeFilling() const
```

Returned value

Order filling type (value of [ENUM_ORDER_TYPE_FILLING](#) enumeration) .

Note

The order should be selected using the [Select](#) (by ticket) or [SelectByIndex](#) (by index) methods.

TypeFillingDescription

Gets the order filling type as a string.

```
string TypeFillingDescription() const
```

Returned value

Order filling type as a string.

Note

The order should be selected using the [Select](#) (by ticket) or [SelectByIndex](#) (by index) methods.

TypeTime

Gets the type of order at the time of the expiration.

```
ENUM_ORDER_TYPE_TIME TypeTime () const
```

Returned value

Type of order at the time of the expiration.

Note

The order should be selected using the [Select](#) (by ticket) or [SelectByIndex](#) (by index) methods.

TypeTimeDescription

Gets the order type by expiration time as a string.

```
string TypeTimeDescription() const
```

Returned value

Order type by expiration time as a string.

Note

The order should be selected using the [Select](#) (by ticket) or [SelectByIndex](#) (by index) methods.

Magic

Gets the ID of expert, that placed the order.

```
long Magic() const
```

Returned value

ID of expert, that placed the order.

Note

The order should be selected using the [Select](#) (by ticket) or [SelectByIndex](#) (by index) methods.

PositionId

Gets the ID of position.

```
long PositionId() const
```

Returned value

ID of position, in which the order was involved.

Note

The order should be selected using the [Select](#) (by ticket) or [SelectByIndex](#) (by index) methods.

VolumeInitial

Gets the initial volume of order.

```
double VolumeInitial() const
```

Returned value

Initial volume of order.

Note

The order should be selected using the [Select](#) (by ticket) or [SelectByIndex](#) (by index) methods.

VolumeCurrent

Gets the unfilled volume of order.

```
double VolumeCurrent() const
```

Returned value

Unfilled volume of order.

Note

The order should be selected using the [Select](#) (by ticket) or [SelectByIndex](#) (by index) methods.

PriceOpen

Gets the order price.

```
double PriceOpen() const
```

Returned value

Price of order placement.

Note

The order should be selected using the [Select](#) (by ticket) or [SelectByIndex](#) (by index) methods.

StopLoss

Gets the order's Stop Loss.

```
double StopLoss() const
```

Returned value

Order's Stop Loss.

Note

The order should be selected using the [Select](#) (by ticket) or [SelectByIndex](#) (by index) methods.

TakeProfit

Gets the order's Take Profit.

```
double TakeProfit() const
```

Returned value

Order's Take Profit.

Note

The order should be selected using the [Select](#) (by ticket) or [SelectByIndex](#) (by index) methods.

PriceCurrent

Gets the current price by order symbol.

```
double PriceCurrent() const
```

Returned value

Current price by order symbol.

Note

The order should be selected using the [Select](#) (by ticket) or [SelectByIndex](#) (by index) methods.

PriceStopLimit

Gets the price of setting limit order.

```
double PriceStopLimit() const
```

Returned value

Price of setting limit order.

Note

The order should be selected using the [Select](#) (by ticket) or [SelectByIndex](#) (by index) methods.

Symbol

Gets the name of order symbol.

```
string Symbol() const
```

Returned value

Name of order symbol.

Note

The order should be selected using the [Select](#) (by ticket) or [SelectByIndex](#) (by index) methods.

Comment

Gets the order comment.

```
string Comment() const
```

Returned value

Order comment.

Note

The order should be selected using the [Select](#) (by ticket) or [SelectByIndex](#) (by index) methods.

InfoInteger

Gets the value of specified integer type property.

```
bool InfoInteger (
    ENUM_ORDER_PROPERTY_INTEGER prop_id,    // property ID
    long& var                                // reference to variable
) const
```

Parameters

prop_id

[in] ID of integer type property (value of [ENUM_ORDER_PROPERTY_INTEGER](#) enumeration) .

var

[out] Reference to long type variable to place result.

Returned value

true - in case of success, false - if unable to get property value.

Note

The order should be selected using the [Select](#) (by ticket) or [SelectByIndex](#) (by index) methods.

InfoDouble

Gets the value of specified double type property.

```
bool InfoDouble(  
    ENUM_ORDER_PROPERTY_DOUBLE prop_id,    // property ID  
    double& var                            // reference to variable  
) const
```

Parameters

prop_id

[in] ID of double type property (value of [ENUM_ORDER_PROPERTY_DOUBLE](#) enumeration) .

var

[out] Reference to double type variable to place result.

Returned value

true - in case of success, false - if unable to get property value.

Note

The order should be selected using the [Select](#) (by ticket) or [SelectByIndex](#) (by index) methods.

InfoString

Gets the value of specified string type property.

```
bool InfoString(  
    ENUM_ORDER_PROPERTY_STRING prop_id,    // property ID  
    string& var                            // reference to variable  
) const
```

Parameters

prop_id

[in] ID of text property.

var

[out] Reference to string type variable to place result.

Returned value

true - in case of success, false - if unable to get property value.

Note

The order should be selected using the [Select](#) (by ticket) or [SelectByIndex](#) (by index) methods.

StoreState

Saves the order parameters.

```
void StoreState()
```

Returned value

None.

CheckState

Checks the current parameters against the saved parameters.

```
bool CheckState()
```

Returned value

true - if the order parameters have changed since the last call of the [StoreState\(\)](#) method,
otherwise - false.

Select

Selects an order by ticket for further access to its properties.

```
bool Select(  
    ulong ticket // order ticket  
)
```

Returned value

true - in case of success, false - if unable to select order.

SelectByIndex

Selects an order by index for further access to its properties.

```
bool  SelectByIndex(  
    int  index    // order index  
)
```

Returned value

true - in case of success, false - if unable to select order.

CHistoryOrderInfo

CHistoryOrderInfo is a class for easy access to the history order properties.

Description

CHistoryOrderInfo class provides easy access to the history order properties.

Declaration

```
class CHistoryOrderInfo : public CObject
```

Title

```
#include <Trade\HistoryOrderInfo.mqh>
```

Class methods by groups

Access to integer type properties	
TimeSetup	Gets the time of order placement
OrderType	Gets the order type
OrderTypeDescription	Gets the order type as a string
State	Gets the order state
StateDescription	Gets the order state as a string
TimeExpiration	Gets the time of order expiration
TimeDone	Gets the time of order expiration or cancellation
TypeFilling	Gets the type of order execution by remainder
TypeFillingDescription	Gets the type of order execution by remainder as a string
TypeTime	Gets the type of order at the time of the expiration
TypeTimeDescription	Gets the order type by expiration time as a string
Magic	Gets the ID of expert, that placed the order
PositionId	Gets the ID of position
Access to double type properties	
VolumeInitial	Gets the initial volume of order
VolumeCurrent	Gets the unfilled volume of order
PriceOpen	Gets the order price
StopLoss	Gets the order's Stop Loss
TakeProfit	Gets the order's Take Profit

PriceCurrent	Gets the current price by order symbol
PriceStopLimit	Gets the price of setting limit order
Access to text properties	
Symbol	Gets the order symbol
Comment	Gets the order comment
Access to MQL5 API functions	
InfoInteger	Gets the value of specified integer type property
InfoDouble	Gets the value of specified double type property
InfoString	Gets value of specified string type property
Selection	
Ticket	Gets the ticket/selects the order
SelectByIndex	Selects the order by index

TimeSetup

Gets the time of order placement.

```
datetime TimeSetup() const
```

Returned value

Time of order placement.

Note

The historical order should be selected using the [Ticket](#) (by ticket) or [SelectByIndex](#) (by index) methods.

OrderType

Gets the order type.

```
ENUM_ORDER_TYPE OrderType() const
```

Returned value

Order type (value of [ENUM_ORDER_TYPE](#) enumeration) .

Note

The historical order should be selected using the [Ticket](#) (by ticket) or [SelectByIndex](#) (by index) methods.

TypeDescription

Gets the order type as a string.

```
string TypeDescription() const
```

Returned value

Order type as a string.

Note

The historical order should be selected using the [Ticket](#) (by ticket) or [SelectByIndex](#) (by index) methods.

State

Gets the order state.

```
ENUM_ORDER_STATE State() const
```

Returned value

Order state (value of [ENUM_ORDER_STATE](#) enumeration) .

Note

The historical order should be selected using the [Ticket](#) (by ticket) or [SelectByIndex](#) (by index) methods.

StateDescription

Gets the order state as a string.

```
string StateDescription() const
```

Returned value

Order state as a string.

Note

The historical order should be selected using the [Ticket](#) (by ticket) or [SelectByIndex](#) (by index) methods.

TimeExpiration

Gets the time of order expiration.

```
datetime TimeExpiration() const
```

Returned value

Time of order expiration, set on its placement.

Note

The historical order should be selected using the [Ticket](#) (by ticket) or [SelectByIndex](#) (by index) methods.

TimeDone

Gets the time of order execution or cancellation.

```
datetime TimeDone() const
```

Returned value

Time of order execution or cancellation.

Note

The historical order should be selected using the [Ticket](#) (by ticket) or [SelectByIndex](#) (by index) methods.

TypeFilling

Gets the type of order execution by remainder.

```
ENUM_ORDER_TYPE_FILLING TypeFilling() const
```

Returned value

Type of order execution by remainder (value of [ENUM_ORDER_TYPE_FILLING](#) enumeration) .

Note

The historical order should be selected using the [Ticket](#) (by ticket) or [SelectByIndex](#) (by index) methods.

TypeFillingDescription

Gets the type of order execution by remainder as a string.

```
string TypeFillingDescription() const
```

Returned value

Type order of execution by remainder as a string.

Note

The historical order should be selected using the [Ticket](#) (by ticket) or [SelectByIndex](#) (by index) methods.

TypeTime

Gets the type of order at the time of the expiration.

```
ENUM_ORDER_TYPE_TIME TypeTime() const
```

Returned value

Type of order at the time of the expiration (value of [ENUM_ORDER_TYPE_TIME](#) enumeration) .

Note

The historical order should be selected using the [Ticket](#) (by ticket) or [SelectByIndex](#) (by index) methods.

TypeTimeDescription

Gets the order type by expiration time as a string.

```
string TypeTimeDescription() const
```

Returned value

Order type by expiration time as a string.

Note

The historical order should be selected using the [Ticket](#) (by ticket) or [SelectByIndex](#) (by index) methods.

Magic

Gets the ID of the Expert Advisor, that placed the order.

```
long Magic() const
```

Returned value

ID of the Expert Advisor, that placed the order.

Note

The historical order should be selected using the [Ticket](#) (by ticket) or [SelectByIndex](#) (by index) methods.

PositionId

Gets the ID of position.

```
long PositionId() const
```

Returned value

ID of position, in which the order was involved.

Note

The historical order should be selected using the [Ticket](#) (by ticket) or [SelectByIndex](#) (by index) methods.

VolumeInitial

Gets the initial volume of order.

```
double VolumeInitial() const
```

Returned value

Initial volume of order.

Note

The historical order should be selected using the [Ticket](#) (by ticket) or [SelectByIndex](#) (by index) methods.

VolumeCurrent

Gets the unfilled volume of order.

```
double VolumeCurrent() const
```

Returned value

Unfilled volume of order.

Note

The historical order should be selected using the [Ticket](#) (by ticket) or [SelectByIndex](#) (by index) methods.

PriceOpen

Gets the order price.

```
double PriceOpen() const
```

Returned value

Price of order placement.

Note

The historical order should be selected using the [Ticket](#) (by ticket) or [SelectByIndex](#) (by index) methods.

StopLoss

Gets the Stop Loss price of the order.

```
double StopLoss() const
```

Returned value

Stop Loss price of the order.

Note

The historical order should be selected using the [Ticket](#) (by ticket) or [SelectByIndex](#) (by index) methods.

TakeProfit

Gets the the Take Profit price of the order.

```
double TakeProfit() const
```

Returned value

The Take Profit price of the order.

Note

The historical order should be selected using the [Ticket](#) (by ticket) or [SelectByIndex](#) (by index) methods.

PriceCurrent

Gets the current price of the order's symbol.

```
double PriceCurrent() const
```

Returned value

The current price of order's symbol.

Note

The historical order should be selected using the [Ticket](#) (by ticket) or [SelectByIndex](#) (by index) methods.

PriceStopLimit

Gets the stop limit price of the order.

```
double PriceStopLimit() const
```

Returned value

Stop Limit price of the order.

Note

The historical order should be selected using the [Ticket](#) (by ticket) or [SelectByIndex](#) (by index) methods.

Symbol

Gets the name of order symbol.

```
string Symbol() const
```

Returned value

Name of order symbol.

Note

The historical order should be selected using the [Ticket](#) (by ticket) or [SelectByIndex](#) (by index) methods.

Comment

Gets the order comment.

```
string Comment() const
```

Returned value

Order comment.

Note

The historical order should be selected using the [Ticket](#) (by ticket) or [SelectByIndex](#) (by index) methods.

InfoInteger

Gets the value of specified integer type property.

```
bool InfoInteger (
    ENUM_ORDER_PROPERTY_INTEGER prop_id,    // property ID
    long& var                                // reference to variable
) const
```

Parameters

prop_id

[in] ID of integer type property (value of [ENUM_ORDER_PROPERTY_INTEGER](#) enumeration) .

var

[out] Reference to long type variable to place result.

Returned value

true - in case of success, false - if unable to get property value.

Note

The historical order should be selected using the [Ticket](#) (by ticket) or [SelectByIndex](#) (by index) methods.

InfoDouble

Gets the value of specified double type property.

```
bool InfoDouble(  
    ENUM_ORDER_PROPERTY_DOUBLE prop_id,    // property ID  
    double& var                            // reference to variable  
) const
```

Parameters

prop_id

[in] ID of double type property (value of [ENUM_ORDER_PROPERTY_DOUBLE](#) enumeration) .

var

[out] Reference to double type variable to place result.

Returned value

true - in case of success, false - if unable to get property value.

Note

The historical order should be selected using the [Ticket](#) (by ticket) or [SelectByIndex](#) (by index) methods.

InfoString

Gets the value of specified string type property.

```
bool InfoString(  
    ENUM_ORDER_PROPERTY_STRING prop_id,    // property ID  
    string& var                            // reference to variable  
) const
```

Parameters

prop_id

[in] ID of text property (value of [ENUM_ORDER_PROPERTY_STRING](#) enumeration) .

var

[out] Reference to string type variable to place result.

Returned value

true - in case of success, false - if unable to get property value.

Note

The historical order should be selected using the [Ticket](#) (by ticket) or [SelectByIndex](#) (by index) methods.

Ticket (Get method)

Gets the order ticket.

```
ulong Ticket() const
```

Returned value

Order ticket.

Ticket (Set method)

Select the order for further work.

```
void Ticket(  
    ulong ticket    // order ticket  
)
```

Parameters

ticket

[in] Order ticket.

SelectByIndex

Selects an order by index for further access to its properties.

```
bool  SelectByIndex(  
    int  index    // order index  
)
```

Returned value

true - in case of success, false - if unable to select order.

CPositionInfo

CPositionInfo is a class for easy access to the open position properties.

Description

CPositionInfo class provides easy access to the open position properties.

Declaration

```
class CPositionInfo : public CObject
```

Title

```
#include <Trade\PositionInfo.mqh>
```

Class methods by groups

Access to integer type properties	
Time	Gets the time of position opening
PositionType	Gets the position type
TypeDescription	Gets the position type as a string
Magic	Gets the ID of expert, that opened the position
Identifier	Gets the ID of position
Access to double type properties	
Volume	Gets the volume of position
PriceOpen	Gets the price of position opening
StopLoss	Gets the price of position's Stop Loss
TakeProfit	Gets the price of position's Take Profit
PriceCurrent	Gets the current price by position symbol
Commission	Gets the amount of commission by position
Swap	Gets the amount of swap by position
Profit	Gets the amount of current profit by position
Access to text properties	
Symbol	Gets the name of position symbol
Access to MQL5 API functions	
InfoInteger	Gets the value of specified integer type property

InfoDouble	Gets the value of specified double type property
InfoString	Gets the value of specified string type property
Selection	
Select	Selects the position
SelectByIndex	Selects the position by index
State	
StoreState	Saves the position parameters
CheckState	Checks the current parameters against the saved parameters

Time

Gets the time of position opening.

```
datetime Time() const
```

Returned value

Time of position opening.

Note

The position should be selected using the [Select](#) (by ticket) or [SelectByIndex](#) (by index) methods.

PositionType

Gets the position type.

```
ENUM_POSITION_TYPE PositionType() const
```

Returned value

Position type (value of [ENUM_POSITION_TYPE](#) enumeration) .

Note

The position should be selected using the [Select](#) (by ticket) or [SelectByIndex](#) (by index) methods.

TypeDescription

Gets the position type as a string.

```
string TypeDescription() const
```

Returned value

Position type as a string.

Note

The position should be selected using the [Select](#) (by ticket) or [SelectByIndex](#) (by index) methods.

Magic

Gets the ID of Expert Advisor, opened the position.

```
long Magic() const
```

Returned value

ID of the Expert Advisor, opened the position.

Note

The position should be selected using the [Select](#) (by ticket) or [SelectByIndex](#) (by index) methods.

Identifier

Gets the ID of position.

```
long Identifier() const
```

Returned value

ID of position.

Note

The position should be selected using the [Select](#) (by ticket) or [SelectByIndex](#) (by index) methods.

Volume

Gets the volume of position.

```
double Volume() const
```

Returned value

Volume of position.

Note

The position should be selected using the [Select](#) (by ticket) or [SelectByIndex](#) (by index) methods.

PriceOpen

Gets the price of position opening.

```
double PriceOpen() const
```

Returned value

Position open price.

Note

The position should be selected using the [Select](#) (by ticket) or [SelectByIndex](#) (by index) methods.

StopLoss

Gets the Stop Loss price of the position.

```
double StopLoss() const
```

Returned value

The Stop Loss price of the position.

Note

The position should be selected using the [Select](#) (by ticket) or [SelectByIndex](#) (by index) methods.

TakeProfit

Gets the Take Profit price of the position.

```
double TakeProfit() const
```

Returned value

The Take Profit price of the position.

Note

The position should be selected using the [Select](#) (by ticket) or [SelectByIndex](#) (by index) methods.

PriceCurrent

Gets the current price by position symbol.

```
double PriceCurrent() const
```

Returned value

Current price by position symbol.

Commission

Gets the amount of commission of the position.

```
double Commission() const
```

Returned value

Amount of commission of the position (in deposit currency) .

Note

The position should be selected using the [Select](#) (by ticket) or [SelectByIndex](#) (by index) methods.

Swap

Gets the amount of swap of the position.

```
double Swap() const
```

Returned value

Amount of swap of the position (in deposit currency) .

Note

The position should be selected using the [Select](#) (by ticket) or [SelectByIndex](#) (by index) methods.

Profit

Gets the amount of current profit of the position.

```
double Profit() const
```

Returned value

Amount of current profit of the position (in deposit currency) .

Note

The position should be selected using the [Select](#) (by ticket) or [SelectByIndex](#) (by index) methods.

Symbol

Gets the name of position symbol.

```
string Symbol() const
```

Returned value

Name of position symbol.

Note

The position should be selected using the [Select](#) (by ticket) or [SelectByIndex](#) (by index) methods.

InfoInteger

Gets the value of specified integer type property.

```
bool InfoInteger (
    ENUM_POSITION_PROPERTY_INTEGER prop_id,    // property ID
    long& var                                // reference to variable
) const
```

Parameters

prop_id

[in] ID of integer type property (value of [ENUM_POSITION_PROPERTY_INTEGER](#) enumeration) .

var

[out] Reference to long type variable to place result.

Returned value

true - in case of success, false - if unable to get property value.

Note

The position should be selected using the [Select](#) (by ticket) or [SelectByIndex](#) (by index) methods.

InfoDouble

Gets the value of specified double type property.

```
bool   InfoDouble (
    ENUM_POSITION_PROPERTY_DOUBLE prop_id,    // property ID
    double&          var                      // reference to variable
) const
```

Parameters

prop_id

[in] ID of double type property (value of [ENUM_POSITION_PROPERTY_DOUBLE](#) enumeration) .

var

[in] Reference to double type variable to place result.

Returned value

true - in case of success, false - if unable to get property value.

Note

The position should be selected using the [Select](#) (by ticket) or [SelectByIndex](#) (by index) methods.

InfoString

Gets the value of specified string type property.

```
bool InfoString(  
    ENUM_POSITION_PROPERTY_STRING prop_id,    // property ID  
    string& var                                // reference to variable  
) const
```

Parameters

prop_id

[in] ID of text property (value of [ENUM_POSITION_PROPERTY_STRING](#) enumeration) .

var

[out] Reference to string type variable to place result.

Returned value

true - in case of success, false - if unable to get property value.

Note

The position should be selected using the [Select](#) (by ticket) or [SelectByIndex](#) (by index) methods.

Select

Select the position for further work.

```
bool Select(  
    const string symbol    // symbol  
)
```

Parameters

symbol

[in] Symbol for position selection.

SelectByIndex

Selects the position by index for further access to its properties.

```
bool  SelectByIndex(  
    int  index    // position index  
)
```

Returned value

true - in case of success, false - if unable to select position.

StoreState

Saves the position parameters.

```
void StoreState()
```

Returned value

None.

CheckState

Checks the current parameters against the saved parameters.

```
bool CheckState()
```

Returned value

true - if the position parameters have changed since the last call of the [StoreState\(\)](#) method,
otherwise - false.

CDealInfo

CDealInfo is a class for easy access to the deal properties.

Description

CDealInfo class provides access to the deal properties.

Declaration

```
class CDealInfo : public CObject
```

Title

```
#include <Trade\DealInfo.mqh>
```

Class methods by groups

Access to integer type properties	
Order	Gets the order by which the deal is executed
Time	Gets the time of deal execution
DealType	Gets the deal type
TypeDescription	Gets the deal type as a string
Entry	Gets the deal direction
EntryDescription	Gets the deal direction as a string
Magic	Gets the ID of expert, that executed the deal
PositionId	Gets the ID of position, in which the deal was involved
Access to double type properties	
Volume	Gets the volume of deal
Price	Gets the deal price
Commision	Gets the amount of commission by deal
Swap	Gets the amount of swap when position is closed
Profit	Gets the financial result of deal
Access to text properties	
Symbol	Gets the name of deal symbol
Comment	Gets the deal comment
Access to MQL5 API functions	

InfoInteger	Gets the value of specified integer type property
InfoDouble	Gets the value of specified double type property
InfoString	Gets value of specified string type property
Selection	
Ticket	Gets ticket/selects the deal
SelectByIndex	Selects the deal by index

Order

Gets the order by which the deal is executed.

```
long Order() const
```

Returned value

Order by which the deal is executed.

Note

The deal should be selected using the [Ticket](#) (by ticket) or [SelectByIndex](#) (by index) methods.

Time

Gets the time of deal execution.

```
datetime Time() const
```

Returned value

Time of deal execution.

Note

The deal should be selected using the [Ticket](#) (by ticket) or [SelectByIndex](#) (by index) methods.

DealType

Gets the deal type.

```
ENUM_DEAL_TYPE DealType() const
```

Returned value

Deal type (value of [ENUM_DEAL_TYPE](#) enumeration) .

Note

The deal should be selected using the [Ticket](#) (by ticket) or [SelectByIndex](#) (by index) methods.

TypeDescription

Gets the deal type as a string.

```
string TypeDescription() const
```

Returned value

Deal type as a string.

Note

The deal should be selected using the [Ticket](#) (by ticket) or [SelectByIndex](#) (by index) methods.

Entry

Gets the deal direction.

```
ENUM_DEAL_ENTRY Entry() const
```

Returned value

Deal direction (value of [ENUM_DEAL_ENTRY](#) enumeration.) .

Note

The deal should be selected using the [Ticket](#) (by ticket) or [SelectByIndex](#) (by index) methods.

EntryDescription

Gets the deal direction as a string.

```
string EntryDescription() const
```

Returned value

Deal direction as a string.

Note

The deal should be selected using the [Ticket](#) (by ticket) or [SelectByIndex](#) (by index) methods.

Magic

Gets the ID of the Expert Advisor, that executed the deal.

```
long Magic() const
```

Returned value

ID of the Expert Advisor, that executed the deal.

Note

The deal should be selected using the [Ticket](#) (by ticket) or [SelectByIndex](#) (by index) methods.

PositionId

Gets the ID of position, in which the deal was involved.

```
long PositionId() const
```

Returned value

ID of position, in which the deal was involved.

Note

The deal should be selected using the [Ticket](#) (by ticket) or [SelectByIndex](#) (by index) methods.

Volume

Gets the volume of deal.

```
double Volume() const
```

Returned value

Volume of deal.

Note

The deal should be selected using the [Ticket](#) (by ticket) or [SelectByIndex](#) (by index) methods.

Price

Gets the deal price.

```
double Price() const
```

Returned value

Deal price.

Note

The deal should be selected using the [Ticket](#) (by ticket) or [SelectByIndex](#) (by index) methods.

Commission

Gets the amount of commission of the deal.

```
double Commission() const
```

Returned value

Amount of commission of the deal.

Note

The deal should be selected using the [Ticket](#) (by ticket) or [SelectByIndex](#) (by index) methods.

Swap

Gets the amount of swap when position is closed.

```
double Swap() const
```

Returned value

Amount of swap when position is closed.

Note

The deal should be selected using the [Ticket](#) (by ticket) or [SelectByIndex](#) (by index) methods.

Profit

Gets the financial result of the deal.

```
double Profit() const
```

Returned value

Financial result of the deal (in deposit currency) .

Note

The deal should be selected using the [Ticket](#) (by ticket) or [SelectByIndex](#) (by index) methods.

Symbol

Gets the name of the deal symbol.

```
string Symbol() const
```

Returned value

Name of the deal symbol.

Note

The deal should be selected using the [Ticket](#) (by ticket) or [SelectByIndex](#) (by index) methods.

Comment

Gets the deal comment.

```
string Comment() const
```

Returned value

Deal comment.

Note

The deal should be selected using the [Ticket](#) (by ticket) or [SelectByIndex](#) (by index) methods.

InfoInteger

Gets the value of specified integer type property.

```
bool InfoInteger (
    ENUM_DEAL_PROPERTY_INTEGER prop_id,    // property ID
    long& var                               // reference to variable
) const
```

Parameters

prop_id

[in] ID of integer type property (value of [ENUM_DEAL_PROPERTY_INTEGER](#) enumeration) .

var

[out] Reference to long type variable to place result.

Returned value

true - in case of success, false - if unable to get property value.

Note

The deal should be selected using the [Ticket](#) (by ticket) or [SelectByIndex](#) (by index) methods.

InfoDouble

Gets the value of specified double type property.

```
bool InfoDouble(  
    ENUM_DEAL_PROPERTY_DOUBLE prop_id,    // property ID  
    double& var                            // reference to variable  
) const
```

Parameters

prop_id

[in] ID of double type property (value of [ENUM_DEAL_PROPERTY_DOUBLE](#) enumeration) .

var

[in] Reference to double type variable to place result.

Returned value

true - in case of success, false - if unable to get property value.

Note

The deal should be selected using the [Ticket](#) (by ticket) or [SelectByIndex](#) (by index) methods.

InfoString

Gets the value of specified string type property.

```
bool InfoString(  
    ENUM_DEAL_PROPERTY_STRING prop_id,    // property ID  
    string& var                            // reference to variable  
) const
```

Parameters

prop_id

[in] ID of text property (value of [ENUM_DEAL_PROPERTY_STRING](#) enumeration) .

var

[out] Reference to string type variable to place result.

Returned value

true - in case of success, false - if unable to get property value.

Note

The deal should be selected using the [Ticket](#) (by ticket) or [SelectByIndex](#) (by index) methods.

Ticket (Get method)

Gets the deal ticket.

```
ulong Ticket() const
```

Returned value

Deal ticket.

Ticket (Set method)

Select the position for further work.

```
void Ticket(  
    ulong ticket    // ticket  
)
```

Parameters

ticket

[in] Deal ticket.

SelectByIndex

Selects the deal by index for further access to its properties.

```
bool  SelectByIndex(  
    int  index    // order index  
)
```

Returned value

true - in case of success, false - if unable to select the deal.

CTrade

CTrade is a class for easy access to the trade functions.

Description

CTrade class provides easy access to the trade functions.

Declaration

```
class CTrade : public CObject
```

Title

```
#include <Trade\Trade.mqh>
```

Class methods by groups

Setting parameters	
LogLevel	Sets logging level
SetExpertMagicNumber	Sets the expert ID
SetDeviationInPoints	Sets the allowed deviation
SetTypeFilling	Sets filling type of the order
Operations with orders	
OrderOpen	Places the pending order with set parameters
OrderModify	Modifies the pending order parameters
OrderDelete	Deletes the pending order
Operations with positions	
PositionOpen	Opens the position with set parameters
PositionModify	Modifies the position parameters
PositionClose	Closes the position
Additional methods	
Buy	Opens a long position with specified parameters
Sell	Opens a short position with specified parameters
BuyLimit	Places the pending order of Buy Limit type with specified parameters
BuyStop	Places the pending order of Buy Stop type with specified parameters
SellLimit	Places the pending order of Sell Limit type with specified parameters

SellStop	Places the pending order of Sell Stop type with specified parameters
Access to the last request parameters	
Request	Gets the copy of the last request structure
RequestAction	Gets the trade operation type
RequestActionDescription	Gets the trade operation type as string
RequestMagic	Gets the magic number of the Expert Advisor
RequestOrder	Gets the order ticket, used in the last request
RequestSymbol	Gets the name of the symbol, used in the last request
RequestVolume	Gets the trade volume (in lots) , used in the last request
RequestPrice	Gets the price, used in the last request
RequestStopLimit	Gets the price of pending order of Stop Limit type, used in the last request
RequestSL	Gets the Stop Loss price of the order, used in the last request
RequestTP	Gets the Take Profit price of the order, used in the last request
RequestDeviation	Gets the price deviation of the order, used in the last request
RequestType	Gets the type of the order, used in the last request
RequestTypeDescription	Gets the type of the order (as string) , used in the last request
RequestTypeFilling	Gets the filling type of the order, used in the last request
RequestTypeFillingDescription	Gets the filling type of the order (as string) , used in the last request
RequestTypeTime	Gets the validity period of the order, used in the last request
RequestTypeTimeDescription	Gets the validity period of the order (as string) , used in the last request
RequestExpiration	Gets the expiration time of the order, used in the last request
RequestComment	Gets the comment of the order, used in the last request
Access to the last request checking results	
CheckResult	Gets the copy of the structure of the last request check result.
CheckResultRetcode	Gets the value of the retcode field of MqlTradeCheckResult type, filled while checking of the request correctness
CheckResultRetcodeDescription	Gets the string description of the retcode field of MqlTradeCheckResult type, filled while checking of the request correctness
CheckResultBalance	Gets the value of the balance field of MqlTradeCheckResult type,

	filled while checking of the request correctness
<u>CheckResultEquity</u>	Gets the value of the equity field of <u>MqlTradeCheckResult</u> type, filled while checking of the request correctness
<u>CheckResultProfit</u>	Gets the value of the profit field of <u>MqlTradeCheckResult</u> type, filled while checking of the request correctness
<u>CheckResultMargin</u>	Gets the value of the margin field of <u>MqlTradeCheckResult</u> type, filled while checking of the request correctness
<u>CheckResultMarginFree</u>	Gets the value of the margin_free field of <u>MqlTradeCheckResult</u> type, filled while checking of the request correctness
<u>CheckResultMarginLevel</u>	Gets the value of the margin_level field of <u>MqlTradeCheckResult</u> type, filled while checking of the request correctness
<u>CheckResultComment</u>	Gets the value of the comment field of <u>MqlTradeCheckResult</u> type, filled while checking of the request correctness
Access to the last request execution results	
<u>Result</u>	Gets the copy of the structure of the last request result
<u>ResultRetcode</u>	Gets the code of request result
<u>ResultRetcodeDescription</u>	Gets the code of request result as text
<u>ResultDeal</u>	Gets the deal ticket
<u>ResultOrder</u>	Gets the order ticket
<u>ResultVolume</u>	Gets the volume of deal or order
<u>ResultPrice</u>	Gets the price, confirmed by broker
<u>ResultBid</u>	Gets the current bid price (the requote)
<u>ResultAsk</u>	Gets the current ask price (the requote)
<u>ResultComment</u>	Gets the broker comment
Auxiliary methods	
<u>PrintRequest</u>	Prints the last request parameters into journal
<u>PrintResult</u>	Prints the results of the last request into journal
<u>FormatRequest</u>	Prepares the formatted string with last request parameters
<u>FormatRequestResult</u>	Prepares the formatted string with results of the last request execution

LogLevel

Sets logging level for messages.

```
void LogLevel(  
    int log_level    // logging level  
)
```

Parameters

log_level
[in] Logging level.

Returned value

None.

Note

Log_level = 0 - logging disabled (used in optimization mode) .
Log_level = 1 - logging error messages (default) .
Log_level = 2 - logging all messages (used in testing mode) .

SetExpertMagicNumber

Sets the expert ID.

```
void SetExpertMagicNumber (  
    ulong magic      // ID  
)
```

Parameters

magic

[in] New ID of the expert.

Returned value

None.

SetDeviationInPoints

Sets the allowed deviation.

```
void SetDeviationInPoints(  
    ulong deviation    // deviation  
)
```

Parameters

deviation

[in] Allowed deviation.

Returned value

None.

SetTypeFilling

Sets filling type of the order.

```
void SetTypeFilling(  
    ENUM_ORDER_TYPE_FILLING filling    // order filling type  
)
```

Parameters

filling

[in] Order filling type (value of [ENUM_ORDER_TYPE_FILLING](#) enumeration) .

Returned value

.

OrderOpen

Places the pending order with set parameters.

```
bool OrderOpen(
    const string      symbol,           // symbol
    ENUM_ORDER_TYPE   order_type,      // order type
    double            volume,           // order volume
    double             limit_price,      // StopLimit price
    double             price,            // execution price
    double             sl,               // Stop Loss price
    double             tp,               // Take Profit price
    ENUM_ORDER_TYPE_TIME type_time,     // type by expiration
    datetime           expiration,       // expiration
    const string       comment=""       // comment
)
```

Parameters

symbol

[in] Name of trade instrument.

order_type

[in] Type of order trade operation (value of [ENUM_ORDER_TYPE](#) enumeration) .

volume

[in] Requested order volume.

limit_price

[in] Price at which the StopLimit order will be placed.

price

[in] Price at which the order must be executed.

sl

[in] Price at which the Stop Loss will trigger.

tp

[in] Price at which the Take Profit will trigger.

type_time

[in] Order type by execution (value of [ENUM_ORDER_TYPE_TIME](#) enumeration) .

expiration

[in] Expiration date of pending order.

comment=""

[in] Order comment.

Returned value

true - in case of successful check of the basic structures, otherwise - false.

Note

Successful completion of the `OrderSend(...)` method does not always mean successful execution of the trade operation. It's necessary to check the result of trade request (trade server return code) using [ResultRetcode\(\)](#) and value, returned by [ResultOrder\(\)](#).

OrderModify

Modifies the pending order parameters.

```
bool OrderModify(  
    ulong          ticket,          // order ticket  
    double         price,           // execution price  
    double         sl,              // Stop Loss price  
    double         tp,              // Take Profit price  
    ENUM_ORDER_TYPE_TIME type_time, // type by expiration  
    datetime       expiration       // expiration  
)
```

Parameters

ticket

[in] Order ticket.

price

[in] The new price by which the order must be executed (or the previous value, if the change is not necessary) .

sl

[in] The new price by which the Stop Loss will trigger (or the previous value, if the change is not necessary) .

tp

[in] The new price by which the Take Profit will trigger (or the previous value, if the change is not necessary) .

type_time

[in] The new type of order by expiration (or the previous value, if the change is not necessary) , value of [ENUM_ORDER_TYPE_TIME](#) enumeration.

expiration

[in] The new expiration date of pending order (or the previous value, if the change is not necessary) .

Returned value

true - in case of successful check of the basic structures, otherwise - false.

Note

Successful completion of the OrderModify(...) method does not always mean successful execution of the trade operation. It's necessary to check the result of trade request (trade server return code) using [ResultRetcode\(\)](#).

OrderDelete

Deletes the pending order.

```
bool OrderDelete(  
    ulong ticket    // order ticket  
)
```

Parameters

ticket

[in] Order ticket.

Returned value

true - in case of successful check of the basic structures, otherwise - false.

Note

Successful completion of the `OrderDelete(...)` method does not always mean successful execution of the trade operation. It's necessary to check the result of trade request (trade server return code) using [ResultRetcode\(\)](#).

PositionOpen

Opens the position with set parameters.

```
bool PositionOpen(  
    const string      symbol,           // symbol  
    ENUM_ORDER_TYPE  order_type,       // position type  
    double            volume,           // position volume  
    double            price,            // execution price  
    double            sl,               // Stop Loss price  
    double            tp,               // Take Profit price  
    const string      comment=""        // comment  
)
```

Parameters

symbol

[in] Name of trade instrument, by which it is intended to open position.

order_type

[in] Type of position trade operation (value of [ENUM_ORDER_TYPE](#) enumeration) .

volume

[in] Requested position volume.

price

[in] Price at which the position must be opened.

sl

[in] Price at which the Stop Loss will trigger.

tp

[in] Price at which the Take Profit will trigger.

comment=""

[in] Position comment.

Returned value

true - in case of successful check of the basic structures, otherwise - false.

Note

Successful completion of the `PositionOpen(...)` method does not always mean successful execution of the trade operation. It's necessary to check the result of trade request (trade server return code) using [ResultRetcode\(\)](#) and value, returned by [ResultDeal\(\)](#).

PositionModify

Modifies the position parameters by specified symbol.

```
bool PositionModify(  
    const string symbol,      // symbol  
    double      sl,          // Stop Loss price  
    double      tp           // Take Profit price  
)
```

Parameters

symbol

[in] Name of trade instrument, by which it is intended to modify position.

sl

[in] The new price by which the Stop Loss will trigger (or the previous value, if the change is not necessary) .

tp

[in] The new price by which the Take Profit will trigger (or the previous value, if the change is not necessary) .

Returned value

true - in case of successful check of the basic structures, otherwise - false.

Note

Successful completion of the PositionModify(...) method does not always mean successful execution of the trade operation. It's necessary to check the result of trade request (trade server return code) using [ResultRetcode\(\)](#).

PositionClose

Closes the position by specified symbol.

```
bool PositionClose(  
    const string  symbol,           // symbol  
    ulong        deviation=ULONG_MAX // deviation  
)
```

Parameters

symbol

[in] Name of trade instrument, by which it is intended to close position.

deviation=ULONG_MAX

[in] Maximal deviation from the current price (in points) .

Returned value

true - in case of successful check of the basic structures, otherwise - false.

Note

Successful completion of the `PositionClose(...)` method does not always mean successful execution of the trade operation. It's necessary to check the result of trade request (trade server return code) using [ResultRetcode\(\)](#).

Buy

Opens a long position with specified parameters.

```
bool Buy(  
    double      volume,           // position volume  
    const string symbol=NULL,     // symbol  
    double      price=0.0,        // price  
    double      sl=0.0,           // stop loss price  
    double      tp=0.0,           // take profit price  
    const string comment=""       // comment  
)
```

Parameters

volume

[in] Position volume.

symbol=NULL

[in] Position symbol. If the symbol isn't specified, the current symbol will be used.

price=0.0

[in] Price. If the price isn't specified, the current market Ask price will be used.

sl=0.0

[in] Stop Loss price.

tp=0.0

[in] Take Profit price.

comment=""

[in] Comment.

Returned value

true - in case of successful check of the structures, otherwise - false.

Note

Successful completion of the `Buy(...)` method does not always mean successful execution of the trade operation. It's necessary to check the result of trade request (trade server [return code](#)) using [ResultRetcode\(\)](#) and value, returned by [ResultDeal\(\)](#).

Sell

Opens a short position with specified parameters.

```
bool Sell(
    double      volume,           // position volume
    const string symbol=NULL,     // symbol
    double      price=0.0,        // price
    double      sl=0.0,           // stop loss price
    double      tp=0.0,           // take profit price
    const string comment=""       // comment
)
```

Parameters

volume

[in] Position volume.

symbol=NULL

[in] Position symbol. If the symbol isn't specified, the current symbol will be used.

price=0.0

[in] Price. If the price isn't specified, the current market Bid price will be used.

sl=0.0

[in] Stop Loss price.

tp=0.0

[in] Take Profit price.

comment=""

[in] Comment.

Returned value

true - in case of successful check of the structures, otherwise - false.

Note

Successful completion of the `Sell(...)` method does not always mean successful execution of the trade operation. It's necessary to check the result of trade request (trade server [return code](#)) using [ResultRetcode\(\)](#) and value, returned by [ResultDeal\(\)](#).

BuyLimit

Places the pending order of Buy Limit type (buy at the price, lower than current market price) with specified parameters.

```
bool BuyLimit (
    double          volume,          // order volume
    double          price,           // order price
    const string    symbol=NULL,     // symbol
    double          sl=0.0,          // stop loss price
    double          tp=0.0,          // take profit price
    ENUM_ORDER_TYPE_TIME type_time=ORDER_TIME_GTC, // order lifetime
    datetime        expiration=0,    // order expiration time
    const string    comment=""       // comment
)
```

Parameters

volume

[in] Order volume.

price

[in] Order price.

symbol=NULL

[in] Order symbol. If the symbol isn't specified, the current symbol will be used.

sl=0.0

[in] Stop Loss price.

tp=0.0

[in] Take Profit price.

type_time=ORDER_TIME_GTC

[in] Order lifetime (value of [ENUM_ORDER_TYPE_TIME](#) enumeration) .

expiration=0

[in] Order expiration time (used only if type_time=[ORDER_TIME_SPECIFIED](#)) .

comment=""

[in] Order comment.

Returned value

true - in case of successful check of the structures, otherwise - false.

Note

Successful completion of the `BuyLimit(...)` method does not always mean successful execution of the trade operation. It's necessary to check the result of trade request (trade server [return code](#)) using [ResultRetcode\(\)](#) and value, returned by [ResultDeal\(\)](#).

BuyStop

Places the pending order of Buy Stop type (buy at the price, higher than current market price) with specified parameters.

```
bool BuyStop(
    double          volume,           // order volume
    double          price,            // order price
    const string    symbol=NULL,      // symbol
    double          sl=0.0,           // stop loss price
    double          tp=0.0,           // take profit price
    ENUM_ORDER_TYPE_TIME type_time=ORDER_TIME_GTC, // order lifetime
    datetime        expiration=0,     // order expiration time
    const string    comment=""        // comment
)
```

Parameters

volume

[in] Order volume.

price

[in] Order price.

symbol=NULL

[in] Order symbol. If the symbol isn't specified, the current symbol will be used.

sl=0.0

[in] Stop Loss price.

tp=0.0

[in] Take Profit price.

type_time=ORDER_TIME_GTC

[in] Order lifetime (value of [ENUM_ORDER_TYPE_TIME](#) enumeration) .

expiration=0

[in] Order expiration time (used only if type_time=ORDER_TIME_SPECIFIED) .

comment=""

[in] Order comment.

Returned value

true - in case of successful check of the structures, otherwise - false.

Note

Successful completion of the BuyStop(...) method does not always mean successful execution of the trade operation. It's necessary to check the result of trade request (trade server [return code](#)) using [ResultRetcode\(\)](#) and value, returned by [ResultDeal\(\)](#).

SellLimit

Places the pending order of Sell Limit type (sell at the price, higher than current market price) with specified parameters.

```
bool SellLimit(
    double          volume,           // order volume
    double          price,           // order price
    const string    symbol=NULL,     // symbol
    double          sl=0.0,          // stop loss price
    double          tp=0.0,          // take profit price
    ENUM_ORDER_TYPE_TIME type_time=ORDER_TIME_GTC, // order lifetime
    datetime        expiration=0,    // order expiration time
    const string    comment=""       // comment
)
```

Parameters

volume

[in] Order volume.

price

[in] Order price.

symbol=NULL

[in] Order symbol. If the symbol isn't specified, the current symbol will be used.

sl=0.0

[in] Stop Loss price.

tp=0.0

[in] Take Profit price.

type_time=ORDER_TIME_GTC

[in] Order lifetime (value of [ENUM_ORDER_TYPE_TIME](#) enumeration) .

expiration=0

[in] Order expiration time (used only if type_time=ORDER_TIME_SPECIFIED) .

comment=""

[in] Order comment.

Returned value

true - in case of successful check of the structures, otherwise - false.

Note

Successful completion of the SellLimit(...) method does not always mean successful execution of the trade operation. It's necessary to check the result of trade request (trade server [return code](#)) using [ResultRetcode\(\)](#) and value, returned by [ResultDeal\(\)](#).

SellStop

Places the pending order of Buy Stop type (sell at the price, lower than current market price) with specified parameters.

```
bool SellStop(
    double          volume,           // order volume
    double          price,            // order price
    const string    symbol=NULL,      // symbol
    double          sl=0.0,           // stop loss price
    double          tp=0.0,           // take profit price
    ENUM_ORDER_TYPE_TIME type_time=ORDER_TIME_GTC, // order lifetime
    datetime        expiration=0,     // order expiration time
    const string    comment=""        // comment
)
```

Parameters

volume

[in] Order volume.

price

[in] Order price.

symbol=NULL

[in] Order symbol. If the symbol isn't specified, the current symbol will be used.

sl=0.0

[in] Stop Loss price.

tp=0.0

[in] Take Profit price.

type_time=ORDER_TIME_GTC

[in] Order lifetime (value of [ENUM_ORDER_TYPE_TIME](#) enumeration) .

expiration=0

[in] Order expiration time (used only if type_time=ORDER_TIME_SPECIFIED) .

comment=""

[in] Order comment.

Returned value

true - in case of successful check of the structures, otherwise - false.

Note

Successful completion of the SellStop(...) method does not always mean successful execution of the trade operation. It's necessary to check the result of trade request (trade server [return code](#)) using [ResultRetcode\(\)](#) and value, returned by [ResultDeal\(\)](#).

Request

Gets the copy of the last request structure.

```
void Request(  
    MqlTradeRequest& request // target structure  
    ) const
```

Parameters

request

[out] Reference to the structure of [MqlTradeRequest](#) type.

Returned value

None.

RequestAction

Gets the trade operation type.

```
ENUM_TRADE_REQUEST_ACTIONS RequestAction() const
```

Returned value

Trade operation type, used in the last request.

RequestActionDescription

Gets the trade operation type as string.

```
string RequestActionDescription() const
```

Returned value

Trade operation type (as string) , used in the last request.

RequestMagic

Gets the magic number of the Expert Advisor.

```
ulong RequestMagic() const
```

Returned value

The magic number (ID) of the Expert Advisor, used in the last request.

RequestOrder

Gets the order ticket, used in the last request.

```
ulong RequestOrder() const
```

Returned value

Order ticket of the last request.

RequestSymbol

Gets the name of the symbol, used in the last request.

```
string RequestSymbol() const
```

Returned value

The name of the symbol, used in the last request.

RequestVolume

Gets the trade volume (in lots) , used in the last request.

```
double RequestVolume() const
```

Returned value

The trade volume (in lots) , used in the last request.

RequestPrice

Gets the price, used in the last request.

```
double RequestPrice() const
```

Returned value

Order price, used in the last request.

RequestStopLimit

Gets the price of pending order of Stop Limit type, used in the last request.

```
double RequestStopLimit() const
```

Returned value

The price of pending order of Stop Limit type, used in the last request.

RequestSL

Gets the Stop Loss price of the order, used in the last request.

```
double RequestSL() const
```

Returned value

The Stop Loss price, used in the last request.

RequestTP

Gets the Take Profit price of the order, used in the last request.

```
double RequestTP() const
```

Returned value

The Take Profit price, used in the last request.

RequestDeviation

Gets the price deviation of the order, used in the last request.

```
ulong RequestDeviation() const
```

Returned value

The price deviation of the order, used in the last request.

RequestType

Gets the type of the order, used in the last request.

```
ENUM_ORDER_TYPE RequestType() const
```

Returned value

Order type, used in the last request (value of [ENUM_ORDER_TYPE](#) enumeration) .

RequestTypeDescription

Gets the type of the order (as string) , used in the last request.

```
string RequestTypeDescription() const
```

Returned value

The order type (as string) , used in the last request.

RequestTypeFilling

Gets the filling type of the order, used in the last request.

```
ENUM_ORDER_TYPE_FILLING RequestTypeFilling() const
```

Returned value

The filling type of the order (value of [ENUM_ORDER_TYPE_FILLING](#)) , used in the last request.

RequestTypeFillingDescription

Gets the filling type of the order (as string) , used in the last request.

```
string RequestTypeFillingDescription() const
```

Returned value

The filling type (as string) of the order, used in the last request.

RequestTypeTime

Gets the validity period of the order, used in the last request.

```
ENUM_ORDER_TYPE_TIME RequestTypeTime() const
```

Returned value

The validity period of the order (value of [ENUM_ORDER_TYPE_TIME](#) enumeration) , used in the last request.

RequestTypeTimeDescription

Gets the validity period of the order (as string) , used in the last request.

```
string RequestTypeTimeDescription() const
```

Returned value

The validity period of the order (as string) , used in the last request.

RequestExpiration

Gets the expiration time of the order, used in the last request.

```
datetime RequestExpiration() const
```

Returned value

The expiration time of the order, used in the last request.

RequestComment

Gets the comment of the order, used in the last request.

```
string RequestComment() const
```

Returned value

The comment of the order, used in the last request.

Result

Gets the copy of the structure of the last request result.

```
void Result(  
    MqlTradeResult& result    // reference  
    ) const
```

Parameters

result

[out] Reference to the structure of [MqlTradeRequest](#) type.

Returned value

None.

ResultRetcode

Gets the code of request result.

```
uint ResultRetcode() const
```

Returned value

The [Code](#) of request result.

ResultRetcodeDescription

Gets the code of request result as text.

```
string ResultRetcodeDescription() const
```

Returned value

Code of the last request result as text.

ResultDeal

Gets the deal ticket.

```
ulong ResultDeal() const
```

Returned value

Deal ticket, if the deal is executed.

ResultOrder

Gets the order ticket.

```
ulong ResultOrder() const
```

Returned value

Order ticket, if the order is placed.

ResultVolume

Gets the volume of deal or order.

```
double ResultVolume() const
```

Returned value

Volume of deal or order.

ResultPrice

Gets the price, confirmed by broker.

```
double ResultPrice() const
```

Returned value

Price, confirmed by the broker.

ResultBid

Gets the current bid price (the requote) .

```
double ResultBid() const
```

Returned value

Current bid price (the requote) .

ResultAsk

Gets the current ask price (the requote) .

```
double ResultAsk() const
```

Returned value

Current ask price (the requote) .

ResultComment

Gets the broker comment.

```
string ResultComment() const
```

Returned value

Broker comment to the operation.

CheckResult

Gets the copy of the structure of the last request check result.

```
void CheckResult(  
    MqlTradeCheckResult& check_result    // reference  
    ) const
```

Parameters

check_result

[out] Reference to the target structure of the [MqlTradeCheckResult](#) type.

Returned value

None.

CheckResultRetcode

Gets the value of the retcode field of [MqlTradeCheckResult](#) type, filled while checking of the request correctness.

```
uint CheckResultRetcode() const
```

Returned value

The value of the retcode field (error code) of [MqlTradeCheckResult](#) type, filled while checking of the request correctness.

CheckResultRetcodeDescription

Gets the string description of the retcode field of [MqlTradeCheckResult](#) type, filled while checking of the request correctness.

```
string ResultRetcodeDescription() const
```

Returned value

The string description of the retcode field (Error code) of [MqlTradeCheckResult](#) type, filled while checking of the request correctness.

CheckResultBalance

Gets the value of the balance field of [MqlTradeCheckResult](#) type, filled while checking of the request correctness.

```
double CheckResultBalance() const
```

Returned value

The value of the balance field (balance value that will be after the execution of the trade operation) of [MqlTradeCheckResult](#) type, filled while checking of the request correctness.

CheckResultEquity

Gets the value of the equity field of [MqlTradeCheckResult](#) type, filled while checking of the request correctness.

```
double CheckResultEquity() const
```

Returned value

The value of the equity field (equity value that will be after the execution of the trade operation) of [MqlTradeCheckResult](#) type, filled while checking of the request correctness.

CheckResultProfit

Gets the value of the profit field of [MqlTradeCheckResult](#) type, filled while checking of the request correctness.

```
double CheckResultProfit() const
```

Returned value

The value of the profit field (profit value that will be after the execution of the trade operation) of [MqlTradeCheckResult](#) type, filled while checking of the request correctness.

CheckResultMargin

Gets the value of the margin field of [MqlTradeCheckResult](#) type, filled while checking of the request correctness.

```
double CheckResultMargin() const
```

Returned value

The value of the margin field (margin required for the trade operation) of [MqlTradeCheckResult](#) type, filled while checking of the request correctness.

CheckResultMarginFree

Gets the value of the margin_free field of [MqlTradeCheckResult](#) type, filled while checking of the request correctness.

```
double CheckResultMarginFree () const
```

Returned value

The value of the margin_free field (free margin that will be left after the execution of the trade operation) of [MqlTradeCheckResult](#) type, filled while checking of the request correctness.

CheckResultMarginLevel

Gets the value of the margin_level field of [MqlTradeCheckResult](#) type, filled while checking of the request correctness.

```
double CheckResultMarginLevel () const
```

Returned value

The value of the margin_level field (margin level that will be set after the execution of the trade operation) of [MqlTradeCheckResult](#) type, filled while checking of the request correctness.

CheckResultComment

The value of the comment field of [MqlTradeCheckResult](#) type, filled while checking of the request correctness.

```
string CheckResultComment () const
```

Returned value

The value of the comment field (Comment to the reply code, error description) of [MqlTradeCheckResult](#) type, filled while checking of the request correctness.

PrintRequest

Prints the last request parameters into journal.

```
void PrintRequest() const
```

Returned value

None.

PrintResult

Prints the results of the last request into journal.

```
void PrintResult() const
```

Returned value

None.

FormatRequest

Prepares the formatted string with last request parameters.

```
string FormatRequest(  
    string&          str,           // target string  
    const MqlTradeRequest& request // request  
) const
```

Parameters

str

[in] Target string, passed by reference.

request

[in] A structure of [MqlTradeRequest](#) type with parameters of the last request.

Returned value

None.

FormatRequestResult

Prepares the formatted string with results of the last request execution.

```
string FormatRequestResult(  
    string&          str,           // string  
    const MqlTradeRequest& request, // request structure  
    const MqlTradeResult& result    // result structure  
) const
```

Parameters

str

[in] Target string, passed by reference.

request

[in] A structure of [MqlTradeRequest](#) type with parameters of the last request.

result

[in] A structure of [MqlTradeResult](#) type with results of the last request.

Returned value

None.

Trading Strategy Classes

This section contains technical details of working with classes for creation and testing of trading strategies and description of the relevant components of the MQL5 standard library.

The use of these classes will save time when creating the trading strategies.

MQL5 Standard Library (in terms of trading strategies) is placed in the terminal directory, in the Include\Expert folder.

Base classes	Description
<u>CExpert</u>	Base class for Expert Advisor
<u>CExpertSignal</u>	Base class for Trading Signal classes
<u>CExpertTrailing</u>	Base class for Trailing Stop classes
<u>CExpertMoney</u>	Base class for Money Management classes

<u>CSignalAC</u>	The module of signals based on market models of the indicator Accelerator Oscillator.
<u>CSignalAMA</u>	The module of signals based on market models of the indicator Adaptive Moving Average.
<u>CSignalAO</u>	The module of signals based on market models of the indicator Awesome Oscillator.
<u>CSignalBearsPower</u>	The module of signals based on market models of the oscillator Bears Power.
<u>CSignalBullsPower</u>	The module of signals based on market models of the oscillator Bulls Power.
<u>CSignalCCI</u>	The module of signals based on market models of the oscillator Commodity Channel Index.
<u>CSignalDeM</u>	The module of signals based on market models of the oscillator DeMarker.
<u>CSignalDEMA</u>	The module of signals based on market models of the indicator Double Exponential Moving Average.
<u>CSignalEnvelopes</u>	The module of signals based on market models of the indicator Envelopes.
<u>CSignalFrAMA</u>	The module of signals based on market models of the indicator Fractal Adaptive Moving Average.
<u>CSignalITE</u>	The module of filtration of signals by time.
<u>CSignalMACD</u>	The module of signals based on market models of the oscillator MACD.

<u>CSignalMA</u>	The module of signals based on market models of the indicator Moving Average.
<u>CSignalSAR</u>	The module of signals based on market models of the indicator Parabolic SAR.
<u>CSignalRSI</u>	The module of signals based on market models of the oscillator Relative Strength Index.
<u>CSignalRVI</u>	The module of signals based on market models of the oscillator Relative Vigor Index.
<u>CSignalStoch</u>	The module of signals based on market models of the oscillator Stochastic.
<u>CSignalTRIX</u>	The module of signals based on market models of the oscillator Triple Exponential Average.
<u>CSignalTEMA</u>	The module of signals based on market models of the indicator Triple Exponential Moving Average.
<u>CSignalWPR</u>	The module of signals based on market models of the oscillator Williams Percent Range.

Trailing Stop classes	Description
<u>CTrailingFixedPips</u>	This class implements Trailing Stop algorithm based on fixed points
<u>CTrailingMA</u>	This class implements Trailing Stop algorithm based on the values of Moving Average indicator
<u>CTrailingNone</u>	A gag class, it doesn't uses any Trailing Stop algorithm
<u>CTrailingPSAR</u>	This class implements Trailing Stop algorithm based on the values of Parabolic SAR indicator

Money Management classes	Description
<u>CMoneyFixedLot</u>	A class with an algorithm, based on trading with predefined fixed lot size.
<u>CMoneyFixedMargin</u>	A class with an algorithm, based on trading with predefined fixed margin.
<u>CMoneyFixedRisk</u>	A class with an algorithm, based on trading with predefined risk.
<u>CMoneyNone</u>	A class with an algorithm, based on trading with minimal allowed lot size.
<u>CMoneySizeOptimized</u>	A class with an algorithm, based on trading with

	variable lot size, depending on the results of the previous deals.
--	--

Base classes for Expert Advisors

This section contains technical details of working with classes for creation and testing of trading strategies and description of the relevant components of the MQL5 standard library.

The use of these classes will save time when creating the trading strategies.

MQL5 Standard Library (in terms of trading strategies) is placed in the terminal directory, in the Include\Expert folder.

Class	Description
<u>CExpert</u>	Base class for Expert Advisor
<u>CExpertSignal</u>	Base class for Trading Signal classes
<u>CExpertTrailing</u>	Base class for Trailing Stop classes
<u>CExpertMoney</u>	Base class for Money Management classes

CExpert

CExpert is a base class for trading strategies. It has built-in algorithms for working with time series and indicators and a set of virtual methods for trading strategy.

How to use it:

1. Prepare an algorithm of the strategy;
2. Create your own class, inherited from CExpert class;
3. Override the virtual methods in your class with your own algorithms.

Description

The CExpert class is a set of virtual methods for implementation of trading strategies.

Declaration

```
class CExpert : public CObject
```

Title

```
#include <Expert\CExpert.mqh>
```

Class Methods

Public methods:

Initialization/Deinitialization	
<u>Init</u>	Class instance initialization method
virtual <u>InitSignal</u>	Initializes Trading Signal object
virtual <u>InitTrailing</u>	Initializes Trailing Stop object
virtual <u>InitMoney</u>	Initializes Money Management object
virtual <u>Deinit</u>	Class instance deinitialization method
Access to Protected Data	
<u>MaxOrders</u>	Gets/Sets maximal allowed number of orders
Event Processing Methods	
virtual <u>OnTick</u>	OnTick event handler
virtual <u>OnTrade</u>	OnTrade event handler
virtual <u>OnTimer</u>	OnTimer event handler

Protected methods:

Initialization/Deinitialization	
---------------------------------	--

virtual InitParameters	Parameters initialization method
virtual InitIndicators	Indicators initialization method
virtual InitTrade	Initializes Trade object
virtual DeinitTrade	Deinitializes Trade object
virtual DeinitSignal	Deinitializes Trading Signal object
virtual DeinitTrailing	Deinitializes Trailing Stop object
virtual DeinitMoney	Deinitializes Money Management object
virtual DeinitIndicators	Deinitializes Indicators
Update Methods	
virtual Refresh	Updates all data
Processing Methods	
virtual Processing	Main processing algorithm
Market Entry Methods	
virtual CheckOpen	Checks position opening conditions
virtual CheckOpenLong	Checks conditions to open long position
virtual CheckOpenShort	Checks conditions to open short position
virtual OpenLong	Opens the long position
virtual OpenShort	Opens the short position
Market Exit Methods	
virtual CheckClose	Checks conditions to close current position
virtual CheckCloseLong	Checks conditions to close long position
virtual CheckCloseShort	Checks conditions to close short position
virtual CloseAll	Closes the opened position and delete all orders
virtual Close	Closes the opened position
virtual CloseLong	Closes the long position
virtual CloseShort	Closes the short position
Position Reverse Methods	
virtual CheckReverse	Checks conditions to reverse opened position
virtual CheckReverseLong	Checks conditions to reverse long position
virtual CheckReverseShort	Checks conditions to reverse short position
virtual ReverseLong	Perform reverse operation of long position
virtual ReverseShort	Perform reverse operation of short position

Trailing Methods	
virtual CheckTrailingStop	Checks conditions to modify position parameters
virtual CheckTrailingStopLong	Checks Trailing Stop conditions of long position
virtual CheckTrailingStopShort	Checks Trailing Stop conditions of short position
virtual TrailingStopLong	Performs Trailing Stop for long position
virtual TrailingStopShort	Performs Trailing Stop for short position
virtual CheckTrailingOrderLong	Checks Trailing Stop conditions of buy limit/stop pending order
virtual CheckTrailingOrderShort	Checks Trailing Stop conditions of sell limit/stop pending order
virtual TrailingOrderLong	Performs Trailing Stop for buy limit/stop pending order
virtual TrailingOrderShort	Performs Trailing Stop for sell limit/stop pending order
Order Delete Methods	
virtual CheckDeleteOrderLong	Checks conditions to delete buy pending order
virtual CheckDeleteOrderShort	Checks conditions to delete sell pending order
virtual DeleteOrders	Deletes all orders
virtual DeleteOrder	Deletes the stop/limit pending order
virtual DeleteOrderLong	Deletes the buy limit/stop pending order
virtual DeleteOrderShort	Deletes the sell limit/stop pending order
Trade Volume Methods	
LotOpenLong	Gets trade volume for buy operation
LotOpenShort	Gets trade volume for sell operation
LotReverse	Gets trade volume for position reverse operation
Trade History Methods	
PrepareHistoryDate	Sets starting date for trade history tracking
HistoryPoint	Creates checkpoint of trade history (saves number of positions, orders, deals and historical orders)
CheckTradeState	Compares the current state with the saved one and calls the corresponding event handle
Event flags	
WaitEvent	Sets event waiting flag
NoWaitEvent	Resets event waiting flag
IsWaitingPositionOpened	Gets the flag of "Position opened" event tracking
IsWaitingPositionVolumeChanged	Gets the flag of "Volume changed" event tracking

<u>IsWaitingPositionModified</u>	Gets the flag of "Position modified" event tracking
<u>IsWaitingPositionClosed</u>	Gets the flag of "Position closed" event tracking
<u>IsWaitingPositionStopTake</u>	Gets the flag of "Position Stop Loss/Take Profit triggered" event tracking
<u>IsWaitingOrderPlaced</u>	Gets the flag of "Pending order placed" event tracking
<u>IsWaitingOrderModified</u>	Gets the flag of "Pending order modified" event tracking
<u>IsWaitingOrderDeleted</u>	Gets the flag of "Pending order deleted" event tracking
<u>IsWaitingOrderTriggered</u>	Gets the flag of "Pending order triggered" event tracking
Event Processing Methods	
virtual <u>TradeEventPositionStopTake</u>	"Position Stop Loss/Take Profit triggered" event handler
virtual <u>TradeEventOrderTriggered</u>	"Pending order triggered" event handler
virtual <u>TradeEventPositionOpened</u>	"Position opened" event handler
virtual <u>TradeEventPositionVolumeChanged</u>	"Position volume changed" event handler
virtual <u>TradeEventPositionModified</u>	"Position modified" event handler
virtual <u>TradeEventPositionClosed</u>	"Position closed" event handler
virtual <u>TradeEventOrderPlaced</u>	"Pending order placed" event handler
virtual <u>TradeEventOrderModified</u>	"Pending order modified" event handler
virtual <u>TradeEventOrderDeleted</u>	"Pending order deleted" event handler
virtual <u>TradeEventNotIdentified</u>	Event handler of the non-identified event
Service methods	
<u>TimeframeAdd</u>	Adds a timeframe to track
<u>TimeframesFlags</u>	Gets the flag indicating timeframes with a new bar

Init

Class instance initialization method.

```
bool Init(  
    string      symbol,          // Symbol  
    ENUM_TIMEFRAMES period,      // Period  
    bool        every_tick,      // Flag  
    long        magic==0        // Magic number  
)
```

Parameters

symbol

[in] Symbol.

period

[in] Period ([ENUM_TIMEFRAMES](#) enumeration) .

every_tick

[in] Flag.

magic==0

[in] Expert Advisor ID (Magic number) .

Returned value

true if successful, otherwise false.

InitSignal

Initializes Trading Signal object.

```
virtual bool InitSignal(  
    CExpertSignal* signal==NULL    // Pointer  
)
```

Parameters

signal==[NULL](#)

[in] Pointer to the [CExpertSignal](#) class object (or its heir) .

Returned value

true if successful, otherwise false.

Note

If signal is [NULL](#), the [CExpertSignal](#) class will be used, it does nothing.

InitTrailing

Initializes Trailing Stop object.

```
virtual bool InitTrailing(  
    CExpertTrailing* trailing==NULL    // pointer  
)
```

Parameters

trailing==[NULL](#)

[in] Pointer to the [CExpertTrailing](#) class object (or its heir) .

Returned value

true if successful, otherwise false.

Note

If trailing is [NULL](#), the [ExpertTrailing](#) class will be used, it does nothing.

InitMoney

Initializes the Money Management object.

```
virtual bool InitMoney(  
    CExpertMoney* money==NULL // pointer  
)
```

Parameters

money==[NULL](#)

[in] Pointer to the [CExpertMoney](#) class object (or its heir) .

Returned value

true if successful, otherwise false.

Note

If money is [NULL](#), the [ExpertMoney](#) class will be used, it does nothing.

Deinit

Class instance deinitialization method.

```
virtual void Deinit()
```

Returned value

None.

MaxOrders (Get Method)

Gets the maximal number of allowed orders.

```
int MaxOrders()
```

Returned value

Maximal number of allowed orders.

MaxOrders (Set Method)

Sets the maximal number of allowed orders.

```
void MaxOrders (  
    int max_orders    // max orders  
)
```

Parameters

max_orders

[in] New value of maximal allowed number of orders.

Note

By default, the maximal allowed number of orders is equal to 1.

OnTick

The OnTick event handler.

```
virtual void OnTick()
```

Returned value

None.

Note

It calls [Refresh](#) and [Processing](#) virtual methods.

OnTrade

The OnTrade event handler.

```
virtual void OnTrade ()
```

Returned value

None.

Note

It calls [CheckTradeState](#) virtual method.

OnTimer

The OnTimer event handler.

```
virtual void OnTimer ()
```

Returned value

None.

InitParameters

Initializes parameters.

```
virtual bool InitParameters()
```

Returned value

true if successful, otherwise false.

Note

The InitParameters() function of [CExpert](#) base class does nothing and always returns true.

InitIndicators

Initializes all indicators and time series.

```
virtual bool InitIndicators()
```

Returned value

true if successful, otherwise false.

Note

It calls consequentially InitIndicators() virtual methods of trading signal, trailing stop and money management objects.

InitTrade

Initializes Trade object.

```
virtual bool InitTrade(  
    long magic    // Magic number  
)
```

Parameters

magic

[in] Magic number of Expert Advisor, it will be used in trade operations.

Returned value

true if successful, otherwise false.

DeinitTrade

Deinitializes Trade object.

```
virtual void DeinitTrade()
```

Returned value

None.

DeinitSignal

Deinitializes Expert Signal object.

```
virtual void DeinitSignal()
```

Returned value

None.

DeinitTrailing

Deinitializes Trailing Stop object.

```
virtual void DeinitTrailing()
```

Returned value

None.

DeinitMoney

Deinitializes Money Management object.

```
virtual void DeinitMoney()
```

Returned value

None.

DeinitIndicators

Deinitializes all indicators and time series.

```
virtual void DeinitIndicators ()
```

Returned value

None.

Refresh

Updates all data.

```
virtual bool Refresh()
```

Returned value

true if tick processing is needed, otherwise false.

Note

It allows to determine the need of tick processing. If it needed, it updates all quotes and time series and indicators data and returns true.

Processing

Main processing algorithm.

```
virtual bool Processing()
```

Returned value

true if trade operation has been executed, otherwise false.

Note

It does the following steps:

1. Checks the presence of the opened position on the symbol. If there isn't opened position, skip steps ? 2, ? 3 and ? 4.
2. Checks conditions to reverse opened position (calls [CheckReverse](#) method) . If position has been "reversed", exit.
3. Checks conditions to close position (calls [CheckClose](#) method) . If position has been closed, skip step ? 4.
4. Checks conditions to modify position parameters (calls [CheckTrailingStop](#) method) . If position parameters has been modified, exit.
5. Check the presence of pending orders on the symbol. If there isn't any pending orders, go to step ? 9.
6. Checks condition to delete order (calls [CheckDeleteOrderLong](#) for buy pending orders or [CheckDeleteOrderShort](#) for sell pending orders) . If the order has been deleted, go to step ? 9.
7. Check conditions to modify pending order parameters (calls [CheckTrailingOrderLong](#) for buy orders or [CheckTrailingOrderShort](#) for sell orders) . If the order parameters has been modified, exit.
8. Exit.
9. Checks conditions to open position (calls [CheckOpen](#) method) .

Implementation

```
//+-----+
//| Main function                                     |
//| INPUT:  no.                                       |
//| OUTPUT: true-if any trade operation processed, false otherwise. |
//| REMARK: no.                                       |
//+-----+
bool CExpert::Processing()
{
//--- check if open positions
if(m_position.Select(m_symbol.Name()))
{
//--- open position is available
//--- check the possibility of closing the position/delete pending orders
if(!CheckClose())
{
//--- check the possibility of modifying the position
if(CheckTrailingStop()) return(true);
}
```

```
    }
}
//--- check the possibility of opening a position/setting pending order
if(CheckOpen()) return(true);
//--- check if plased pending orders
int total=OrdersTotal();
if(total!=0)
{
    for(int i=total-1;i>=0;i--)
    {
        m_order.Select(OrderGetTicket(i));
        if(m_order.Symbol()!=m_symbol.Name()) continue;
        if(m_order.Type()==ORDER_TYPE_BUY_LIMIT || m_order.Type()==ORDER_TYPE_BUY_STOP)
        {
            //--- check the ability to delete a pending order to buy
            if(CheckDeleteOrderLong()) return(true);
            //--- check the possibility of modifying a pending order to buy
            if(CheckTrailingOrderLong()) return(true);
        }
        else
        {
            //--- check the ability to delete a pending order to sell
            if(CheckDeleteOrderShort()) return(true);
            //--- check the possibility of modifying a pending order to sell
            if(CheckTrailingOrderShort()) return(true);
        }
    }
}
//--- return without operations
return(false);
}
```

CheckOpen

Checks conditions to open position.

```
virtual bool CheckOpen()
```

Returned value

true if trade operation has been executed, otherwise false.

Note

It checks conditions to open long ([CheckOpenLong](#)) and short ([CheckOpenShort](#)) positions.

Implementation

```
//+-----+  
//| Check for position open or limit/stop order set |  
//| INPUT: no. |  
//| OUTPUT: true-if trade operation processed, false otherwise. |  
//| REMARK: no. |  
//+-----+  
bool CExpert::CheckOpen()  
{  
    if(CheckOpenLong()) return(true);  
    if(CheckOpenShort()) return(true);  
    /--- return without operations  
    return(false);  
}
```


CheckOpenLong

Checks conditions to open long position.

```
virtual bool CheckOpenLong()
```

Returned value

true if trade operation has been executed, otherwise false.

Note

It checks conditions to open long position (by calling CheckOpenLong method of Expert Signal object) and opens the long position (by calling [OpenLong](#) method) if necessary.

Implementation

```
//+-----+
//| Check for long position open or limit/stop order set |
//| INPUT: no. |
//| OUTPUT: true-if trade operation processed, false otherwise. |
//| REMARK: no. |
//+-----+
bool CExpert::CheckOpenLong()
{
    double price,sl,tp;
    datetime expiration=TimeCurrent();
    ///--- check signal for long enter operations
    if(m_signal.CheckOpenLong(price,sl,tp,expiration))
    {
        if(!m_trade.SetOrderExpiration(expiration))
        {
            m_expiration=expiration;
        }
        return(OpenLong(price,sl,tp));
    }
    ///--- return without operations
    return(false);
}
```

CheckOpenShort

Checks conditions to open short position.

```
virtual bool CheckOpenShort()
```

Returned value

true if trade operation has been executed, otherwise false.

Note

It checks conditions to open short position (by calling CheckOpenShort method of Expert Signal object) and opens the short position (by calling [OpenShort](#) method) if necessary.

Implementation

```
//+-----+
//| Check for short position open or limit/stop order set |
//| INPUT: no. |
//| OUTPUT: true-if trade operation processed, false otherwise. |
//| REMARK: no. |
//+-----+
bool CExpert::CheckOpenShort()
{
    double price,sl,tp;
    datetime expiration=TimeCurrent();
    ///--- check signal for short enter operations
    if(m_signal.CheckOpenShort(price,sl,tp,expiration))
    {
        if(!m_trade.SetOrderExpiration(expiration))
        {
            m_expiration=expiration;
        }
        return(OpenShort(price,sl,tp));
    }
    ///--- return without operations
    return(false);
}
```

OpenLong

Opens the long position.

```
virtual bool OpenLong(  
    double price,      // Price  
    double sl,         // Stop Loss price  
    double tp         // Take Profit price  
)
```

Parameters

price

[in] Price.

sl

[in] Stop Loss price.

tp

[in] Take Profit price.

Returned value

true if trade operation has been executed, otherwise false.

Note

It gets trading volume (by calling [LotOpenLong](#) method) and opens the long position (by calling Buy method of Trade object) if trading volume is not equal to 0.

Implementation

```
//+-----+  
//| Long position open or limit/stop order set |  
//| INPUT: no. |  
//| OUTPUT: true-if trade operation processed, false otherwise. |  
//| REMARK: no. |  
//+-----+  
bool CExpert::OpenLong(double price,double sl,double tp)  
{  
    //--- get lot for open  
    double lot=LotOpenLong(price,sl);  
    //--- check lot for open  
    if(lot==0.0) return(false);  
    //---  
    return(m_trade.Buy(lot,price,sl,tp));  
}
```

OpenShort

Opens the short position.

```
virtual bool OpenShort(  
    double price,      // Price  
    double sl,         // Stop Loss price  
    double tp          // Take Profit price  
)
```

Parameters

price

[in] Price.

sl

[in] Stop Loss price.

tp

[in] Take Profit price.

Returned value

true if trade operation has been executed, otherwise false.

Note

It gets trading volume (by calling [LotOpenShort](#) method) and opens the short position (by calling Sell method of Trade object) if trading volume is not equal to 0.

Implementation

```
//+-----+  
//| Short position open or limit/stop order set |  
//| INPUT: no. |  
//| OUTPUT: true-if trade operation successful, false otherwise. |  
//| REMARK: no. |  
//+-----+  
bool CExpert::OpenShort(double price,double sl,double tp)  
{  
    //--- get lot for open  
    double lot=LotOpenShort(price,sl);  
    //--- check lot for open  
    if(lot==0.0) return(false);  
    //---  
    return(m_trade.Sell(lot,price,sl,tp));  
}
```

CheckClose

Checks conditions to close position.

```
virtual bool CheckClose()
```

Returned value

true if trade operation has been executed, otherwise false.

Note

1. It checks Expert Advisor Stop Out conditions (by calling CheckClose method of money management object) . If condition is satisfied, it closes the position and deletes all orders (by calling [CloseAll](#)) and exit.
2. It checks conditions to close long or short position (by calling [CheckCloseLong](#) or [CheckCloseShort](#) methods) and if position is closed, it deletes all orders (by calling [DeleteOrders](#) method) .

Implementation

```
//+-----+
//| Check for position close or limit/stop order delete |
//| INPUT: no. |
//| OUTPUT: true-if trade operation processed, false otherwise. |
//| REMARK: no. |
//+-----+
bool CExpert::CheckClose()
{
    double lot;
    //--- position must be selected before call
    if((lot=m_money.CheckClose(GetPointer(m_position)))!=0.0)
        return(CloseAll(lot));
    //--- check for position type
    if(m_position.Type()==POSITION_TYPE_BUY)
    {
        //--- check the possibility of closing the long position / delete pending order
        if(CheckCloseLong())
        {
            DeleteOrders();
            return(true);
        }
    }
    else
    {
        //--- check the possibility of closing the short position / delete pending order
        if(CheckCloseShort())
        {
            DeleteOrders();
        }
    }
}
```

```
        return(true);  
    }  
}  
//--- return without operations  
return(false);  
}
```

CheckCloseLong

Checks conditions to close long position.

```
virtual bool CheckCloseLong()
```

Returned value

true if trade operation has been executed, otherwise false.

Note

It checks conditions to close long position (by calling CheckCloseLong method of Expert Signal object) and if it satisfied, it closes the opened position (by calling [CloseLong](#) method) .

Implementation

```
//+-----+
//| Check for long position close or limit/stop order delete |
//| INPUT:  no. |
//| OUTPUT: true-if trade operation processed, false otherwise. |
//| REMARK: no. |
//+-----+
bool CExpert::CheckCloseLong()
{
    double price;
    ///--- check for long close operations
    if(m_signal.CheckCloseLong(price))
        return(CloseLong(price));
    ///--- return without operations
    return(false);
}
```

CheckCloseShort

Checks conditions to close short position.

```
virtual bool CheckCloseShort()
```

Returned value

true if trade operation has been executed, otherwise false.

Note

It checks conditions to close short position (by calling CheckCloseShort method of Expert Signal object) and if it satisfied, it closes the position (by calling [CloseShort](#) method) .

Implementation

```
//+-----+
//| Check for short position close or limit/stop order delete |
//| INPUT: no. |
//| OUTPUT: true-if trade operation processed, false otherwise. |
//| REMARK: no. |
//+-----+
bool CExpert::CheckCloseShort()
{
    double price;
    ///--- check for short close operations
    if(m_signal.CheckCloseShort(price))
        return(CloseShort(price));
    ///--- return without operations
    return(false);
}
```


CloseAll

It performs partial of full position closing.

```
virtual bool CloseAll(  
    double lot      // Lots to close  
)
```

Parameters

lot

[in] Number of lots to reduce the position.

Returned value

true if trade operation has been executed, otherwise false.

Note

It performs partial of full position closing (by calling the Sell and Buy methods of CTrade class object for the long/short positions) and deletes all orders (by calling the [DeleteOrders](#) method) .

Implementation

```
//+-----+  
//| Position close and orders delete |  
//| INPUT: lot. |  
//| OUTPUT: true-if trade operation processed, false otherwise. |  
//| REMARK: no. |  
//+-----+  
bool CExpert::CloseAll(double lot)  
{  
    bool result;  
    //--- check for close operations  
    if(m_position.Type()==POSITION_TYPE_BUY) result=m_trade.Sell(lot,0,0,0);  
    else result=m_trade.Buy(lot,0,0,0);  
    result|=DeleteOrders();  
    //---  
    return(result);  
}
```

Close

Closes the opened position.

```
virtual bool Close()
```

Returned value

true if trade operation has been executed, otherwise false.

Note

Closes the position (by calling PositionClose method of CTrade class object) .

Implementation

```
//+-----+  
//| Position close |  
//| INPUT: no. |  
//| OUTPUT: true-if trade operation processed, false otherwise. |  
//| REMARK: no. |  
//+-----+  
bool CExpert::Close()  
{  
    return(m_trade.PositionClose(m_symbol.Name()));  
}
```

CloseLong

Closes the long position.

```
virtual bool CloseLong(  
    double price    // price  
)
```

Parameters

price

[in] Closing price.

Returned value

true if trade operation has been executed, otherwise false.

Note

Closes the long position (by calling Sell method of CTrade class object) .

Implementation

```
//+-----+  
//| Long position close |  
//| INPUT: price. |  
//| OUTPUT: true-if trade operation processed, false otherwise. |  
//| REMARK: no. |  
//+-----+  
bool CExpert::CloseLong(double price)  
{  
    return(m_trade.Sell(m_position.Volume(),price,0,0));  
}
```

CloseShort

Closes the short position.

```
virtual bool CloseShort(  
    double price    // price  
)
```

Parameters

price

[in] Closing price.

Returned value

true if trade operation has been executed, otherwise false.

Note

Closes the short position (by calling Buy method of CTrade class object) .

Implementation

```
//+-----+  
//| Short position close |  
//| INPUT: price. |  
//| OUTPUT: true-if trade operation successful, false otherwise. |  
//| REMARK: no. |  
//+-----+  
bool CExpert::CloseShort(double price)  
{  
    return(m_trade.Buy(m_position.Volume(),price,0,0));  
}
```

CheckReverse

Checks conditions to reverse opened position.

```
virtual bool CheckReverse()
```

Returned value

true if trade operation has been executed, otherwise false.

Note

It checks conditions to reverse long ([CheckReverseLong](#)) and short ([CheckReverseShort](#)) positions.

Implementation

```
//+-----+
//| Check for position reverse |
//| INPUT: no. |
//| OUTPUT: true-if trade operation processed, false otherwise. |
//| REMARK: no. |
//+-----+
bool CExpert::CheckReverse()
{
    if(m_position.Type()==POSITION_TYPE_BUY)
    {
        //--- check the possibility of reverse the long position
        if(CheckReverseLong()) return(true);
    }
    else
        //--- check the possibility of reverse the short position
        if(CheckReverseShort()) return(true);
    //--- return without operations
    return(false);
}
```

CheckReverseLong

Checks conditions to reverse long position.

```
virtual bool CheckReverseLong()
```

Returned value

true if trade operation has been executed, otherwise false.

Note

It checks conditions to reverse long position (by calling CheckReverseLong method of Expert Signal object) and perform reverse operation of the current long position (by calling [ReverseLong](#) method) if necessary.

Implementation

```
//+-----+
//| Check for long position reverse |
//| INPUT: no. |
//| OUTPUT: true-if trade operation processed, false otherwise. |
//| REMARK: no. |
//+-----+
bool CExpert::CheckReverseLong()
{
    double price,sl,tp;
    datetime expiration=TimeCurrent();
    //-- check signal for long reverse operations
    if(m_signal.CheckReverseLong(price,sl,tp,expiration)) return(ReverseLong(price,sl,
    //-- return without operations
    return(false);
}
```

CheckReverseShort

Checks conditions to reverse short position.

```
virtual bool CheckReverseShort()
```

Returned value

true if trade operation has been executed, otherwise false.

Note

It checks conditions to reverse short position (by calling CheckReverseShort method of Expert Signal object) and perform reverse operation of the current short position (by calling [ReverseShort](#) method) if necessary.

Implementation

```
//+-----+
//| Check for short position reverse |
//| INPUT: no. |
//| OUTPUT: true-if trade operation processed, false otherwise. |
//| REMARK: no. |
//+-----+
bool CExpert::CheckReverseShort()
{
    double price,sl,tp;
    datetime expiration=TimeCurrent();
    //-- check signal for short reverse operations
    if(m_signal.CheckReverseShort(price,sl,tp,expiration)) return(ReverseShort(price,s
    //-- return without operations
    return(false);
}
```

ReverseLong

Perform reverse operation of long position.

```
virtual bool ReverseLong(
    double price,      // Price
    double sl,         // Stop Loss
    double tp          // Take Profit
)
```

Parameters

price

[in] Price of long position reverse.

sl

[in] Stop Loss price.

tp

[in] Take Profit price.

Returned value

true if trade operation has been executed, otherwise false.

Note

It gets position reverse volume (by calling [LotReverse](#) method) and perform trade operation of long position reverse (by calling Sell method of Trade object) if trading volume is not equal to 0.

Implementation

```
//+-----+
//| Long position reverse |
//| INPUT: price - price, |
//|          sl   - stop loss, |
//|          tp   - take profit. |
//| OUTPUT: true-if trade operation processed, false otherwise. |
//| REMARK: no. |
//+-----+
bool CExpert::ReverseLong(double price,double sl,double tp)
{
    //--- get lot for reverse
    double lot=LotReverse(sl);
    //--- check lot
    if(lot==0.0) return(false);
    //---
    return(m_trade.Sell(lot,price,sl,tp));
}
```


ReverseShort

Perform reverse operation of short position.

```
virtual bool ReverseShort(  
    double price,      // Price  
    double sl,         // Stop Loss  
    double tp          // Take Profit  
)
```

Parameters

price

[in] Price of short position reverse.

sl

[in] Stop Loss price.

tp

[in] Take Profit price.

Returned value

true if trade operation has been executed, otherwise false.

Note

It gets position reverse volume (by calling [LotReverse](#) method) and perform trade operation of short position reverse (by calling Buy method of Trade object) if trading volume is not equal to 0.

Implementation

```
//+-----+  
//| Short position reverse |  
//| INPUT: price - price, |  
//|          sl   - stop loss, |  
//|          tp   - take profit. |  
//| OUTPUT: true-if trade operation processed, false otherwise. |  
//| REMARK: no. |  
//+-----+  
bool CExpert::ReverseShort(double price,double sl,double tp)  
{  
    //--- get lot for reverse  
    double lot=LotReverse(sl);  
    //--- check lot  
    if(lot==0.0) return(false);  
    //---  
    return(m_trade.Buy(lot,price,sl,tp));  
}
```

CheckTrailingStop

It checks Trailing Stop conditions of the opened position.

```
virtual bool CheckTrailingStop()
```

Returned value

true if trade operation has been executed, otherwise false.

Note

It checks Trailing Stop conditions of the opened position (by calling [CheckTrailingStopLong](#) or [CheckTrailingStopShort](#) for long and short positions) .

Implementation

```
//+-----+
//| Check for trailing stop/profit position |
//| INPUT: no. |
//| OUTPUT: true-if trade operation processed, false otherwise. |
//| REMARK: no. |
//+-----+
bool CExpert::CheckTrailingStop()
{
    //--- position must be selected before call
    if(m_position.Type()==POSITION_TYPE_BUY)
    {
        //--- check the possibility of modifying the long position
        if(CheckTrailingStopLong()) return(true);
    }
    else
    {
        //--- check the possibility of modifying the short position
        if(CheckTrailingStopShort()) return(true);
    }
    //--- return without operations
    return(false);
}
```

CheckTrailingStopLong

It checks Trailing Stop conditions of the opened long position.

```
virtual bool CheckTrailingStopLong ()
```

Returned value

true if trade operation has been executed, otherwise false.

Note

It checks Trailing Stop conditions of the opened long position (by calling CheckTrailingStopLong method of Expert Trailing object) . If conditions are satisfied, it modifies the position parameters (by calling [TrailingStopLong](#) method) .

Implementation

```
//+-----+
//| Check for trailing stop/profit long position |
//| INPUT: no. |
//| OUTPUT: true-if trade operation processed, false otherwise. |
//| REMARK: no. |
//+-----+
bool CExpert::CheckTrailingStopLong()
{
    double sl=EMPTY_VALUE;
    double tp=EMPTY_VALUE;
    //--- check for long trailing stop operations
    if(m_trailing.CheckTrailingStopLong(GetPointer(m_position),sl,tp))
    {
        if(sl==EMPTY_VALUE) sl=m_position.StopLoss();
        if(tp==EMPTY_VALUE) tp=m_position.TakeProfit();
        //--- long trailing stop operations
        return(TrailingStopLong(sl,tp));
    }
    //--- return without operations
    return(false);
}
```

CheckTrailingStopShort

It checks Trailing Stop conditions of the opened short position.

```
virtual bool CheckTrailingStopShort()
```

Returned value

true if trade operation has been executed, otherwise false.

Note

It checks Trailing Stop conditions of the opened short position (by calling CheckTrailingStopShort method of Expert Trailing object) . If conditions are satisfied, it modifies the position parameters (by calling [TrailingStopShort](#) method) .

Implementation

```
//+-----+
//| Check for trailing stop/profit short position |
//| INPUT: no. |
//| OUTPUT: true-if trade operation processed, false otherwise. |
//| REMARK: no. |
//+-----+
bool CExpert::CheckTrailingStopShort()
{
    double sl=EMPTY_VALUE;
    double tp=EMPTY_VALUE;
    //--- check for short trailing stop operations
    if(m_trailing.CheckTrailingStopShort(GetPointer(m_position),sl,tp))
    {
        if(sl==EMPTY_VALUE) sl=m_position.StopLoss();
        if(tp==EMPTY_VALUE) tp=m_position.TakeProfit();
        //--- short trailing stop operations
        return(TrailingStopShort(sl,tp));
    }
    //--- return without operations
    return(false);
}
```

TrailingStopLong

The function modifies parameters of the opened long position.

```
virtual bool TrailingStopLong(  
    double sl,      // Stop Loss price  
    double tp       // Take Profit price  
)
```

Parameters

sl

[in] New Stop Loss price.

tp

[in] New Take Profit price.

Returned value

true if trade operation has been executed, otherwise false.

Note

The function modifies parameters of the opened long position (by calling PositionModify method of CTrade class object) .

Implementation

```
//+-----+  
//| Trailing stop/profit long position |  
//| INPUT: no. |  
//| OUTPUT: true-if trade operation successful, false otherwise. |  
//| REMARK: no. |  
//+-----+  
bool CExpert::TrailingStopLong(double sl,double tp)  
{  
    return(m_trade.PositionModify(m_symbol.Name(),sl,tp));  
}
```

TrailingStopShort

The function modifies parameters of the opened short position.

```
virtual bool TrailingStopShort(  
    double sl,      // Stop Loss price  
    double tp       // Take Profit price  
)
```

Parameters

sl

[in] New Stop Loss price.

tp

[in] New Take Profit price.

Returned value

true if trade operation has been executed, otherwise false.

Note

The function modifies parameters of the opened short position (by calling PositionModify method of CTrade class object) .

Implementation

```
//+-----+  
//| Trailing stop/profit short position |  
//| INPUT: no. |  
//| OUTPUT: true-if trade operation successful, false otherwise. |  
//| REMARK: no. |  
//+-----+  
bool CExpert::TrailingStopShort(double sl,double tp)  
{  
    return(m_trade.PositionModify(m_symbol.Name(),sl,tp));  
}
```

CheckTrailingOrderLong

It checks Trailing Stop conditions of buy limit/stop pending order.

```
virtual bool CheckTrailingOrderLong ()
```

Returned value

true if trade operation has been executed, otherwise false.

Note

It checks Trailing Stop conditions for buy limit/stop pending order (by calling CheckTrailingOrderLong method of Trade Signals object) and modifies the order parameters if necessary (by calling [TrailingOrderLong](#) method) .

Implementation

```
//+-----+
//| Check for trailing long limit/stop order |
//| INPUT:  no.                             |
//| OUTPUT: true-if trade operation processed, false otherwise. |
//| REMARK: no.                             |
//+-----+
bool CExpert::CheckTrailingOrderLong ()
{
    double price;
    ///--- check the possibility of modifying the long order
    if(m_signal.CheckTrailingOrderLong(GetPointer(m_order),price))
        return(TrailingOrderLong(price));
    ///--- return without operations
    return(false);
}
```

CheckTrailingOrderShort

It checks Trailing Stop conditions of sell limit/stop pending order.

```
virtual bool CheckTrailingOrderShort()
```

Returned value

true if trade operation has been executed, otherwise false.

Note

It checks Trailing Stop conditions for sell limit/stop pending order (by calling CheckTrailingOrderShort method of Trade Signals object) and modifies the order parameters if necessary (by calling [TrailingOrderShort](#) method) .

Implementation

```
//+-----+
//| Check for trailing short limit/stop order |
//| INPUT: no. |
//| OUTPUT: true-if trade operation processed, false otherwise. |
//| REMARK: no. |
//+-----+
bool CExpert::CheckTrailingOrderShort()
{
    double price;
    ///--- check the possibility of modifying the short order
    if(m_signal.CheckTrailingOrderShort(GetPointer(m_order),price))
        return(TrailingOrderShort(price));
    ///--- return without operations
    return(false);
}
```


TrailingOrderLong

The function modifies parameters of buy limit/stop pending order.

```
virtual bool TrailingOrderLong(
    double delta    // delta
)
```

Parameters

delta

[in] Price delta.

Returned value

true if trade operation has been executed, otherwise false.

Note

It modifies parameters of buy limit/stop order (by calling OrderModify method of CTrade class object) .

Implementation

```
//+-----+
//| Trailing long limit/stop order |
//| INPUT: no. |
//| OUTPUT: true-if trade operation successful, false otherwise. |
//| REMARK: no. |
//+-----+
bool CExpert::TrailingOrderLong(double delta)
{
    ulong ticket=m_order.Ticket();
    double price =m_order.PriceOpen()-delta;
    double sl    =m_order.StopLoss()-delta;
    double tp    =m_order.TakeProfit()-delta;
    ///--- modifying the long order
    return(m_trade.OrderModify(ticket,price,sl,tp,m_order.TypeTime(),m_order.TimeExpir
}
```

TrailingOrderShort

The function modifies parameters of sell limit/stop pending order.

```
virtual bool TrailingOrderShort(  
    double delta    // delta  
)
```

Parameters

delta

[in] Price delta.

Returned value

true if trade operation has been executed, otherwise false.

Note

It modifies parameters of sell limit/stop order (by calling OrderModify method of CTrade class object) .

Implementation

```
//+-----+  
//| Trailing short limit/stop order |  
//| INPUT: no. |  
//| OUTPUT: true-if trade operation successful, false otherwise. |  
//| REMARK: no. |  
//+-----+  
bool CExpert::TrailingOrderShort(double delta)  
{  
    ulong ticket=m_order.Ticket();  
    double price =m_order.PriceOpen()-delta;  
    double sl    =m_order.StopLoss()-delta;  
    double tp    =m_order.TakeProfit()-delta;  
    ///--- modifying the short order  
    return(m_trade.OrderModify(ticket,price,sl,tp,m_order.TypeTime(),m_order.TimeExpir  
}
```

CheckDeleteOrderLong

It checks conditions to delete the buy limit/stop pending order.

```
virtual bool CheckDeleteOrderLong()
```

Returned value

true if trade operation has been executed, otherwise false.

Note

The steps are:

Step 1. It checks the order expiration time.

Step 2. It checks conditions to delete the buy limit/stop pending order (by calling CheckCloseLong method of Expert Signal class object) and deletes the order if condition is satisfied (by calling [DeleteOrderLong](#) method) .

Implementation

```
//+-----+
//| Check for delete long limit/stop order |
//| INPUT: no. |
//| OUTPUT: true-if trade operation processed, false otherwise. |
//| REMARK: no. |
//+-----+
bool CExpert::CheckDeleteOrderLong()
{
    double price;
    ///--- check the possibility of deleting the long order
    if(m_expiration!=0 && TimeCurrent()>m_expiration)
    {
        m_expiration=0;
        return(DeleteOrderLong());
    }
    if(m_signal.CheckCloseLong(price))
        return(DeleteOrderLong());
    ///--- return without operations
    return(false);
}
```

CheckDeleteOrderShort

It checks conditions to delete the sell limit/stop pending order.

```
virtual bool CheckDeleteOrderShort ()
```

Returned value

true if trade operation has been executed, otherwise false.

Note

The steps are:

Step 1. It checks the order expiration time.

Step 2. It checks conditions to delete the sell limit/stop pending order (by calling CheckCloseShort method of Expert Signal class object) and deletes the order if condition is satisfied (by calling [DeleteOrderShort](#) method) .

Implementation

```
//+-----+
//| Check for delete short limit/stop order |
//| INPUT: no. |
//| OUTPUT: true-if trade operation processed, false otherwise. |
//| REMARK: no. |
//+-----+
bool CExpert::CheckDeleteOrderShort()
{
    double price;
    ///--- check the possibility of deleting the short order
    if(m_expiration!=0 && TimeCurrent()>m_expiration)
    {
        m_expiration=0;
        return(DeleteOrderShort());
    }
    if(m_signal.CheckCloseShort(price))
        return(DeleteOrderShort());
    ///--- return without operations
    return(false);
}
```

DeleteOrders

Deletes all orders.

```
virtual bool DeleteOrders()
```

Returned value

true if trade operation has been executed, otherwise false.

Note

It deletes all orders (by calling [DeleteOrder](#) method in the loop) .

Implementation

```
//+-----+
//| Delete all limit/stop orders |
//| INPUT: no. |
//| OUTPUT: true-if trade operation successful, false otherwise. |
//| REMARK: no. |
//+-----+
bool CExpert::DeleteOrders()
{
    bool result=false;
    int total=OrdersTotal();
    //---
    for(int i=total-1;i>=0;i--)
    {
        if(m_order.Select(OrderGetTicket(i)))
        {
            if(m_order.Symbol()!=m_symbol.Name()) continue;
            result|=DeleteOrder();
        }
    }
    //---
    return(result);
}
```

DeleteOrder

Deletes the limit/stop pending order.

```
virtual bool DeleteOrder()
```

Returned value

true if trade operation has been executed, otherwise false.

Note

It deletes the limit/stop pending order (by calling OrderDelete method of CTrade class object) .

Implementation

```
//+-----+
//| Delete limit/stop order |
//| INPUT: no. |
//| OUTPUT: true-if trade operation successful, false otherwise. |
//| REMARK: no. |
//+-----+
bool CExpert::DeleteOrder()
{
    return(m_trade.OrderDelete(m_order.Ticket()));
}
```

DeleteOrderLong

Deletes the buy limit/stop pending order.

```
virtual bool DeleteOrderLong()
```

Returned value

true if trade operation has been executed, otherwise false.

Note

It deletes the buy limit/stop pending order (by calling OrderDelete method of CTrade class object) .

Implementation

```
//+-----+
//| Delete long limit/stop order |
//| INPUT: no. |
//| OUTPUT: true-if trade operation successful, false otherwise. |
//| REMARK: no. |
//+-----+
bool CExpert::DeleteOrderLong()
{
    return(m_trade.OrderDelete(m_order.Ticket()));
}
```

DeleteOrderShort

Deletes the sell limit/stop pending order.

```
virtual bool DeleteOrderShort()
```

Returned value

true if trade operation has been executed, otherwise false.

Note

It deletes the sell limit/stop pending order (by calling OrderDelete method of CTrade class object) .

Implementation

```
//+-----+
//| Delete short limit/stop order |
//| INPUT: no. |
//| OUTPUT: true-if trade operation successful, false otherwise. |
//| REMARK: no. |
//+-----+
bool CExpert::DeleteOrderShort()
{
    return(m_trade.OrderDelete(m_order.Ticket()));
}
```


LotOpenLong

Gets trade volume for buy operation.

```
double LotOpenLong(  
    double price, // Price  
    double sl      // Stop Loss  
)
```

Parameters

sl

[in] Stop Loss price.

tp

[in] Take Profit price.

Returned value

Trade volume (in lots) for buy operation.

Note

It gets trade volume for buy operation (by calling CheckOpenLong method of money management object) .

Implementation

```
//+-----+  
//| Method of getting the lot for open long position. |  
//| INPUT: no. |  
//| OUTPUT: lot for open. |  
//| REMARK: no. |  
//+-----+  
double CExpert::LotOpenLong(double price,double sl)  
{  
    double lot=m_money.CheckOpenLong(price,sl);  
//---  
    return(lot);  
}
```

LotOpenShort

Gets trade volume for sell operation.

```
double LotOpenShort(  
    double price, // Price  
    double sl      // Stop Loss  
)
```

Parameters

sl

[in] Stop Loss price.

tp

[in] Take Profit price.

Returned value

Trade volume (in lots) for sell operation.

Note

It gets trade volume for sell operation (by calling CheckOpenLong method of money management object) .

Implementation

```
//+-----+  
//| Method of getting the lot for open short position. |  
//| INPUT: no. |  
//| OUTPUT: lot for open. |  
//| REMARK: no. |  
//+-----+  
double CExpert::LotOpenShort(double price,double sl)  
{  
    double lot=m_money.CheckOpenShort(price,sl);  
//---  
    return(lot);  
}
```

LotReverse

Gets trade volume for position reverse.

```
double LotReverse(  
    double sl      // Stop Loss  
)
```

Parameters

sl

[in] Stop Loss price.

Returned value

Trade volume (in lots) for position reverse operation.

Note

It gets trade volume for position reverse operation (by calling CheckReverse method of money management object) .

Implementation

```
//+-----+  
//| Method of getting the lot for reverse position. |  
//| INPUT:  sl - stop loss.                        |  
//| OUTPUT: lot for open.                          |  
//| REMARK: no.                                    |  
//+-----+  
double CExpert::LotReverse(double sl)  
{  
    return(m_money.CheckReverse(GetPointer(m_position),sl));  
}
```

PrepareHistoryDate

Sets starting date for the trade history.

```
void PrepareHistoryDate()
```

Note

The trade history tracking period is set from the beginning of the month (but not less than one day) .

HistoryPoint

Creates checkpoint of trade history (saves number of positions, orders, deals and historical orders) .

```
void HistoryPoint(  
    bool from_check_trade==false    // flag  
)
```

Parameters

from_check_trade==false
[in] flag to avoid the recursion.

Note

It saves the number of positions, orders, deals and historical orders.

CheckTradeState

Compares the current state with the saved one and calls the corresponding event handler.

```
bool CheckTradeState ()
```

Returned value

true if event has been handled, otherwise - false.

Note

It checks the number of positions, orders, deals and historical orders by comparing with the values, saved by [HistoryPoint](#) method. If trade history has changed, it calls the corresponding virtual event handler.

WaitEvent

Sets event waiting flag.

```
void WaitEvent(  
    ENUM_TRADE_EVENTS event // Flag  
)
```

Parameters

event

[in] Event to track.

NoWaitEvent

Resets event waiting flag.

```
void NoWaitEvent(  
    ENUM_TRADE_EVENTS event // Flag  
)
```

Parameters

event

[in] Event to reset.

IsWaitingPositionOpened

Returns true if Expert Advisor should handle the "Position Opened" event, otherwise false.

```
bool IsWaitingPositionOpened() const
```

Returned value

true if corresponding flag is set and event should handled, otherwise false. The event is processed by [TradeEventPositionOpened\(\)](#) method.

Note

The event tracking flag can be set/reset using [WaitEvent\(\)](#) and [NoWaitEvent\(\)](#) methods.

IsWaitingPositionVolumeChanged

Returns true if Expert Advisor should handle the "Position volume changed" event, otherwise false.

```
bool IsWaitingPositionVolumeChanged() const
```

Returned value

true if corresponding flag is set and event should handled, otherwise false. The event is processed by [TradeEventPositionVolumeChanged](#) method.

Note

The event tracking flag can be set/reset using [WaitEvent\(\)](#) and [NoWaitEvent\(\)](#) methods.

IsWaitingPositionModified

Returns true if Expert Advisor should handle the "Position modified" event, otherwise false.

```
bool IsWaitingPositionModified() const
```

Returned value

true if corresponding flag is set and event should handled, otherwise false. The event is processed by [TradeEventPositionModified](#) method.

Note

The event tracking flag can be set/reset using [WaitEvent\(\)](#) and [NoWaitEvent\(\)](#) methods.

IsWaitingPositionClosed

Returns true if Expert Advisor should handle the "Position closed" event, otherwise false.

```
bool IsWaitingPositionClosed() const
```

Returned value

true if corresponding flag is set and event should handled, otherwise false. The event is processed by [TradeEventPositionClosed](#) method.

Note

The event tracking flag can be set/reset using [WaitEvent\(\)](#) and [NoWaitEvent\(\)](#) methods.

IsWaitingPositionStopTake

Returns true if Expert Advisor should handle the "Position Stop Loss/Take Profit triggered" event, otherwise false.

```
bool IsWaitingPositionStopTake () const
```

Returned value

true if corresponding flag is set and event should handled, otherwise false. The event is processed by [TradeEventPositionStopTake](#) method.

Note

The event tracking flag can be set/reset using [WaitEvent\(\)](#) and [NoWaitEvent\(\)](#) methods.

IsWaitingOrderPlaced

Returns true if Expert Advisor should handle the "Pending order placed" event, otherwise false.

```
bool IsWaitingOrderPlaced() const
```

Returned value

true if corresponding flag is set and event should handled, otherwise false. The event is processed by [TradeEventOrderPlaced](#) method.

Note

The event tracking flag can be set/reset using [WaitEvent\(\)](#) and [NoWaitEvent\(\)](#) methods.

IsWaitingOrderModified

Returns true if Expert Advisor should handle the "Pending order modified" event, otherwise false.

```
bool IsWaitingOrderModified() const
```

Returned value

true if corresponding flag is set and event should handled, otherwise false. The event is processed by [TradeEventOrderModified](#) method.

Note

The event tracking flag can be set/reset using [WaitEvent\(\)](#) and [NoWaitEvent\(\)](#) methods.

TradeEventOrderDeleted

Returns true if Expert Advisor should handle the "Pending order deleted" event, otherwise false.

```
bool IsWaitingOrderDeleted() const
```

Returned value

true if corresponding flag is set and event should handled, otherwise false. The event is processed by [TradeEventOrderDeleted](#) method.

Note

The event tracking flag can be set/reset using [WaitEvent\(\)](#) and [NoWaitEvent\(\)](#) methods.

IsWaitingOrderTriggered

Returns true if Expert Advisor should handle the "Pending order triggered" event, otherwise false.

```
bool IsWaitingOrderTriggered() const
```

Returned value

true if corresponding flag is set and event should handled, otherwise false. The event is processed by [TradeEventOrderTriggered](#) method.

Note

The event tracking flag can be set/reset using [WaitEvent\(\)](#) and [NoWaitEvent\(\)](#) methods.

TradeEventPositionStopTake

"Position Stop Loss/Take Profit triggered" event handler.

```
virtual bool TradeEventPositionStopTake()
```

Returned value

true if event has been processed, otherwise false.

Note

The virtual method of [CExpert](#) class does nothing and always returns true.

TradeEventOrderTriggered

"Pending order triggered" event handler.

```
virtual bool TradeEventOrderTriggered()
```

Returned value

true if event has been processed, otherwise false.

Note

The virtual method of [CExpert](#) class does nothing and always returns true.

TradeEventPositionOpened

"Position opened" event handler.

```
virtual bool TradeEventPositionOpened()
```

Returned value

true if event has been processed, otherwise false.

Note

The virtual method of [CExpert](#) class does nothing and always returns true.

TradeEventPositionVolumeChanged

"Position volume changed" event handler.

```
virtual bool TradeEventPositionVolumeChanged()
```

Returned value

true if event has been processed, otherwise false.

Note

The virtual method of [CExpert](#) class does nothing and always returns true.

TradeEventPositionModified

"Position modified" event handler.

```
virtual bool TradeEventPositionModified()
```

Returned value

true if event has been processed, otherwise false.

Note

The virtual method of [CExpert](#) class does nothing and always returns true.

TradeEventPositionClosed

"Position closed" event handler.

```
virtual bool TradeEventPositionClosed()
```

Returned value

true if event has been processed, otherwise false.

Note

The virtual method of [CExpert](#) class does nothing and always returns true.

TradeEventOrderPlaced

"Pending order placed" event handler.

```
virtual bool TradeEventOrderPlaced()
```

Returned value

true if event has been processed, otherwise false.

Note

The virtual method of [CExpert](#) class does nothing and always returns true.

TradeEventOrderModified

"Pending order modified" event handler.

```
virtual bool TradeEventOrderModified()
```

Returned value

true if event has been processed, otherwise false.

Note

The virtual method of [CExpert](#) class does nothing and always returns true.

TradeEventOrderDeleted

"Pending order deleted" event handler.

```
virtual bool TradeEventOrderDeleted()
```

Returned value

true if event has been processed, otherwise false.

Note

The virtual method of [CExpert](#) class does nothing and always returns true.

TradeEventNotIdentified

Event handler of the non-identified event.

```
virtual bool TradeEventNotIdentified()
```

Returned value

true if event has been processed, otherwise false.

Note

The virtual method of [CExpert](#) class does nothing and always returns true.

TimeframeAdd

Add a timeframe for tracking.

```
void TimeframeAdd(  
    ENUM_TIMEFRAMES period    // Timeframe  
)
```

Parameters

period

[in] Timeframe to track.

TimeframesFlags

The function returns the flag indicating the timeframes with a new bar.

```
int TimeframesFlags(  
    MqlDateTime& time    // Variable for time  
)
```

Parameters

time

[in] Variable of [MqlDateTime](#) type for new time, passed by reference.

Returned value

It returns the flag, that indicates timeframes with a new bar.

CExpertSignal

CExpertSignal is a base class for trading signals, it does nothing (except [CheckReverseLong](#) and [CheckReverseShort](#) methods) but provides the interfaces.

How to use it:

1. Prepare an algorithm for trading signals;
2. Create your own trading signal class, inherited from CExpertSignal class;
3. Override the virtual methods in your class with your own algorithms.

You can find an examples of trading signal classes in the Expert\Signal\ folder.

Description

CExpertSignal is a base class for implementation of trading signal algorithms.

Declaration

```
class CExpertSignal : public CObject
```

Title

```
#include <Expert\ExpertSignal.mqh>
```

Class Methods

Initialization	
virtual Init	Initializes class members
virtual InitIndicators	Initializes indicators and time series
virtual ValidationSettings	Checks the settings
Check Trading Conditions	
virtual CheckOpenLong	Checks conditions to open long position
virtual CheckCloseLong	Checks conditions to close long position
virtual CheckOpenShort	Checks conditions to open short position
virtual CheckCloseShort	Checks conditions to close short position
virtual CheckReverseLong	Checks conditions of long position reversal
virtual CheckReverseShort	Checks conditions of short position reversal
Check Trailing Stop Conditions	
virtual CheckTrailingOrderLong	Checks conditions to modify buy pending order
virtual CheckTrailingOrderShort	Checks conditions to modify sell pending order

Init

Initializes class members.

```
virtual bool Init(  
    CSymbolInfo*    symbol,           // CSymbolInfo object pointer  
    ENUM_TIMEFRAMES period,         // Timeframe  
    double          adjusted_point   // Point size  
)
```

Parameters

symbol

[in] Pointer to [CSymbolInfo](#) object ([CExpert](#) class member) .

period

[in] Timeframe ([ENUM_TIMEFRAMES](#) enumeration) .

adjusted_point

[in] Point size.

Returned value

true if successful, otherwise false.

InitIndicators

Initializes indicators and time series.

```
virtual bool InitIndicators(  
    CIndicators* indicators // CIndicators collection pointer  
)
```

Parameters

indicators

[in] Pointer to indicators and time series collection ([CExpert](#) class member) .

Returned value

true if successful, otherwise false.

Note

The InitIndicators() method of the base class always returns true.

ValidationSettings

Checks the settings.

```
virtual bool ValidationSettings()
```

Returned value

true if successful, otherwise false.

Note

The ValidationSettings() method of the base class always returns true.

CheckOpenLong

Checks conditions to open long position.

```
virtual bool CheckOpenLong(  
    double& price,           // Price  
    double& sl,              // Stop Loss price  
    double& tp,              // Take Profit price  
    datetime& expiration    // Order expiration time  
)
```

Parameters

price

[in][out] Variable for price, passed by reference.

sl

[in][out] Variable for "Stop Loss" price, passed by reference.

tp

[in][out] Variable for "Take Profit" price, passed by reference.

expiration

[in][out] Variable for order expiration time (if necessary) , passed by reference.

Returned value

true if conditions are satisfied, otherwise false.

Note

The CheckOpenLong() method of the base class always returns false.

CheckCloseLong

Checks conditions to close long position.

```
virtual bool CheckCloseLong(  
    double& price      // Price  
)
```

Parameters

price

[in][out] Variable for price, passed by reference.

Returned value

true if conditions are satisfied, otherwise false.

Note

The CheckCloseLong() method of the base class always returns false.

CheckOpenShort

Checks conditions to open short position.

```
virtual bool CheckOpenShort(  
    double& price,           // Price  
    double& sl,              // Stop Loss price  
    double& tp,              // Take Profit price  
    datetime& expiration     // Order expiration time  
)
```

Parameters

price

[in][out] Variable for opening price, passed by reference.

sl

[in][out] Variable for "Stop Loss" price, passed by reference.

tp

[in][out] Variable for "Take Profit" price, passed by reference.

expiration

[in][out] Variable for order expiration time (if necessary) , passed by reference.

Returned value

true if conditions are satisfied, otherwise false.

Note

The CheckOpenShort() method of the base class always returns false.

CheckCloseShort

Checks conditions to close short position.

```
virtual bool CheckCloseShort(  
    double& price      // Closing price  
)
```

Parameters

price

[in][out] Variable for position closing price, passed by reference.

Returned value

true if condition is satisfied, otherwise false.

Note

The CheckCloseShort() method of the base class always returns false.

CheckReverseLong

Checks conditions of long position reversal.

```
virtual bool CheckReverseLong(  
    double& price,           // Reversal price  
    double& sl,              // Stop Loss price  
    double& tp,              // Take Profit price  
    datetime& expiration    // Order expiration time  
)
```

Parameters

price

[in][out] Variable for reversal price, passed by reference.

sl

[in][out] Variable for "Stop Loss" price, passed by reference.

tp

[in][out] Variable for "Take Profit" price, passed by reference.

expiration

[in][out] Variable for order expiration time (if necessary) , passed by reference.

Returned value

true if conditions are satisfied, otherwise false.

Note

The `CheckReverseLong()` method of the base class returns true if both [CheckCloseLong](#) and [CheckOpenShort](#) conditions are satisfied and closing and opening prices are equal.

CheckReverseShort

Checks conditions of short position reversal.

```
virtual bool CheckReverseShort(  
    double& price,           // Reversal price  
    double& sl,              // Stop Loss price  
    double& tp,              // Take Profit price  
    datetime& expiration    // Order expiration time  
)
```

Parameters

price

[in][out] Variable for reversal price, passed by reference.

sl

[in][out] Variable for "Stop Loss" price, passed by reference.

tp

[in][out] Variable for "Take Profit" price, passed by reference.

expiration

[in][out] Variable for order expiration time (if necessary) , passed by reference.

Returned value

true if conditions are satisfied, otherwise false.

Note

The `CheckReverseShort()` method of the base class returns true if both [CheckCloseShort](#) and [CheckOpenLong](#) conditions are satisfied and closing and opening prices are equal.

CheckTrailingOrderLong

Checks conditions to modify buy pending order.

```
virtual bool CheckTrailingOrderLong(  
    COrderInfo* order,      // COrderInfo object pointer  
    double& price           // New order price  
)
```

Parameters

order

[in] Pointer to [COrderInfo](#) object.

price

[in][out] Variable for new order price, passed by reference.

Returned value

true if conditions are satisfied, otherwise false.

Note

The `CheckTrailingOrderLong()` method of the base class always returns false.

CheckTrailingOrderShort

Checks conditions to modify sell pending order.

```
virtual bool CheckTrailingOrderShort(  
    COrderInfo* order,      // COrderInfo object pointer  
    double& price           // New order price  
)
```

Parameters

order

[in] Pointer to [COrderInfo](#) object.

price

[in][out] Variable for new order price, passed by reference.

Returned value

true if conditions are satisfied, otherwise false.

Note

The `CheckTrailingOrderShort()` method of the base class always returns false.

CExpertTrailing

CExpertTrailing is a base class for trailing algorithms, it does nothing but provides the interfaces.

How to use it:

1. Prepare an algorithm for trailing;
2. Create your own trailing class, inherited from CExpertTrailing class;
3. Override the virtual methods in your class with your own algorithms.

You can find an examples of trailing classes in the Expert\Trailing\ folder.

Description

CExpertTrailing is a base class for implementation of trailing algorithms.

Declaration

```
class CExpertTrailing : public CObject
```

Title

```
#include <Expert\ExpertTrailing.mqh>
```

Class Methods

Initialization	
virtual Init	Initializes class members
virtual InitIndicators	Initializes indicators and time series
virtual ValidationSettings	Checks the settings
Checking Trailing Stop	
virtual CheckTrailingStopLong	Checks conditions to modify long position
virtual CheckTrailingStopShort	Checks conditions to modify short position

Init

Initializes class members.

```
virtual bool Init(  
    CSymbolInfo*    symbol,           // CSymbolInfo object pointer  
    ENUM_TIMEFRAMES period,         // Timeframe  
    double          adjusted_point   // Point size  
)
```

Parameters

symbol

[in] Pointer to [CSymbolInfo](#) object ([CExpert](#) class member) .

period

[in] Timeframe.

adjusted_point

[in] Point size.

Returned value

true if successful, otherwise false.

InitIndicators

Initializes indicators and time series.

```
virtual bool InitIndicators(  
    CIndicators* indicators // CIndicators collection pointer  
)
```

Parameters

indicators

[in] A pointer to indicators and time series collection ([CExpert](#) class member) .

Returned value

true if successful, otherwise false.

Note

The InitIndicators() method of the base class always returns true.

ValidationSettings

Checks the settings.

```
virtual bool ValidationSettings()
```

Returned value

true if successful, otherwise false.

Note

The ValidationSettings() method of the base class always returns true.

CheckTrailingStopLong

Checks conditions to modify long position.

```
virtual bool CheckTrailingStopLong (
    CPositionInfo* position,      // CPositionInfo object pointer
    double& sl,                  // Stop Loss price
    double& tp                    // Take Profit price
)
```

Parameters

position

[in] Pointer to [CPositionInfo](#) object.

sl

[in][out] Variable for Stop Loss price, passed by reference.

tp

[in][out] Variable for Take Profit price, passed by reference.

Returned value

true if conditions are satisfied, otherwise false.

Note

The CheckTrailingStopLong() method of the base class always returns false.

CheckTrailingStopShort

Checks conditions to modify short position.

```
virtual bool CheckTrailingStopShort(  
    CPositionInfo* position,      // CPositionInfo object pointer  
    double& sl,                  // Stop Loss price  
    double& tp                    // Take Profit price  
)
```

Parameters

position

[in] Pointer to [CPositionInfo](#) object.

sl

[in][out] Variable for Stop Loss price, passed by reference.

tp

[in][out] Variable for Take Profit price, passed by reference.

Returned value

true if conditions are satisfied, otherwise false.

Note

The `CheckTrailingStopShort()` method of the base class always returns false.

CExpertMoney

CExpertMoney is a base class for money and risk management algorithms.

Description

CExpertMoney is a base class for implementation of money and risk management classes.

Declaration

```
class CExpertMoney : public CObject
```

Title

```
#include <Expert\ExpertMoney.mqh>
```

Class Methods

Access to Protected Data	
Percent	Sets the value of "Risk percent" parameter
Initialization	
virtual Init	Initializes class members
virtual InitIndicators	Initializes indicators and time series
virtual ValidationSettings	Checks the settings
Checking Trading Conditions	
virtual CheckOpenLong	Gets the volume for long position
virtual CheckOpenShort	Gets the volume for short position
virtual CheckReverse	Gets the volume for reverse of the position
virtual CheckClose	Checks conditions to close opened position

Percent

Sets the value of "Risk percent" parameter.

```
void Percent(  
    double percent    // Risk percent  
)
```

Parameters

percent

[in] Risk percent.

Returned value

None.

Init

Initializes class members.

```
virtual bool Init(  
    CSymbolInfo*    symbol,           // CSymbolInfo object pointer  
    ENUM_TIMEFRAMES period,          // Timeframe  
    double          adjusted_point    // Point size  
)
```

Parameters

symbol

[in] Pointer to [CSymbolInfo](#) object ([CExpert](#) class member) .

period

[in] Timeframe ([ENUM_TIMEFRAMES](#) enumeration) .

adjusted_point

[in] Point size.

Returned value

true if successful, otherwise false.

InitIndicators

Initializes indicators and time series.

```
virtual bool InitIndicators(  
    CIndicators* indicators // CIndicators collection pointer  
)
```

Parameters

indicators

[in] Pointer to indicators and time series collection ([CExpert](#) class member) .

Returned value

true if successful, otherwise false.

Note

The InitIndicators() method of the base class always returns true.

ValidationSettings

Checks the settings.

```
virtual bool ValidationSettings()
```

Returned value

true if successful, otherwise false.

Note

The ValidationSettings() method of the base class always returns true.

CheckOpenLong

Gets the volume for long position.

```
virtual double CheckOpenLong(  
    double price,      // Price  
    double sl          // Stop Loss price  
)
```

Parameters

price

[in] Opening price of long position.

sl

[in] Stop Loss price of long position.

Returned value

Trade volume for long position.

CheckOpenShort

Gets the volume for short position.

```
virtual double CheckOpenShort(  
    double price,      // Price  
    double sl          // Stop Loss price  
)
```

Parameters

price

[in] Opening price of short position.

sl

[in] Stop Loss price of short position.

Returned value

Trade volume for short position.

CheckReverse

Gets the volume for reverse of the position.

```
virtual double CheckReverse(  
    CPositionInfo* position,    // CPositionInfo object pointer  
    double sl                  // Stop Loss price  
)
```

Parameters

position

[in] Pointer to [CPositionInfo](#) object.

sl

[in] Stop Loss price.

Returned value

Volume for reverse of the position.

CheckClose

Checks conditions to close position.

```
virtual double CheckClose()
```

Returned value

true if condition is satisfied, otherwise false.

Modules of Trade Signals

The standard delivery of the client terminal includes a set of ready-made modules of trade signals for "MQL5 Wizard". When creating an Expert Advisor in MQL5 Wizard, you can use any combination of the modules of trade signals (up to 64) . The final decision on a trade operation is made on the basis of complex analysis of signals obtained from all included modules. The detailed description of the mechanism of making trade decisions is given [below](#).

The standard delivery includes the following modules of signals:

- [Signals of the Indicator Accelerator Oscillator](#)
- [Signals of the Indicator Adaptive Moving Average](#)
- [Signals of the Indicator Awesome Oscillator](#)
- [Signals of the Oscillator Bears Power](#)
- [Signals of the Oscillator Bulls Power](#)
- [Signals of the Oscillator Commodity Channel Index](#)
- [Signals of the Oscillator DeMarker](#)
- [Signals of the Indicator Double Exponential Moving Average](#)
- [Signals of the Indicator Envelopes](#)
- [Signals of the Indicator Fractal Adaptive Moving Average](#)
- [Signals of the Intraday Time Filter](#)
- [Signals of the Oscillator MACD](#)
- [Signals of the Indicator Moving Average](#)
- [Signals of the Indicator Parabolic SAR](#)
- [Signals of the Oscillator Relative Strength Index](#)
- [Signals of the Oscillator Relative Vigor Index](#)
- [Signals of the Oscillator Stochastic](#)
- [Signals of the Oscillator Triple Exponential Average](#)
- [Signals of the Indicator Triple Exponential Moving Average](#)
- [Signals of the Oscillator Williams Percent Range](#)

The Mechanism of Making Trade Decisions on the Basis of Signal Modules

The mechanism of making trade decisions can be represented as the following list of basic principles:

- Each of the modules of signals has its set of market modules (certain combination of prices and values of an indicator) .
- Each market model has a significance that may vary with the range of 1 to 100. The higher is the significance, the stronger the model is.
- Each of the models generates a forecast of direction of the price movement.
- A forecast of a module is the result of search for embedded models, and it is outputted as a number within the range of -100 to 100. The sign determines the direction of forecast movement (negative sign means the price will fall, positive sign means the price will rise) . The absolute value

corresponds to the strength of the best found model.

- The forecast of each module is sent to the final "voting" with a weight coefficient of 0 to 1 specified in its settings ("Weight") .
- The result of voting is a number within the range of -100 to 100, where the sign determines direction of the forecast movement, and the absolute value characterizes the strength of the signal. It is calculated as the arithmetical mean of weighted forecasts of all the modules of signals.

Each generated Expert Advisor has two adjustable settings – threshold levels of opening and closing a position (ThresholdOpen and ThresholdClose) that can be equal to a value in the range of 0 to 100. If the strength of final signal exceeds a threshold level, a trade operation that corresponds to the sign of the signal is performed.

Examples

Consider an Expert Advisor with the following threshold levels: ThresholdOpen=20 and ThresholdClose=90. Two modules of signals participate in making decisions on trade operations: the [MA](#) module with weight 0.4 and the [Stochastic](#) module with weight 0.8. Let's analyze two variants of obtained trade signals:

Variant 1.

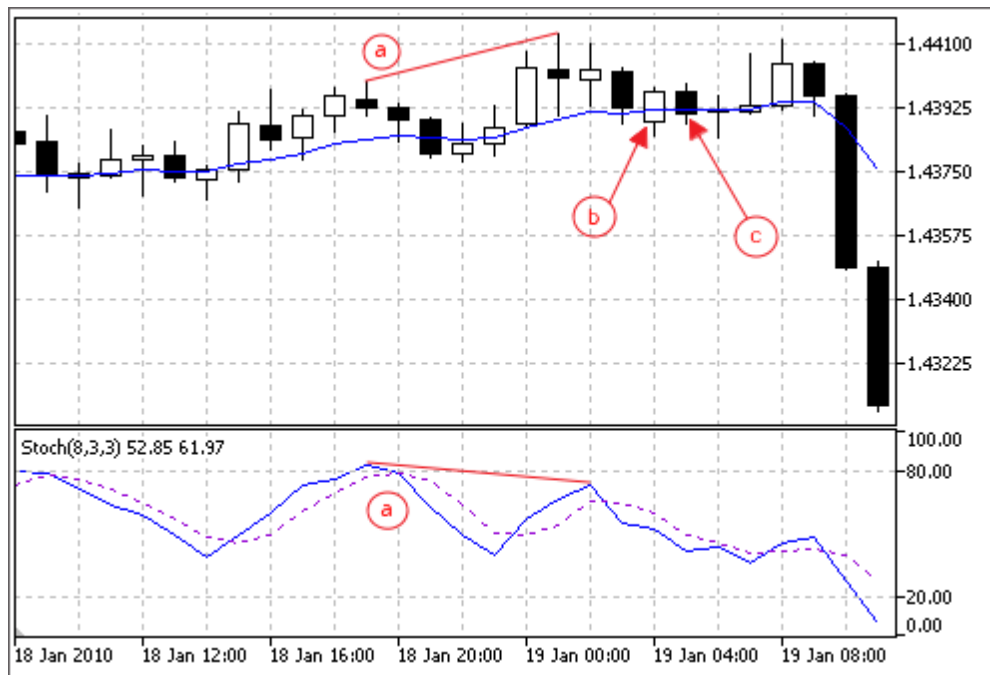
The price crossed the rising MA upwards. This case corresponds to one of the market models implemented in the [MA module](#). This model implies a rise of price. Its significance is equal to 100. At the same time, the Stochastic oscillator turned down and formed a divergence with price. This case corresponds to one of the models implemented in the [Stochastic module](#). This model implies a fall of price. The weight of this model is 80.

Let's calculate the result of final "voting". The rate obtained from the MA module is calculated as $0.4 * 100 = 40$. The value from the Stochastic module is calculated as $0.8 * (-80) = -64$. The final value is calculated as the arithmetical mean of these two rates: $(40 - 64) / 2 = -12$. The result of voting is the signal for selling with relative strength equal to 12. The threshold level that is equal to 20 is not reached. Thus a trade operation is not performed.

Variant 2.

The price crossed the rising MA downwards. This case corresponds to one of the models implemented in the [MA module](#). This model implies a rise of price. Its significance is equal to 10. At the same time, the Stochastic oscillator turned down and formed a divergence with price. This case corresponds to one of the models implemented in the [Stochastic module](#). This model implies a fall of price. The weight of this model is 80.

Let's calculate the result of final "voting". The rate obtained from the MA module is calculated as $0.4 * 10 = 4$. The value from the Stochastic module is calculated as $0.8 * (-80) = -64$. The final value is calculated as the arithmetical mean of these two rates: $(4 - 64) / 2 = -30$. The result of voting is the signal for selling with relative strength equal to 30. The threshold level that is equal to 20 is reached. Thus the result is the signal for opening a short position.



- a) Divergence of the price and the Stochastic oscillator (variants 1 and 2) .
- b) The price crossed the MA indicator upwards (variant 1) .
- c) The price crossed the MA indicator downwards (variants 2) .

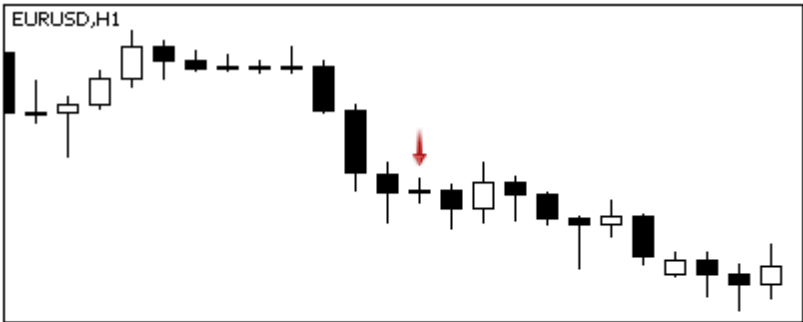
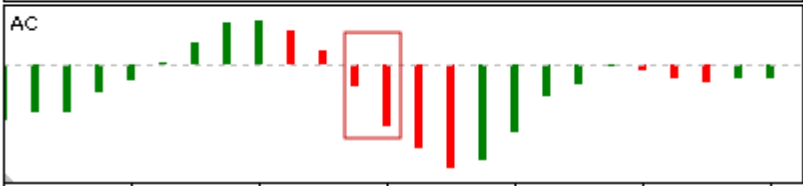
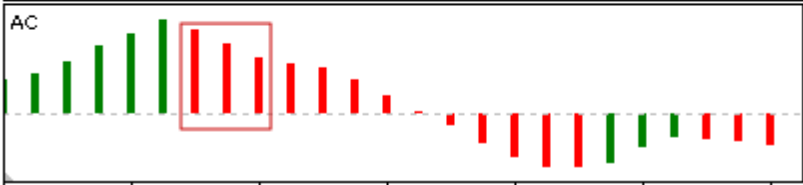
Signals of the Indicator Accelerator Oscillator

This module is based on the market models of the indicator [Accelerator Oscillator](#). The mechanism of making trade decisions based on signals obtained from the modules is described in a [separate section](#).

Conditions of Generation of Signals

Below you can find the description of conditions when the module passes a signal to an Expert Advisor.

Signal Type	Description of Conditions
For buying	<div><ul style="list-style-type: none">The indicator value is above 0 and it rises at the analyzed and at the previous bars.<div><div>EURUSD,H1</div><div>1.44240 1.44005 1.43770 1.43535</div></div><div><div>AC</div><div>0.00</div></div><div>15 Jan 2010 15 Jan 11:00 15 Jan 15:00 15 Jan 19:00 18 Jan 00:00 18 Jan 04:00 18 Jan 08:00</div></div> <div><ul style="list-style-type: none">The indicator value is below 0 and it rises at the analyzed and at the previous bars.<div><div>EURUSD,H1</div><div>1.36910 1.36510 1.36110 1.35710</div></div><div><div>AC</div><div>0.00</div></div><div>17 Feb 2010 17 Feb 19:00 17 Feb 23:00 18 Feb 03:00 18 Feb 07:00 18 Feb 11:00 18 Feb 15:00</div></div>

Signal Type	Description of Conditions
	<div><div><div><div>EURUSD,H1</div></div><div><div>AC</div></div><div>14 Jan 2010 14 Jan 19:00 14 Jan 23:00 15 Jan 03:00 15 Jan 07:00 15 Jan 11:00 15 Jan 15:00</div></div><div><ul style="list-style-type: none">• The indicator value is below 0 and it falls at the analyzed and at the previous bars.</div><div><div><div>EURUSD,H1</div></div><div><div>AC</div></div><div>19 Feb 2010 19 Feb 21:00 22 Feb 02:00 22 Feb 06:00 22 Feb 10:00 22 Feb 14:00 22 Feb 18:00</div></div></div>
No objections to buying	The indicator value grows at the analyzed bar.
No objections to selling	The indicator value falls at the analyzed bar.

Note

Depending on the mode of operation of an Expert Advisor ("Every tick" or "Open prices only") an analyzed bar is either the current bar (with index 0) , or the last formed bar (with index 1) .

Adjustable Parameters

This module has the following adjustable parameters:

Parameter	Description
Weight	Weight of signal of the module in the interval 0 to 1.

Signals of the Indicator Adaptive Moving Average

This module is based on the market models of the indicator [Adaptive Moving Average](#). The mechanism of making trade decisions based on signals obtained from the modules is described in a [separate section](#).

Conditions of Generation of Signals

Below you can find the description of conditions when the module passes a signal to an Expert Advisor.

Signal Type	Description of Conditions
For buying	<ul style="list-style-type: none"> The price has crossed the indicator downwards (the Open price of the analyzed bar is above the indicator and the Close price is below the indicator) and the indicator rises (weak signal) .  <ul style="list-style-type: none"> Moving Average crossover. The price has crossed the indicator upwards (the Open price of the analyzed bar is below the indicator and the Close price is above the indicator) and the indicator rises (strong signal) . 

Signal Type	Description of Conditions
	<div><ul style="list-style-type: none">The lower shadow of the bar has crossed the indicator (the Open and Close prices of the analyzed bar is above the indicator, and the Low price is below the indicator) and the indicator rises (weak signal) .</div> <div></div>
For selling	<div><ul style="list-style-type: none">The price has crossed the indicator upwards (the Open price of the analyzed bar is below the indicator and the Close price is above the indicator) and the indicator falls (weak signal) .</div> <div></div> <div><ul style="list-style-type: none">Moving Average crossover. The price has crossed the indicator downwards (the Open price of the analyzed bar is above the indicator and the Close price is below the indicator) and the indicator falls (strong signal) .</div>

Signal Type	Description of Conditions
	<div><div></div><div><ul style="list-style-type: none">• The upper shadow of the bar has crossed the indicator (the Open and Close prices of the analyzed bar is below the indicator, and the High price is above the indicator) and the indicator falls (weak signal) .</div><div></div></div>
No objections to buying	The prices is above the indicator.
No objections to selling	The price is below the indicator.

Note

Depending on the mode of operation of an Expert Advisor ("Every tick" or "Open prices only") an analyzed bar is either the current bar (with index 0) , or the last formed bar (with index 1) .

Adjustable Parameters

This module has the following adjustable parameters:

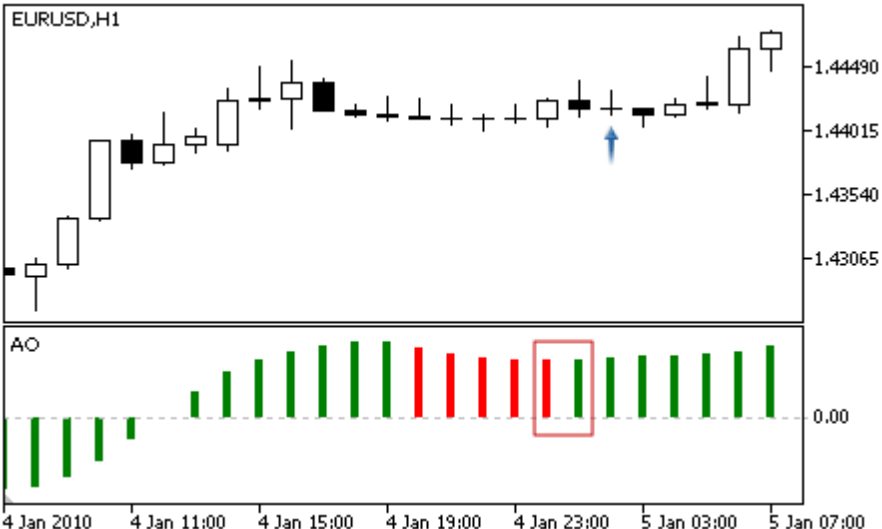
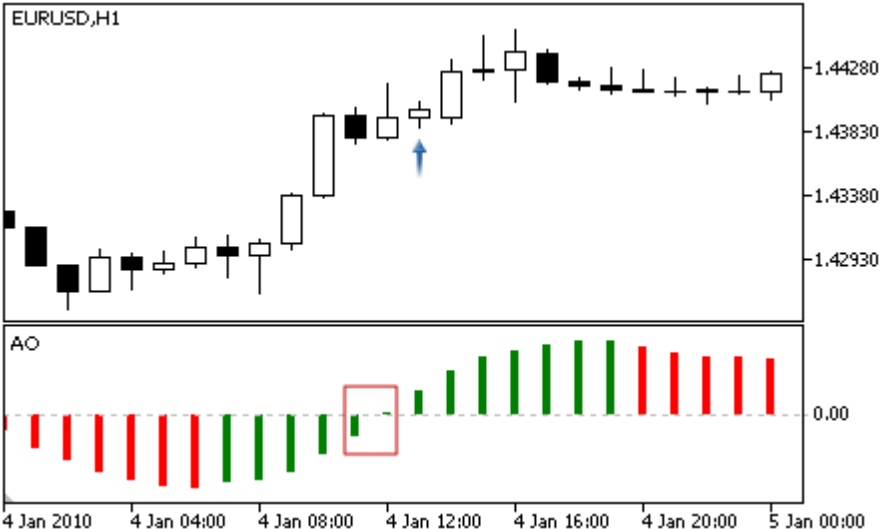
Parameter	Description
Weight	Weight of signal of the module in the interval 0 to 1.
PeriodMA	Period of averaging of the indicator.
Shift	Shift of the indicator along the time axis (in bars) .
Method	Method of averaging .
Applied	A price series used for calculation of the indicator.

Signals of the Indicator Awesome Oscillator

This module of signals is based on the market models of the indicator [Awesome Oscillator](#). The mechanism of making trade decisions based on signals obtained from the modules is described in a [separate section](#).

Conditions of Generation of Signals

Below you can find the description of conditions when the module passes a signal to an Expert Advisor.

Signal Type	Description of Conditions
For buying	<ul style="list-style-type: none"> <p>Saucer – value of the indicator at the analyzed bar rises, and it fell at the previous bars; at that, both values are above 0.</p>  <p>Crossing the zero line – value of the indicator is above 0 at the analyzed bar, and it is below 0 at the previous bar.</p>  <p>Divergence – the first analyzed bottom of the indicator is shallower than the previous one, and the corresponding price valley is deeper than the</p>

Signal Type	Description of Conditions
	<p>previous one. In addition, the indicator must not rise above the zero level.</p>  <p>The top chart is a candlestick plot for EURUSD on an H1 timeframe. It shows an uptrend from point A to point B, followed by a sharp decline. The bottom chart shows the AO indicator for the same period. The indicator bars are green for positive values and red for negative values. A red line connects point 'a' on the AO indicator to point 'b' on the AO indicator, showing a downward trend. The AO indicator value at point 'a' is 0.002791.</p>
For selling	<ul style="list-style-type: none">• Saucer – value of the indicator at the analyzed bar falls, and it rose at the previous bars; at that, both values are below 0.  <p>The top chart is a candlestick plot for EURUSD on an H1 timeframe. It shows an uptrend from point A to point B, followed by a sharp decline. The bottom chart shows the AO indicator for the same period. The indicator bars are green for positive values and red for negative values. A red line connects point 'a' on the AO indicator to point 'b' on the AO indicator, showing a downward trend. The AO indicator value at point 'a' is 0.002791.</p> <ul style="list-style-type: none">• Crossing the zero line – value of the indicator is below 0 at the analyzed bar, and it is above 0 at the previous bar.

Signal Type	Description of Conditions
	<div><div><div><div>EURUSD,H1</div><div>1.44570 1.44130 1.43690 1.43250</div></div><div><div>AO</div><div>0.00</div></div><div>5 Jan 2010 5 Jan 11:00 5 Jan 15:00 5 Jan 19:00 5 Jan 23:00 6 Jan 03:00 6 Jan 07:00</div></div><div><ul style="list-style-type: none">• Divergence – the first analyzed peak of the indicator is lower than the previous one, and the corresponding price peak is higher than the previous one. In addition, the indicator must not falls below the zero level.</div><div><div><div>EURUSD,H1</div><div>1.38170 1.37800 1.37430 1.37060</div></div><div><div>AO</div><div>0.00</div></div><div>9 Feb 2010 9 Feb 20:00 10 Feb 00:00 10 Feb 04:00 10 Feb 08:00 10 Feb 12:00 10 Feb 16:00</div></div></div> <div><div>No objections to buying</div><div>The indicator value grows at the analyzed bar.</div></div> <div><div>No objections to selling</div><div>The indicator value falls at the analyzed bar.</div></div>

Note

Depending on the mode of operation of an Expert Advisor ("Every tick" or "Open prices only") an analyzed bar is either the current bar (with index 0) , or the last formed bar (with index 1) .

Adjustable Parameters

This module has the following adjustable parameters:

Parameter	Description
Weight	Weight of signal of the module in the interval 0 to 1.

Signals of the Oscillator Bears Power

This module of signals is based on the market models of the oscillator [Bears Power](#). The mechanism of making trade decisions based on signals obtained from the modules is described in a [separate section](#).

Conditions of Generation of Signals

Below you can find the description of conditions when the module passes a signal to an Expert Advisor.

Signal Type	Description of Conditions
For buying	<ul style="list-style-type: none"> Reverse - the oscillator turned upwards and its value at the analyzed bar is below 0.  <ul style="list-style-type: none"> Divergence - the first analyzed bottom of the oscillator is higher than the previous one, and the corresponding price bottom is lower than the previous one. In addition, the oscillator must not rise above the zero level. 
For selling	No signals for selling.
No objections to	Value of the oscillator is less than 0.

Signal Type	Description of Conditions
buying	
No objections to selling	No signals.

Note

Depending on the mode of operation of an Expert Advisor ("Every tick" or "Open prices only") an analyzed bar is either the current bar (with index 0) , or the last formed bar (with index 1) .

Adjustable Parameters

This module has the following adjustable parameters:

Parameter	Description
Weight	Weight of signal of the module in the interval 0 to 1.
PeriodBears	Period of calculation of the oscillator.

Bulls Power

This module of signals is based on the market models of the oscillator [Bulls Power](#). The mechanism of making trade decisions based on signals obtained from the modules is described in a [separate section](#).

Conditions of Generation of Signals

Below you can find the description of conditions when the module passes a signal to an Expert Advisor.

Signal Type	Description of Conditions
For buying	No signals for buying.
For selling	<div><ul style="list-style-type: none">Reverse - the oscillator turned downwards and its value at the analyzed bar is above 0.<div></div><ul style="list-style-type: none">Divergence - the first analyzed peak of the oscillator is lower than the previous one, and the corresponding price peak is higher than the previous peak. In addition, the oscillator must not fall below the zero level.<div></div></div>
No objections to	No signals.

Signal Type	Description of Conditions
buying	
No objections to selling	Value of the oscillator is greater than 0.

Note

Depending on the mode of operation of an Expert Advisor ("Every tick" or "Open prices only") an analyzed bar is either the current bar (with index 0) , or the last formed bar (with index 1) .

Adjustable Parameters

This module has the following adjustable parameters:

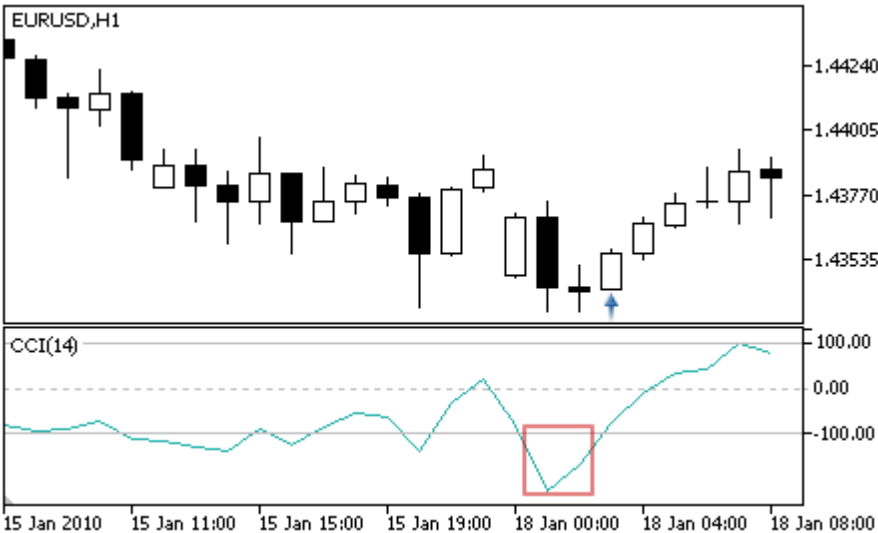
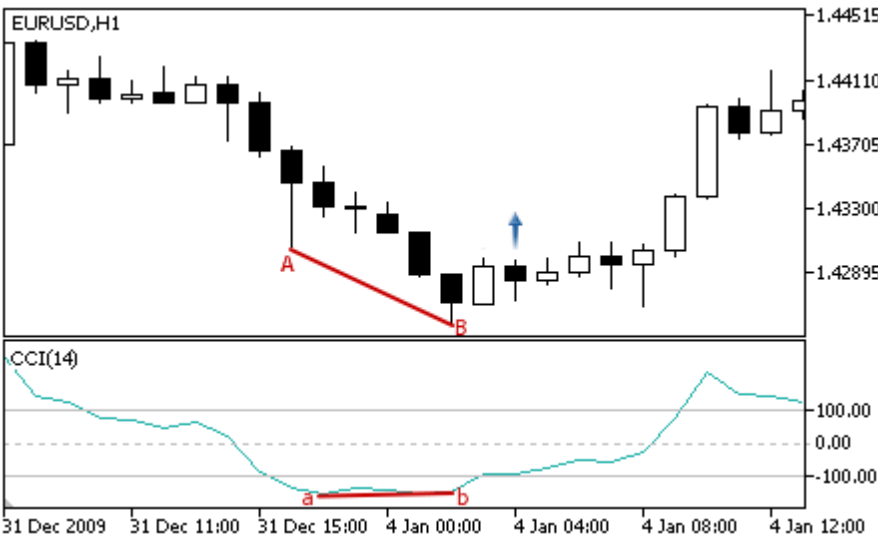
Parameter	Description
Weight	Weight of signal of the module in the interval 0 to 1.
PeriodBulls	Period of calculation of the oscillator.

Signals of the Oscillator Commodity Channel Index

This module of signals is based on the market models of the oscillator [Commodity Channel Index](#). The mechanism of making trade decisions based on signals obtained from the modules is described in a [separate section](#).

Conditions of Generation of Signals

Below you can find the description of conditions when the module passes a signal to an Expert Advisor.

Signal Type	Description of Conditions
For buying	<ul style="list-style-type: none"> Reverse behind the level of overselling – the oscillator turned upwards and its value at the analyzed bar is behind the level of overselling (default value is -100) .  <ul style="list-style-type: none"> Divergence – the first analyzed bottom of the oscillator is higher than the previous one, and the corresponding price bottom is lower than the previous one. 

Signal Type	Description of Conditions
	<div><ul style="list-style-type: none">Double divergence – the oscillator form three consequent bottoms, each of them is higher than the previous one; and the price formed three corresponding bottoms, and each of them is lower than the previous one.</div> <div><p>The chart displays EURUSD on an H1 timeframe. The price chart (top) shows three consecutive price bottoms labeled A, B, and C, connected by a red line. The CCI(14) oscillator (bottom) shows three consecutive bottoms labeled a, b, and c, also connected by a red line. The price bottoms are higher than the previous ones, while the oscillator bottoms are lower than the previous ones, indicating a double divergence.</p></div>
For selling	<div><ul style="list-style-type: none">Reverse behind the overbought level – the oscillator turned downwards and its value at the analyzed bar is behind the overbought level (default value is 100) .</div> <div><p>The chart displays EURUSD on an H1 timeframe. The price chart (top) shows a peak followed by a decline. The CCI(14) oscillator (bottom) shows a peak followed by a decline. A red box highlights the peak of the oscillator, and a red arrow points to the corresponding price peak. The oscillator value at the analyzed bar is behind the overbought level (100).</p></div> <div><ul style="list-style-type: none">Divergence – the first analyzed peak of the oscillator is lower than the previous one, and the corresponding price peak is higher than the previous peak.</div>

Signal Type	Description of Conditions
	<div><div><div><div>EURUSD,H1</div><div>1.37110 1.36800 1.36490 1.36180 1.35870</div></div><div><div>CCI(14)</div><div>100.00 0.00 -100.00</div></div><div>5 Mar 2010 5 Mar 20:00 8 Mar 01:00 8 Mar 05:00 8 Mar 09:00 8 Mar 13:00 8 Mar 17:00</div></div><div><ul style="list-style-type: none">• Double divergence – the oscillator formed three consequent peaks, each of them is lower than the previous one; and the price formed three corresponding peaks, each of them is higher than the previous one.</div><div><div><div>EURUSD,H1</div><div>1.23045 1.22750 1.22455 1.22160 1.21865</div></div><div><div>CCI(14)</div><div>100.00 0.00 -100.00</div></div><div>14 Jun 2010 14 Jun 09:00 14 Jun 13:00 14 Jun 17:00 14 Jun 21:00 15 Jun 01:00 15 Jun 05:00</div></div></div>

Note

Depending on the mode of operation of an Expert Advisor ("Every tick" or "Open prices only") an analyzed bar is either the current bar (with index 0) , or the last formed bar (with index 1) .

Adjustable Parameters

This module has the following adjustable parameters:

Parameter	Description
Weight	Weight of signal of the module in the interval 0 to 1.
PeriodCCI	Period of calculation of the oscillator.
Applied	A price series used for calculation of the oscillator.

DeMarker

This module of signals is based on the market models of the oscillator [DeMarker](#). The mechanism of making trade decisions based on signals obtained from the modules is described in a [separate section](#).

Conditions of Generation of Signals

Below you can find the description of conditions when the module passes a signal to an Expert Advisor.

Signal Type	Description of Conditions
For buying	<ul style="list-style-type: none"> Reverse behind the oversold level - the oscillator turned upwards and its value at the analyzed bar is behind the oversold level (default value is 0.3) .  <ul style="list-style-type: none"> Divergence - the first analyzed bottom of the oscillator is higher than the previous one, and the corresponding price bottom is lower than the previous one.  <ul style="list-style-type: none"> Double divergence - the oscillator form three consequent bottoms,

Signal Type	Description of Conditions
	<p>each of them is higher than the previous one; and the price formed three corresponding bottoms, and each of them is lower than the previous one.</p> <div></div>
For selling	<ul style="list-style-type: none">Reverse behind the overbought level - the oscillator turned downwards and its value at the analyzed bar is behind the overbought level (default value is 0.7) . <div></div> <ul style="list-style-type: none">Divergence - the first analyzed peak of the oscillator is lower than the previous one, and the corresponding price peak is higher than the previous peak.

Signal Type	Description of Conditions
	<div><div></div><div></div><div><ul style="list-style-type: none">• Double divergence – the oscillator formed three consequent peaks, each of them is lower than the previous one; and the price formed three corresponding peaks, each of them is higher than the previous one.</div><div></div><div></div></div>
No objections to buying	Value of the oscillator grows at the analyzed bar.
No objections to selling	Value of the oscillator falls at the analyzed bar.

Note

Depending on the mode of operation of an Expert Advisor ("Every tick" or "Open prices only") an analyzed bar is either the current bar (with index 0) , or the last formed bar (with index 1) .

Adjustable Parameters

This module has the following adjustable parameters:

Parameter	Description
Weight	Weight of signal of the module in the interval 0 to 1.
PeriodDeM	Period of calculation of the oscillator.

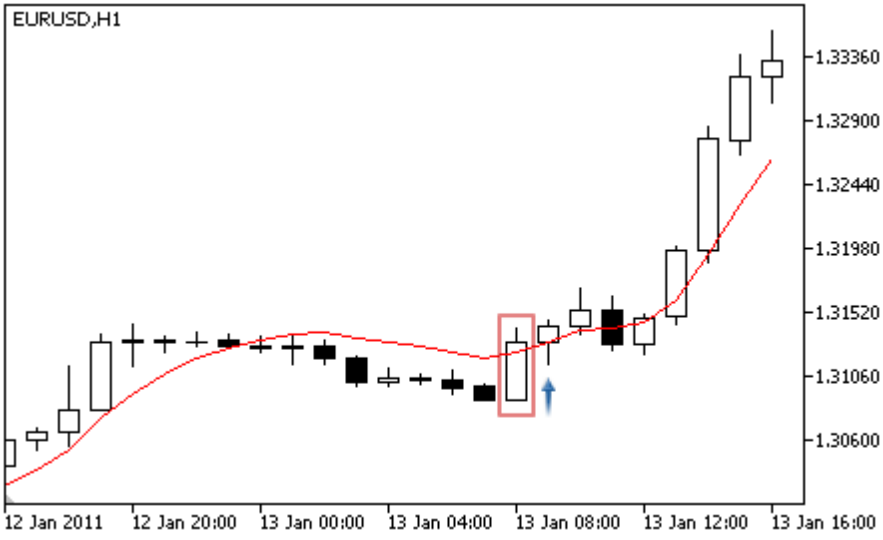
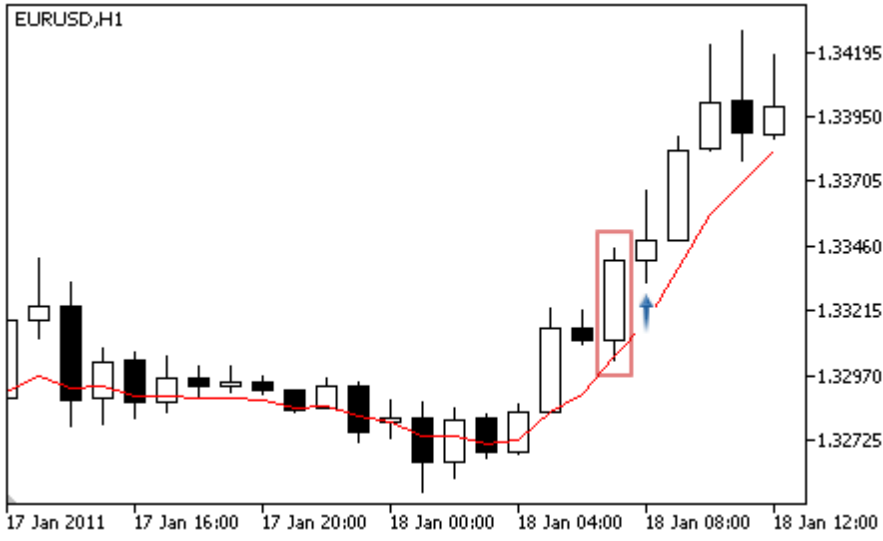
Signals of the Indicator Double Exponential Moving Average

This module is based on the market models of the indicator [Double Exponential Moving Average](#). The mechanism of making trade decisions based on signals obtained from the modules is described in a [separate section](#).

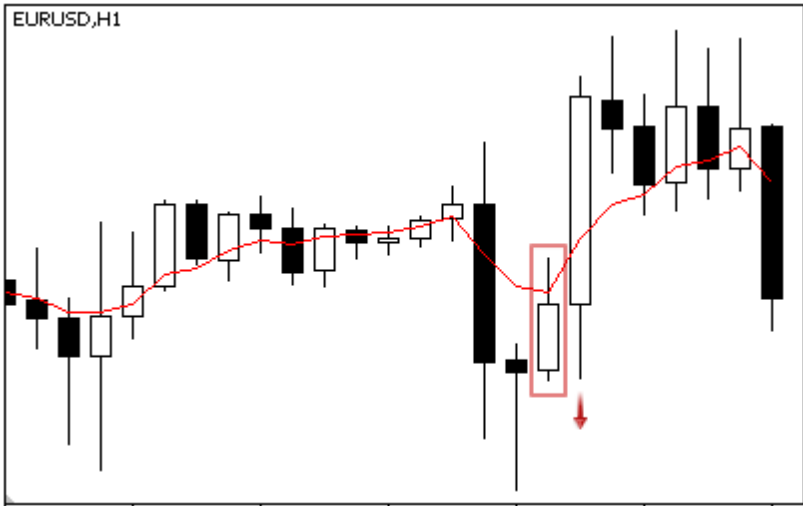
Conditions of Generation of Signals

Below you can find the description of conditions when the module passes a signal to an Expert Advisor.

Signal Type	Description of Conditions
For buying	<div><ul style="list-style-type: none">The price has crossed the indicator downwards (the Open price of the analyzed bar is above the indicator and the Close price is below the indicator) and the indicator rises (weak signal) .<div><div>EURUSD,H1</div><p>The chart displays price movement from 18 Jan 2011 to 19 Jan 2011. A red line represents the Double Exponential Moving Average. A red box highlights a candle where the price crossed the indicator downwards. A blue arrow points to the indicator line below the box.</p></div><ul style="list-style-type: none">Moving Average crossover. The price has crossed the indicator upwards (the Open price of the analyzed bar is below the indicator and the Close price is above the indicator) and the indicator rises (strong signal) .</div>

Signal Type	Description of Conditions
	<div><div><div>EURUSD,H1</div></div><div><ul style="list-style-type: none">• The lower shadow of the bar has crossed the indicator (the Open and Close prices of the analyzed bar is above the indicator, and the Low price is below the indicator) and the indicator rises (weak signal) .</div><div><div><div>EURUSD,H1</div></div><div><ul style="list-style-type: none">• The price has crossed the indicator upwards (the Open price of the analyzed bar is below the indicator and the Close price is above the indicator) and the indicator falls (weak signal) .</div></div></div>
For selling	<ul style="list-style-type: none">• The price has crossed the indicator upwards (the Open price of the analyzed bar is below the indicator and the Close price is above the indicator) and the indicator falls (weak signal) .

Signal Type	Description of Conditions
	<div><div><div>EURUSD,H1</div></div><div><ul style="list-style-type: none">• Moving Average crossover. The price has crossed the indicator downwards (the Open price of the analyzed bar is above the indicator and the Close price is below the indicator) and the indicator falls (strong signal) .</div></div> <div><div><div>EURUSD,H1</div></div><div><ul style="list-style-type: none">• The upper shadow of the bar has crossed the indicator (the Open and Close prices of the analyzed bar is below the indicator, and the High price is above the indicator) and the indicator falls (weak signal) .</div></div>

Signal Type	Description of Conditions
	<div><div>EURUSD,H1</div><div>26 Jan 2011 26 Jan 21:00 27 Jan 01:00 27 Jan 05:00 27 Jan 09:00 27 Jan 13:00 27 Jan 17:00</div><div>1.37520 1.37350 1.37180 1.37010 1.36840 1.36670 1.36500</div></div>
No objections to buying	The price is above the indicator.
No objections to selling	The price is below the indicator.

Note

Depending on the mode of operation of an Expert Advisor ("Every tick" or "Open prices only") an analyzed bar is either the current bar (with index 0) , or the last formed bar (with index 1) .

Adjustable Parameters

This module has the following adjustable parameters:

Parameter	Description
Weight	Weight of signal of the module in the interval 0 to 1.
PeriodMA	Period of averaging of the indicator.
Shift	Shift of the indicator along the time axis (in bars) .
Method	Method of averaging.
Applied	A price series used for calculation of the indicator.

Signals of the Indicator Envelopes

This module of signals is based on the market models of the indicator [Envelopes](#). The mechanism of making trade decisions based on signals obtained from the modules is described in a [separate section](#).

Conditions of Generation of Signals

Below you can find the description of conditions when the module passes a signal to an Expert Advisor.

Signal Type	Description of Conditions
For buying	<ul style="list-style-type: none"> The price is near the lower line of the indicator at the analyzed bar.  <ul style="list-style-type: none"> The price crossed the upper line of the indicator at the analyzed bar. 
For selling	<ul style="list-style-type: none"> The price is near the upper line of the indicator at the analyzed bar.

Signal Type	Description of Conditions
	<div><div><div>EURUSD,H1</div></div><div><ul style="list-style-type: none">• The price crossed the lower line of the indicator at the analyzed bar.</div><div><div>EURUSD,H1</div></div></div>
No objections to buying	No signals.
No objections to selling	No signals.

Note

Depending on the mode of operation of an Expert Advisor ("Every tick" or "Open prices only") an analyzed bar is either the current bar (with index 0) , or the last formed bar (with index 1) .

Adjustable Parameters

This module has the following adjustable parameters:

Parameter	Description
Weight	Weight of signal of the module in the interval 0 to 1.
PeriodMA	Period of calculation of the indicator.
Shift	Shift of the indicator along the time axis (in bars) .
Method	Method of averaging .
Applied	A price series used for calculation of the indicator.
Deviation	Deviation of the envelope borders from the center line (MA) in percentage terms.

Signals of the Indicator Fractal Adaptive Moving Average

This module of signals is based on the market models of the indicator [Fractal Adaptive Moving Average](#) . The mechanism of making trade decisions based on signals obtained from the modules is described in a [separate section](#).

Conditions of Generation of Signals

Below you can find the description of conditions when the module passes a signal to an Expert Advisor.

Signal Type	Description of Conditions
For buying	<div><ul style="list-style-type: none">The price has crossed the indicator downwards (the Open price of the analyzed bar is above the indicator and the Close price is below the indicator) and the indicator rises (weak signal) . </div> <div><ul style="list-style-type: none">Moving Average crossover. The price has crossed the indicator upwards (the Open price of the analyzed bar is below the indicator and the Close price is above the indicator) and the indicator rises (strong signal) . </div>

Signal Type	Description of Conditions
	<div><ul style="list-style-type: none">The lower shadow of the bar has crossed the indicator (the Open and Close prices of the analyzed bar is above the indicator, and the Low price is below the indicator) and the indicator rises (weak signal) .</div> <div></div>
For selling	<div><ul style="list-style-type: none">The price has crossed the indicator upwards (the Open price of the analyzed bar is below the indicator and the Close price is above the indicator) and the indicator falls (weak signal) .</div> <div></div> <div><ul style="list-style-type: none">Moving Average crossover. The price has crossed the indicator downwards (the Open price of the analyzed bar is above the indicator and the Close price is below the indicator) and the indicator falls (strong signal) .</div>

Signal Type	Description of Conditions
	<div><div><div>EURUSD,H1</div></div><div><ul style="list-style-type: none">• The upper shadow of the bar has crossed the indicator (the Open and Close prices of the analyzed bar is below the indicator, and the High price is above the indicator) and the indicator falls (weak signal) .</div><div><div>EURUSD,H1</div></div></div>
No objections to buying	The prices is above the indicator.
No objections to selling	The price is below the indicator.

Note

Depending on the mode of operation of an Expert Advisor ("Every tick" or "Open prices only") an analyzed bar is either the current bar (with index 0) , or the last formed bar (with index 1) .

Adjustable Parameters

This module has the following adjustable parameters:

Parameter	Description
Weight	Weight of signal of the module in the interval 0 to 1.
PeriodMA	Period of averaging of the indicator.
Shift	Shift of the indicator along the time axis (in bars) .
Method	Method of averaging .
Applied	A price series used for calculation of the indicator.

Signals of the Intraday Time Filter

This module is based on the assumption that the efficiency of market models changes in time. Using this module, you can filter signals received from the other modules by hour and days of week. It allows increasing the quality of generated signals due to cutting off the unfavorable time periods. The mechanism of making trade decisions on the basis of signals of the modules is described in a [separate section](#).

Conditions of Generation of Signals

Below you can find the description of conditions when the module passes a signal to an Expert Advisor.

Signal Type	Description of Conditions
For buying	No signals.
For selling	No signals.
No objections to buying	The current date and time meet the specified parameters.
No objections to selling	The current date and time meet the specified parameters.

Adjustable Parameters

This module has the following adjustable parameters:

Parameter	Description
Weight	Weight of signal of the module in the interval 0 to 1.
GoodHourOfDay	Number of the only hour of day (from 0 to 23) when trade signals will be enabled. If the value is -1, the signals will be enabled through the whole day.
BadHoursOfDay	The bit field. Each bit of this field corresponds to an hour of day (0 bit - 0 hour, ..., 23 bit - 23-rd hour) . If the value of a bit is equal to 0, trade signals will be enabled during the corresponding hour. If the value of a bit is equal to 1, trade signals will be disabled during the corresponding hour. A specified number is represented as a binary number and is used as bit mask. Disabled hours have higher priority than the enabled ones.
GoodDayOfWeek	Number of the only day of week (from 0 to 6, where 0 is Sunday) , when trade signals will be enabled. If the value is -1, the signals will be enabled on any day.
BadDaysOfWeek	The bit field. Each bit of this field corresponds to a day of week (0 bit - Sunday, ..., 6 bit - Saturday) . If the value of a bit is equal to 0, trade signals will be enabled during the corresponding day. If the value of a bit is equal to 1, trade signals will be disabled during the corresponding day. A specified number is represented as a binary number and is used as bit mask.

Parameter	Description
	Disabled days have higher priority than the enabled ones.

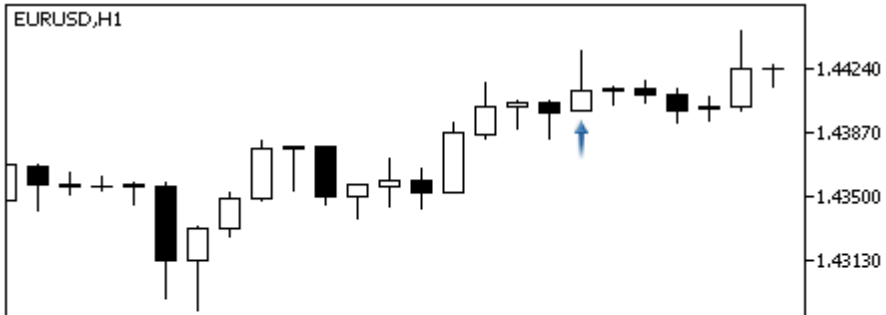
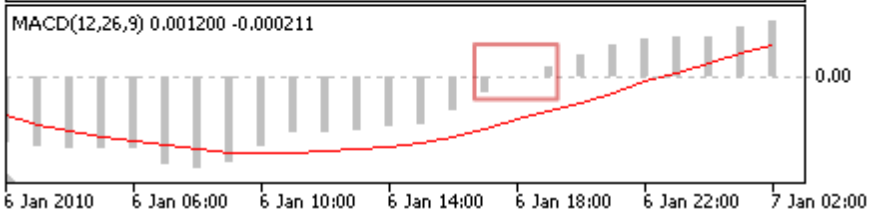
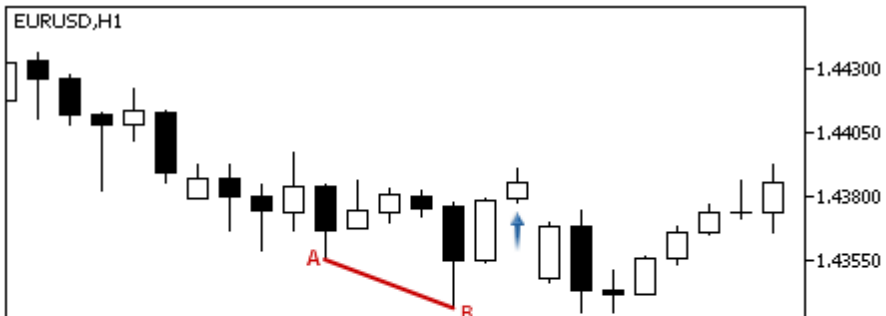
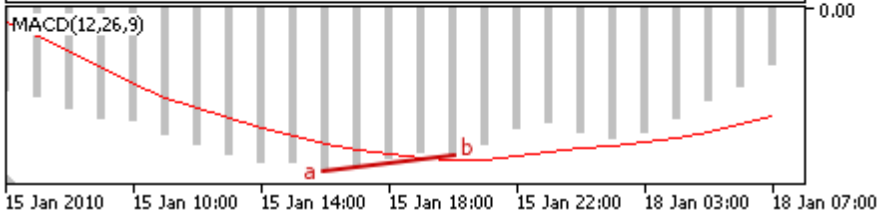
Signals of the Oscillator MACD

This module of signals is based on the market models of the oscillator [MACD](#). The mechanism of making trade decisions based on signals obtained from the modules is described in a [separate section](#).

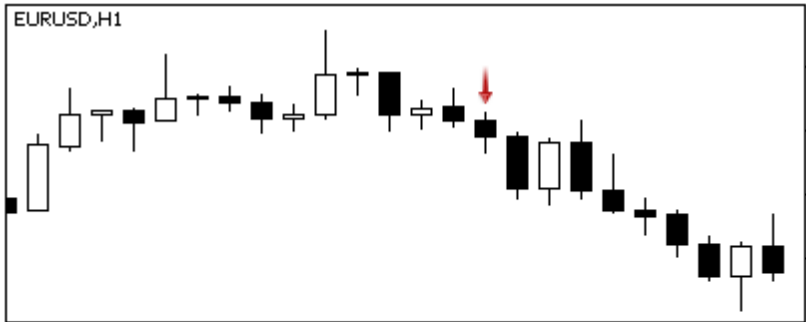
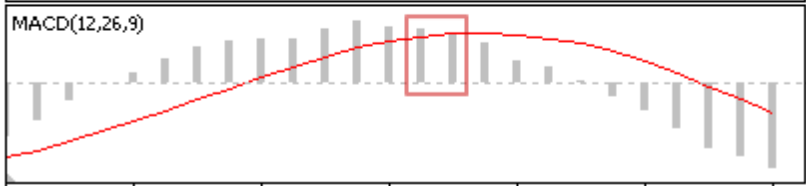
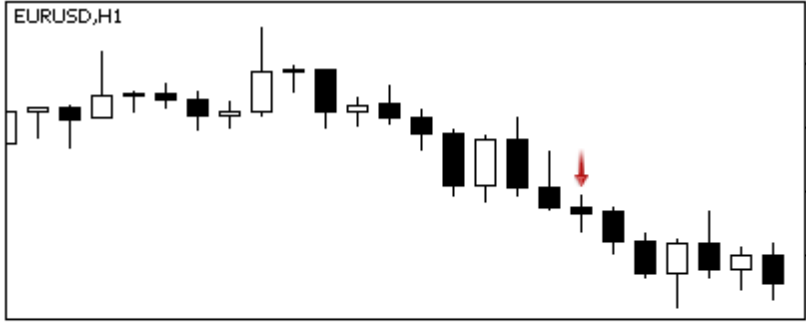
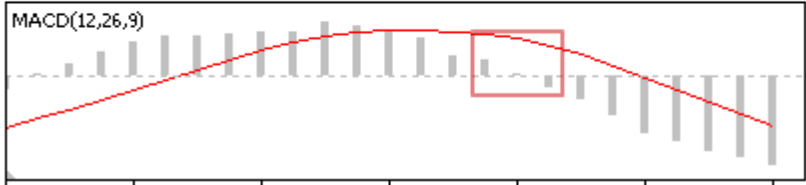
Conditions of Generation of Signals

Below you can find the description of conditions when the module passes a signal to an Expert Advisor.

Signal Type	Description of Conditions
For buying	<ul style="list-style-type: none"> Reverse - the oscillator turned upwards (the oscillator rises at the analyzed bar and falls at the previous one) .  <ul style="list-style-type: none"> Crossover of the main and signal line - the main line is above the signal line at the analyzed bar and below the signal line at the previous one.  <ul style="list-style-type: none"> Crossing the zero level - the main line is above the zero level at the analyzed bar and below the zero level at the previous one.

Signal Type	Description of Conditions
	<div><div><div>EURUSD,H1</div><div>MACD(12,26,9) 0.001200 -0.000211</div><div>6 Jan 2010 6 Jan 06:00 6 Jan 10:00 6 Jan 14:00 6 Jan 18:00 6 Jan 22:00 7 Jan 02:00</div></div><div><ul style="list-style-type: none">• Divergence – the first analyzed bottom of the oscillator is higher than the previous one, and the corresponding price bottom is lower than the previous one.</div><div><div><div>EURUSD,H1</div><div>MACD(12,26,9)</div><div>15 Jan 2010 15 Jan 10:00 15 Jan 14:00 15 Jan 18:00 15 Jan 22:00 18 Jan 03:00 18 Jan 07:00</div></div><div><ul style="list-style-type: none">• Double divergence – the oscillator form three consequent bottoms, each of them is higher than the previous one; and the price formed three corresponding bottoms, and each of them is lower than the previous one.</div></div></div>

Signal Type	Description of Conditions
	<div></div>
For selling	<div><ul style="list-style-type: none">Reverse - the oscillator turned downwards (the oscillator falls at the analyzed bar and rises at the previous one) .<div></div><ul style="list-style-type: none">Crossover of the main and signal line - the main line is below the signal line at the analyzed bar and above the signal line at the previous one.</div>

Signal Type	Description of Conditions
	<div><div><div><div>EURUSD,H1</div><div>1.44270 1.43935 1.43600 1.43265</div></div><div><div>MACD(12,26,9)</div><div>0.00</div></div><div>6 Jan 2010 6 Jan 19:00 6 Jan 23:00 7 Jan 03:00 7 Jan 07:00 7 Jan 11:00 7 Jan 15:00</div></div><div><ul style="list-style-type: none">• Crossing the zero level – the main line is below the zero level at the analyzed bar and above the zero level at the previous one.</div><div><div><div>EURUSD,H1</div><div>1.44270 1.43935 1.43600 1.43265</div></div><div><div>MACD(12,26,9)</div><div>0.00</div></div><div>6 Jan 2010 6 Jan 21:00 7 Jan 01:00 7 Jan 05:00 7 Jan 09:00 7 Jan 13:00 7 Jan 17:00</div></div><div><ul style="list-style-type: none">• Divergence – the first analyzed peak of the oscillator is lower than the previous one, and the corresponding price peak is higher than the previous peak.</div></div>

Signal Type	Description of Conditions
	<div><div><div><div>EURUSD,H1</div><div>1.41680 1.41465 1.41250 1.41035</div></div><div><div>MACD(12,26,9)</div><div>0.00</div></div><div>22 Jan 10:0022 Jan 05:0022 Jan 09:0022 Jan 13:0022 Jan 17:0022 Jan 21:0025 Jan 02:00</div></div><div><ul style="list-style-type: none">• Double divergence – the oscillator formed three consequent peaks, each of them is lower than the previous one; and the price formed three corresponding peaks, each of them is higher than the previous one.</div><div><div><div>EURUSD,H1</div><div>1.45440 1.45225 1.45010 1.44795</div></div><div><div>MACD(12,26,9)</div><div>0.00</div></div><div>11 Jan 05:0011 Jan 09:0011 Jan 13:0011 Jan 17:0011 Jan 21:0012 Jan 01:00</div></div></div>
No objections to buying	Value of the oscillator grows at the analyzed bar.
No objections to selling	Value of the oscillator falls at the analyzed bar.

Note

Depending on the mode of operation of an Expert Advisor ("Every tick" or "Open prices only") an analyzed bar is either the current bar (with index 0) , or the last formed bar (with index 1) .

Adjustable Parameters

This module has the following adjustable parameters:

Parameter	Description
Weight	Weight of signal of the module in the interval 0 to 1.
PeriodFast	Period of calculation of the fast EMA.
PeriodSlow	Period of calculation of the slow EMA.
PeriodSignal	Period of smoothing.
Applied	A price series used for calculation of the oscillator.

Signals of the Indicator Moving Average

This module of signals is based on the market models of the indicator [Moving Average](#). The mechanism of making trade decisions based on signals obtained from the modules is described in a [separate section](#).

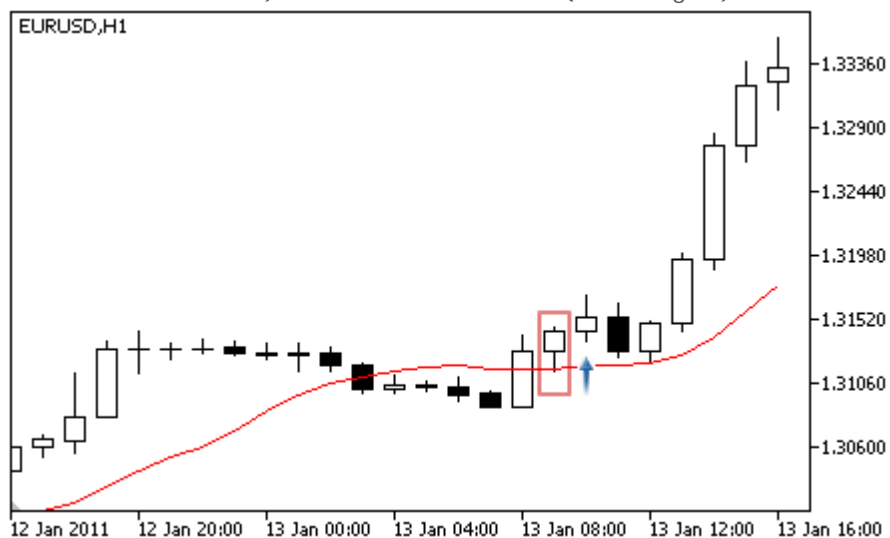
Conditions of Generation of Signals

Below you can find the description of conditions when the module passes a signal to an Expert Advisor.

Signal Type	Description of Conditions
For buying	<div><ul style="list-style-type: none">The price has crossed the indicator downwards (the Open price of the analyzed bar is above the indicator and the Close price is below the indicator) and the indicator rises (weak signal) .<div></div><ul style="list-style-type: none">Moving Average crossover. The price has crossed the indicator upwards (the Open price of the analyzed bar is below the indicator and the Close price is above the indicator) and the indicator rises (strong signal) .<div></div></div>

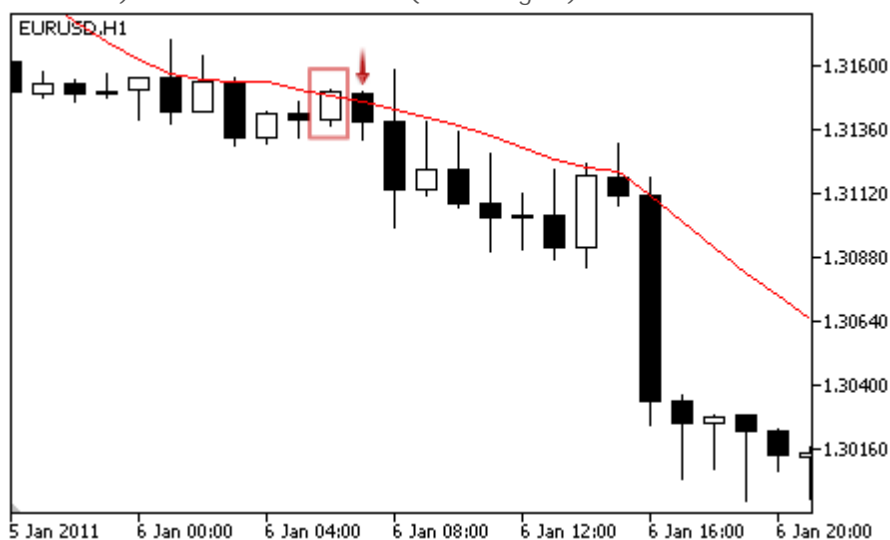
Description of Conditions

- The lower shadow of the bar has crossed the indicator (the Open and Close prices of the analyzed bar is above the indicator, and the Low price is below the indicator) and the indicator rises (weak signal) .



For selling

- The price has crossed the indicator upwards (the Open price of the analyzed bar is below the indicator and the Close price is above the indicator) and the indicator falls (weak signal) .



- **Moving Average crossover.** The price has crossed the indicator downwards (the Open price of the analyzed bar is above the indicator and the Close price is below the indicator) and the indicator falls (strong signal) .

Signal Type	Description of Conditions
	<div><div><div>EURUSD,H1</div><div>1.31475 1.31210 1.30945 1.30680 1.30415 1.30150 1.29885</div><div>6 Jan 2011 6 Jan 09:00 6 Jan 13:00 6 Jan 17:00 6 Jan 21:00 7 Jan 01:00 7 Jan 05:00</div></div><div><ul style="list-style-type: none">• The upper shadow of the bar has crossed the indicator (the Open and Close prices of the analyzed bar is below the indicator, and the High price is above the indicator) and the indicator falls (weak signal) .</div><div><div><div>EURUSD,H1</div><div>1.35605 1.35440 1.35275 1.35110 1.34945 1.34780 1.34615</div><div>11 Feb 2011 11 Feb 15:00 11 Feb 19:00 11 Feb 23:00 14 Feb 03:00 14 Feb 07:00 14 Feb 11:00</div></div></div></div>
No objections to buying	The prices is above the indicator.
No objections to selling	The price is below the indicator.

Note

Depending on the mode of operation of an Expert Advisor ("Every tick" or "Open prices only") an analyzed bar is either the current bar (with index 0) , or the last formed bar (with index 1) .

Adjustable Parameters

This module has the following adjustable parameters:

Parameter	Description
Weight	Weight of signal of the module in the interval 0 to 1.
PeriodMA	Period of averaging of the indicator.
Shift	Shift of the indicator along the time axis (in bars) .
Method	Method of averaging .
Applied	A price series used for calculation of the indicator.

Signals of the indicator Parabolic SAR

This module of signals is based on the market models of the indicator [Parabolic SAR](#). The mechanism of making trade decisions based on signals obtained from the modules is described in a [separate section](#).

Conditions of Generation of Signals

Below you can find the description of conditions when the module passes a signal to an Expert Advisor.

Signal Type	Description of Conditions
For buying	<div>Reverse - the indicator is below the price at the analyzed bar and above the price at the previous one.</div> <div></div>
For selling	<div>Reverse - the indicator is above the price at the analyzed bar and below the price at the previous one.</div> <div></div>
No objections to buying	The prices is above the indicator.

Signal Type	Description of Conditions
No objections to selling	The price is below the indicator.

Note

Depending on the mode of operation of an Expert Advisor ("Every tick" or "Open prices only") an analyzed bar is either the current bar (with index 0) , or the last formed bar (with index 1) .

Adjustable Parameters

This module has the following adjustable parameters:

Parameter	Description
Weight	Weight of signal of the module in the interval 0 to 1.
Step	The increment of speed of the indicator.
Maximum	Maximum rate of the speed of convergence of the indicator with the price.

Signals of the Oscillator Relative Strength Index

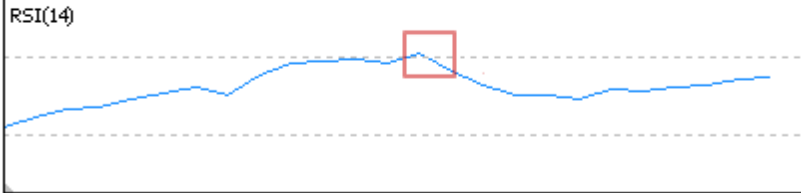
This module of signals is based on the market models of the oscillator [Relative Strength Index](#). The mechanism of making trade decisions based on signals obtained from the modules is described in a [separate section](#).

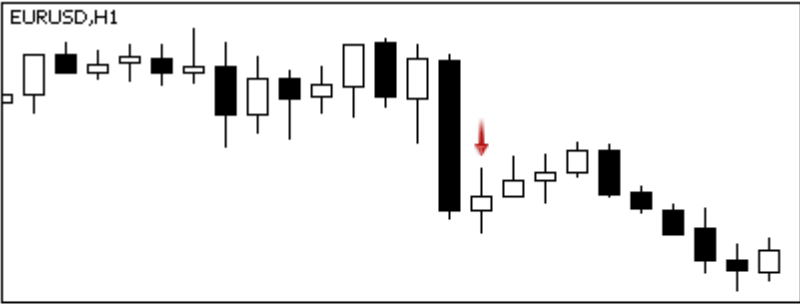
Conditions of Generation of Signals

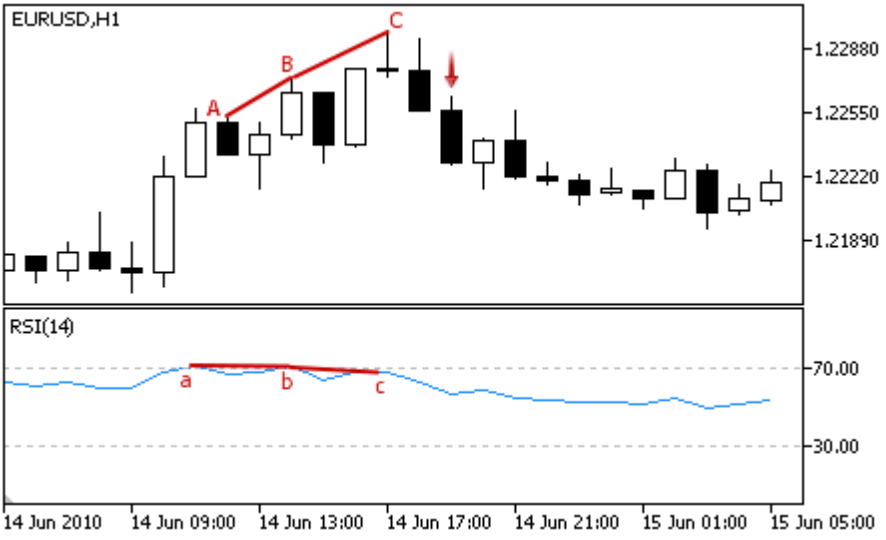
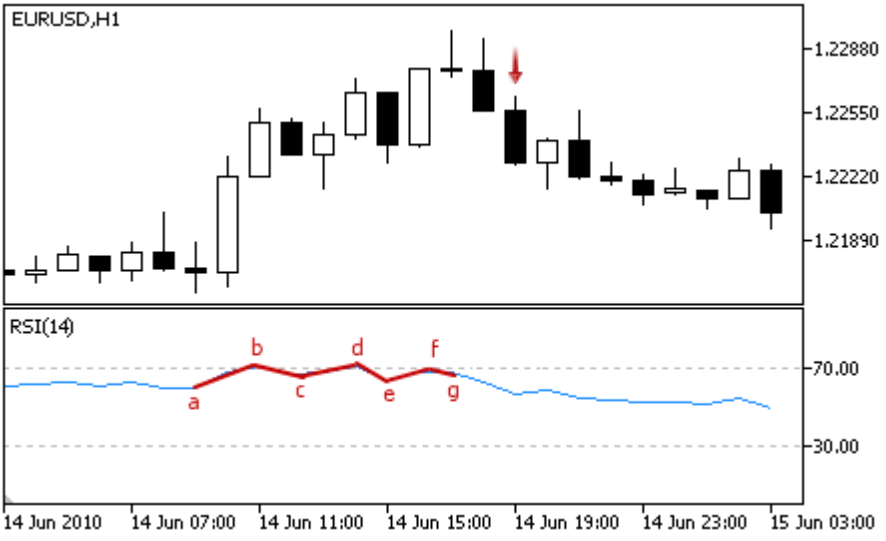
Below you can find the description of conditions when the module passes a signal to an Expert Advisor.

Signal Type	Description of Conditions
For buying	<ul style="list-style-type: none"> Reverse behind the oversold level – the oscillator turned upwards and its value at the analyzed bar is behind the oversold level (default value is 30) .  <ul style="list-style-type: none"> Failed swing – the oscillator rises higher than the previous peak at the analyzed bar.  <ul style="list-style-type: none"> Divergence – the first analyzed bottom of the oscillator is lower than the

Signal Type	Description of Conditions
	<p>previous one, and the corresponding price bottom is lower than the previous one.</p> <div><p>20 Aug 2010 20 Aug 10:00 20 Aug 14:00 20 Aug 18:00 20 Aug 22:00 23 Aug 02:00 23 Aug 06:00</p></div> <ul style="list-style-type: none">• Double divergence – the oscillator form three consequent bottoms, each of them is higher than the previous one; and the price formed three corresponding bottoms, and each of them is lower than the previous one. <div><p>11 Nov 2010 11 Nov 19:00 11 Nov 23:00 12 Nov 03:00 12 Nov 07:00 12 Nov 11:00 12 Nov 15:00</p></div> <ul style="list-style-type: none">• Head/Shoulders – the oscillator formed three consequent bottoms, and the mid one is lower than the others.

Signal Type	Description of Conditions
	<div><div>EURUSD,H1</div><div>RSI(14) 19.78</div><div>16 Feb 10:00 16 Feb 16:00 16 Feb 22:00 17 Feb 02:00</div></div>
For selling	<div><ul style="list-style-type: none">Reverse behind the overbought level - the oscillator turned downwards and its value at the analyzed bar is behind the overbought level (default value is 70) .<div><div>EURUSD,H1</div><div>RSI(14)</div><div>28 Feb 12:00 28 Feb 18:00 28 Feb 24:00 1 Mar 00:00</div></div><ul style="list-style-type: none">Failed swing - the oscillator falls lower than the previous bottom at the analyzed bar.</div>

Signal Type	Description of Conditions
	<div><div><div><div>EURUSD,H1</div><div>1.27220 1.26960 1.26700 1.26440</div></div><div><div>RSI(14)</div><div>70.00 30.00</div></div><div>23 Aug 2010 23 Aug 06:00 23 Aug 10:00 23 Aug 14:00 23 Aug 18:00 23 Aug 22:00 24 Aug 02:00</div></div><div><ul style="list-style-type: none">Divergence – the first analyzed peak of the oscillator is lower than the previous one, and the corresponding price peak is higher than the previous peak.</div><div><div><div><div>EURUSD,H1</div><div>1.34240 1.33740 1.33240 1.32740</div></div><div><div>RSI(14)</div><div>70.00 30.00</div></div><div>21 Sep 2010 22 Sep 02:00 22 Sep 06:00 22 Sep 10:00 22 Sep 14:00 22 Sep 18:00 22 Sep 22:00</div></div><div><ul style="list-style-type: none">Double divergence – the oscillator formed three consequent peaks, each of them is lower than the previous one; and the price formed three corresponding peaks, each of them is higher than the previous one.</div></div></div>

Signal Type	Description of Conditions
	 <p>• Head/Shoulders – the oscillator formed three consequent peaks, and the mid one is higher than the others.</p> 
No objections to buying	Value of the oscillator grows at the analyzed bar.
No objections to selling	Value of the oscillator falls at the analyzed bar.

Note

Depending on the mode of operation of an Expert Advisor ("Every tick" or "Open prices only") an analyzed bar is either the current bar (with index 0) , or the last formed bar (with index 1) .

Adjustable Parameters

This module has the following adjustable parameters:

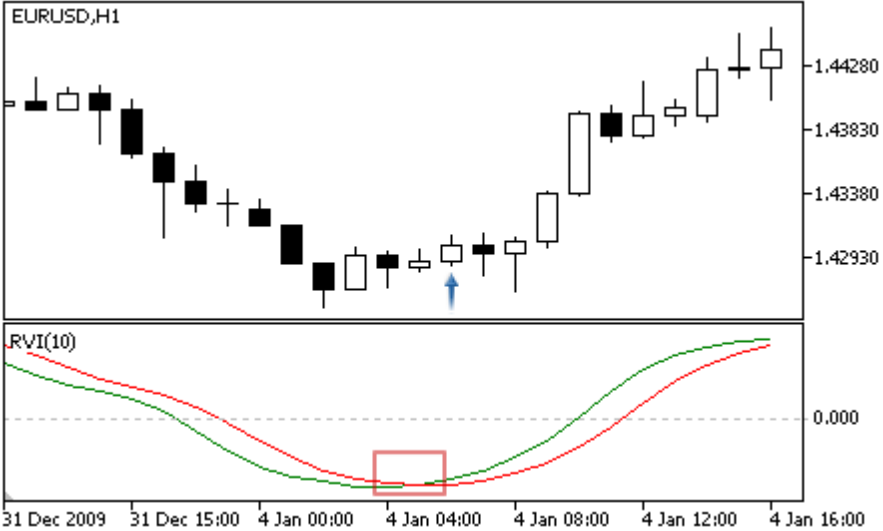
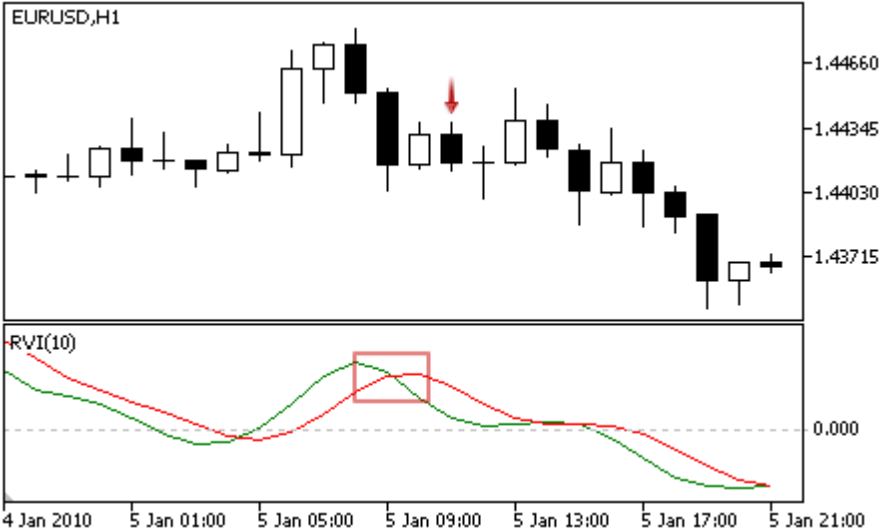
Parameter	Description
Weight	Weight of signal of the module in the interval 0 to 1.
PeriodRSI	Period of calculation of the oscillator.
Applied	A price series used for calculation of the oscillator.

Signals of the Oscillator Relative Vigor Index

This module of signals is based on the market models of the oscillator [Relative Vigor Index](#). The mechanism of making trade decisions based on signals obtained from the modules is described in a [separate section](#).

Conditions of Generation of Signals

Below you can find the description of conditions when the module passes a signal to an Expert Advisor.

Signal Type	Description of Conditions
For buying	<p>Crossing of the main and signal line - the main line is above the signal line at the analyzed bar and below the signal line at the previous one.</p> 
For selling	<p>Crossing of the main and signal line - the main line is below the signal line at the analyzed bar and above the signal line at the previous one.</p> 
No objections to buying	<p>Value of the oscillator grows at the analyzed bar.</p>

Signal Type	Description of Conditions
No objections to selling	Value of the oscillator falls at the analyzed bar.

Note

Depending on the mode of operation of an Expert Advisor ("Every tick" or "Open prices only") an analyzed bar is either the current bar (with index 0) , or the last formed bar (with index 1) .

Adjustable Parameters

This module has the following adjustable parameters:

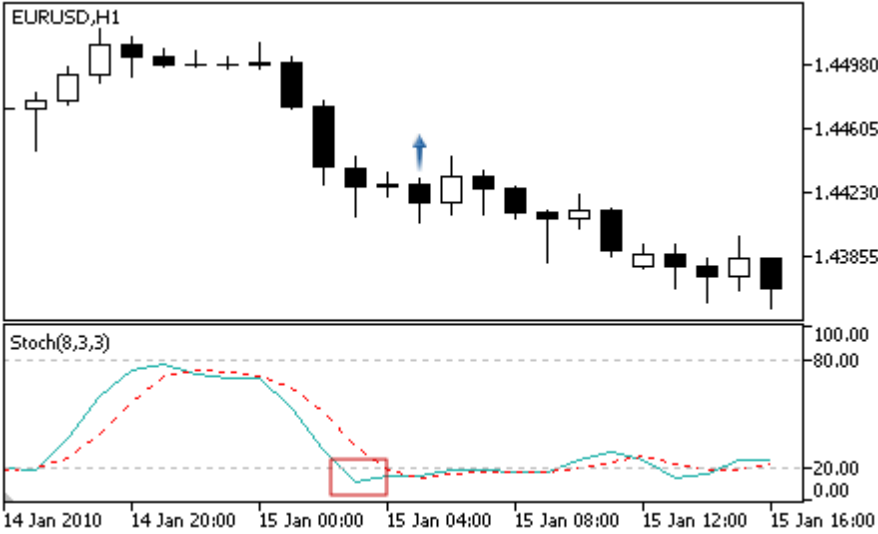
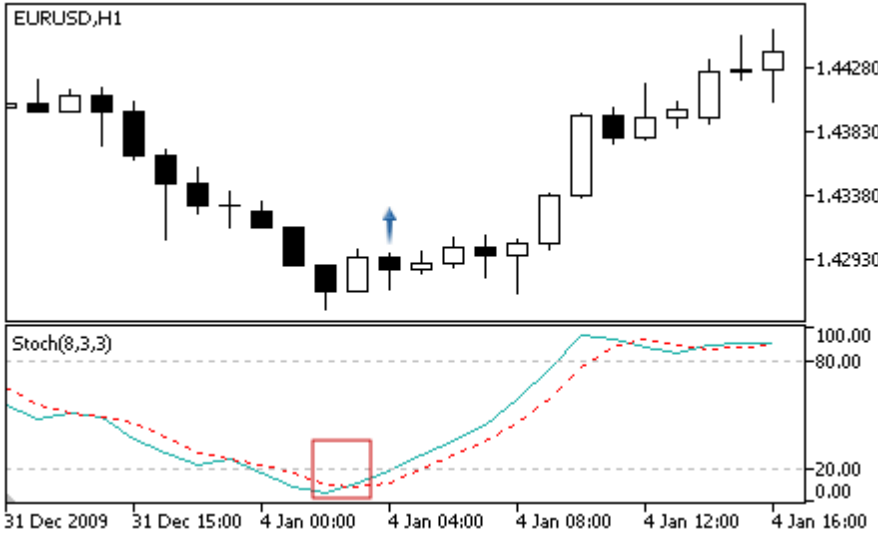
Parameter	Description
Weight	Weight of signal of the module in the interval 0 to 1.
PeriodRVI	Period of calculation of the oscillator.

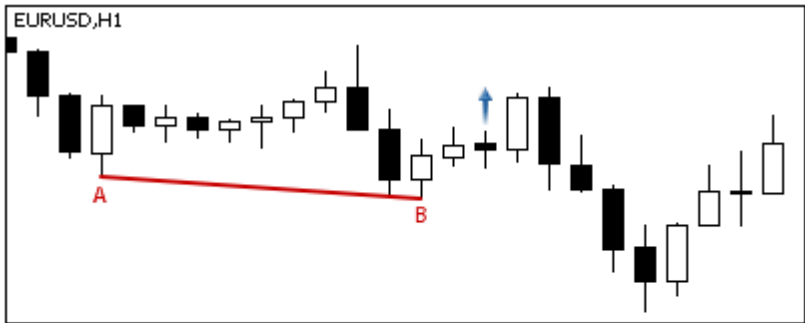
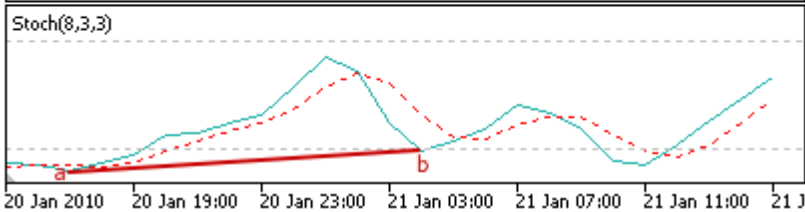
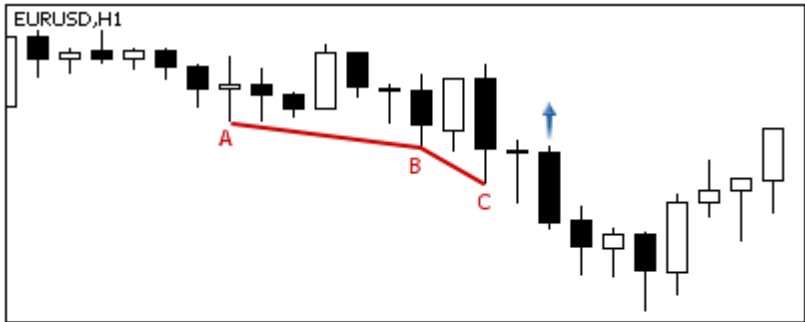
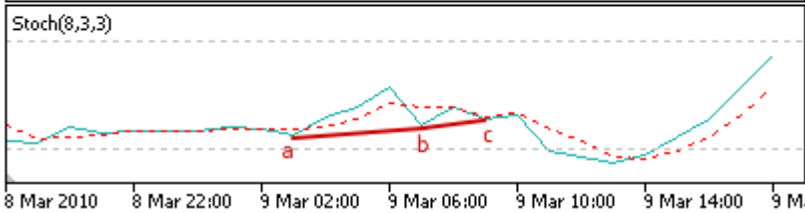
Signals of the Oscillator Stochastic

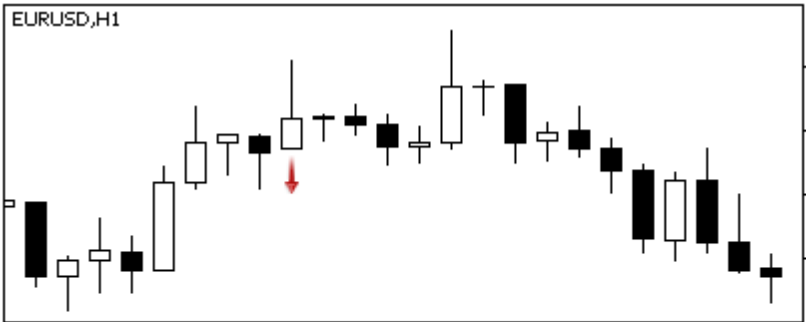
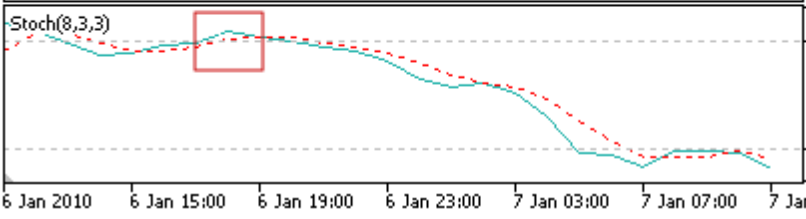
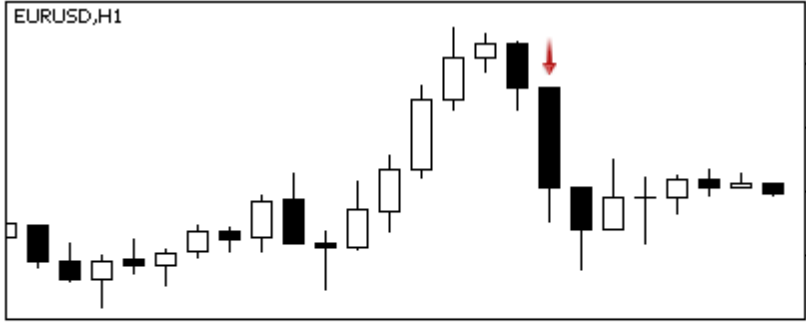
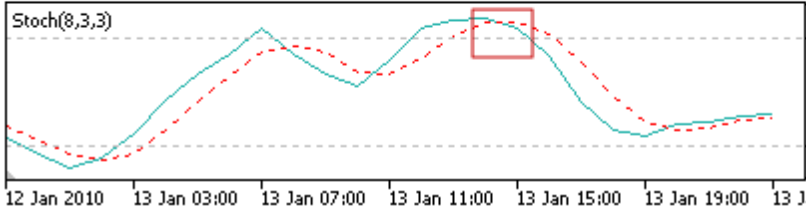
This module of signals based on the market models of the oscillator [Stochastic](#). The mechanism of making trade decisions based on signals obtained from the modules is described in a [separate section](#).

Conditions of Generation of Signals

Below you can find the description of conditions when the module passes a signal to an Expert Advisor.

Signal Type	Description of Conditions
For buying	<div><ul style="list-style-type: none">Reverse - the oscillator turned upwards (the oscillator rises at the analyzed bar and falls at the previous one) . Crossing of the main and signal line - the main line is above the signal line at the analyzed bar and below the signal line at the previous one. Divergence - the first analyzed bottom of the oscillator is higher than the previous one, and the corresponding price bottom is lower than the</div>

Signal Type	Description of Conditions
	<div>previous one.</div> <div> </div> <div><ul style="list-style-type: none">• Double divergence – the oscillator form three consequent bottoms, each of them is higher than the previous one; and the price formed three corresponding bottoms, and each of them is lower than the previous one.</div> <div> </div>
For selling	<ul style="list-style-type: none">• Reverse – the oscillator turned downwards (the oscillator falls at the analyzed bar and rises at the previous one) .

Signal Type	Description of Conditions
	<div><div><div><div>EURUSD,H1</div><div>1.44310 1.44060 1.43810 1.43560</div></div><div><div>Stoch(8,3,3)</div><div>100.00 80.00 20.00 0.00</div></div><div>6 Jan 20106 Jan 15:006 Jan 19:006 Jan 23:007 Jan 03:007 Jan 07:007 Jan 11:00</div></div><div><ul style="list-style-type: none">• Crossing of the main and signal line – the main line is below the signal line at the analyzed bar and above the signal line at the previous one.</div><div><div><div>EURUSD,H1</div><div>1.45620 1.45340 1.45060 1.44780</div></div><div><div>Stoch(8,3,3)</div><div>100.00 80.00 20.00 0.00</div></div><div>12 Jan 201013 Jan 03:0013 Jan 07:0013 Jan 11:0013 Jan 15:0013 Jan 19:0013 Jan 23:00</div></div><div><ul style="list-style-type: none">• Divergence – the first analyzed peak of the oscillator is lower than the previous one, and the corresponding price peak is higher than the previous peak.</div></div>

Signal Type	Description of Conditions
	<div><div><div><div>EURUSD,H1</div></div><div><div>Stoch(8,3,3)</div></div></div><div><ul style="list-style-type: none">• Double divergence – the oscillator formed three consequent peaks, each of them is lower than the previous one; and the price formed three corresponding peaks, each of them is higher than the previous one.</div><div><div><div>EURUSD,H1</div></div><div><div>Stoch(8,3,3)</div></div></div></div>
No objections to buying	Value of the oscillator grows at the analyzed bar.
No objections to selling	Value of the oscillator falls at the analyzed bar.

Note

Depending on the mode of operation of an Expert Advisor ("Every tick" or "Open prices only") an analyzed bar is either the current bar (with index 0) , or the last formed bar (with index 1) .

Adjustable Parameters

This module has the following adjustable parameters:

Parameter	Description
Weight	Weight of signal of the module in the interval 0 to 1.
PeriodK	Period of calculation of the main line of the oscillator.
PeriodD	Period of calculation of the main line of the oscillator.
PeriodSlow	Period of slowing.
Applied	A price series used for calculation of the oscillator.

Signals of the Oscillator Triple Exponential Average

This module of signals is based on the market models of the oscillator [Triple Exponential Average](#). The mechanism of making trade decisions based on signals obtained from the modules is described in a [separate section](#).

Conditions of Generation of Signals

Below you can find the description of conditions when the module passes a signal to an Expert Advisor.

Signal Type	Description of Conditions
For buying	<ul style="list-style-type: none"> Reverse - the oscillator turned upwards (the oscillator rises at the analyzed bar and falls at the previous one) .  <ul style="list-style-type: none"> Crossing the zero level - the main line is above the zero level at the analyzed bar and below the zero level at the previous one.  <ul style="list-style-type: none"> Divergence - the first analyzed bottom of the oscillator is higher than the previous one, and the corresponding price bottom is lower than the

Signal Type	Description of Conditions
	<div>previous one.</div> <div></div> <div>For selling</div> <div><ul style="list-style-type: none">Reverse - the oscillator turned downwards (the oscillator falls at the analyzed bar and rises at the previous one) .</div> <div></div> <div><ul style="list-style-type: none">Crossing the zero level - the main line is below the zero level at the analyzed bar and above the zero level at the previous one.</div>

Signal Type	Description of Conditions
	<div><div><div><div>EURUSD,H1</div></div><div><div>TRIX(12)</div></div><div>6 Jan 2010 7 Jan 00:00 7 Jan 04:00 7 Jan 08:00 7 Jan 12:00 7 Jan 16:00 7 Jan 20:00</div></div><div><ul style="list-style-type: none">• Divergence – the first analyzed peak of the oscillator is lower than the previous one, and the corresponding price peak is higher than the previous peak.</div><div><div><div>EURUSD,H1</div></div><div><div>TRIX(12)</div></div><div>4 Jan 2010 4 Jan 19:00 4 Jan 23:00 5 Jan 03:00 5 Jan 07:00 5 Jan 11:00 5 Jan 15:00</div></div></div>
No objections to buying	Value of the oscillator grows at the analyzed bar.
No objections to selling	Value of the oscillator falls at the analyzed bar.

Note

Depending on the mode of operation of an Expert Advisor ("Every tick" or "Open prices only") an analyzed bar is either the current bar (with index 0) , or the last formed bar (with index 1) .

Adjustable Parameters

This module has the following adjustable parameters:

Parameter	Description
Weight	Weight of signal of the module in the interval 0 to 1.
PeriodTriX	Period of calculation of the oscillator.
Applied	A price series used for calculation of the oscillator.

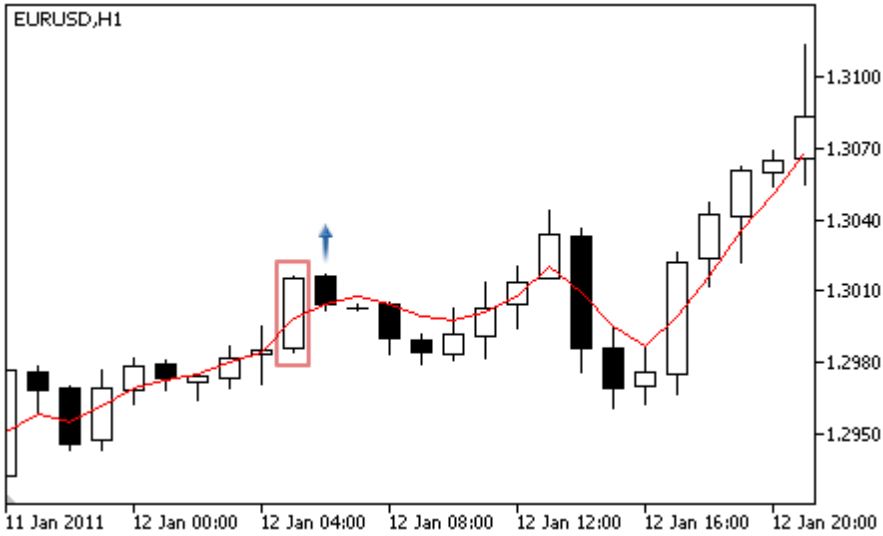
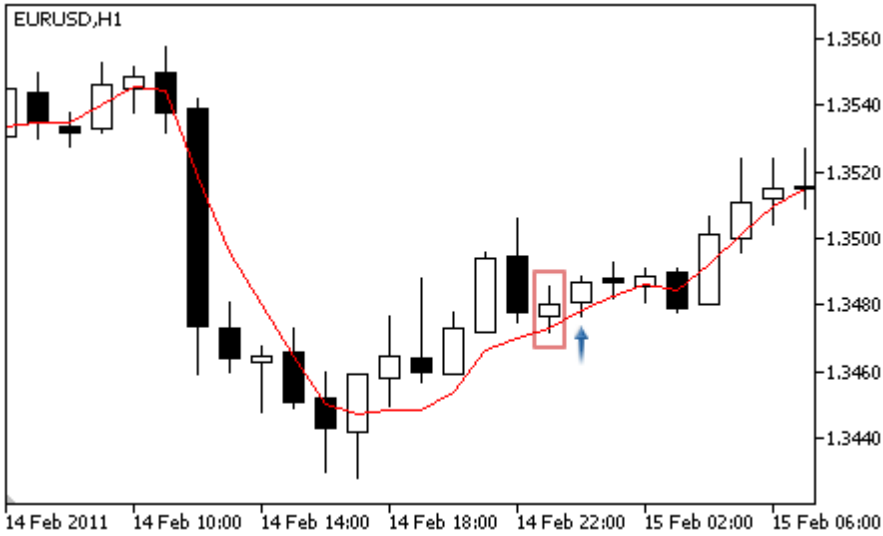
Signals of the Indicator Triple Exponential Moving Average

This module of signals is based on the market models of the indicator [Triple Exponential Moving Average](#). The mechanism of making trade decisions based on signals obtained from the modules is described in a [separate section](#).

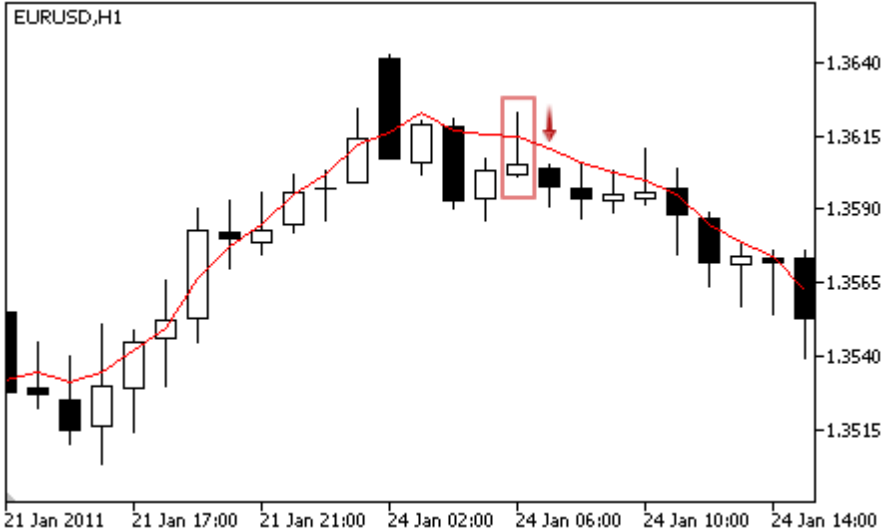
Conditions of Generation of Signals

Below you can find the description of conditions when the module passes a signal to an Expert Advisor.

Signal Type	Description of Conditions
For buying	<div><ul style="list-style-type: none">The price has crossed the indicator downwards (the Open price of the analyzed bar is above the indicator and the Close price is below the indicator) and the indicator rises (weak signal) .<div></div><ul style="list-style-type: none">Moving Average crossover. The price has crossed the indicator upwards (the Open price of the analyzed bar is below the indicator and the Close price is above the indicator) and the indicator rises (strong signal) .</div>

Signal Type	Description of Conditions
	<div><div><div>EURUSD,H1</div></div><div><ul style="list-style-type: none">• The lower shadow of the bar has crossed the indicator (the Open and Close prices of the analyzed bar is above the indicator, and the Low price is below the indicator) and the indicator rises (weak signal) .</div><div><div>EURUSD,H1</div></div></div>
For selling	<ul style="list-style-type: none">• The price has crossed the indicator upwards (the Open price of the analyzed bar is below the indicator and the Close price is above the indicator) and the indicator falls (weak signal) .

Signal Type	Description of Conditions
	<div><div><p>EURUSD,H1</p></div><div><ul style="list-style-type: none">• Moving Average crossover. The price has crossed the indicator downwards (the Open price of the analyzed bar is above the indicator and the Close price is below the indicator) and the indicator falls (strong signal) .</div><div><p>EURUSD,H1</p></div><div><ul style="list-style-type: none">• The upper shadow of the bar has crossed the indicator (the Open and Close prices of the analyzed bar is below the indicator, and the High price is above the indicator) and the indicator falls (weak signal) .</div></div>

Signal Type	Description of Conditions
	<div><div>EURUSD,H1</div><div>21 Jan 2011 21 Jan 17:00 21 Jan 21:00 24 Jan 02:00 24 Jan 06:00 24 Jan 10:00 24 Jan 14:00</div><div>1.3640 1.3615 1.3590 1.3565 1.3540 1.3515</div></div>
No objections to buying	The prices is above the indicator.
No objections to selling	The price is below the indicator.

Note

Depending on the mode of operation of an Expert Advisor ("Every tick" or "Open prices only") an analyzed bar is either the current bar (with index 0) , or the last formed bar (with index 1) .

Adjustable Parameters

This module has the following adjustable parameters:

Parameter	Description
Weight	Weight of signal of the module in the interval 0 to 1.
PeriodMA	Period of averaging of the indicator.
Shift	Shit of the indicator along the time axis (in bars) .
Method	Method of averaging .
Applied	A price series used for calculation of the indicator.

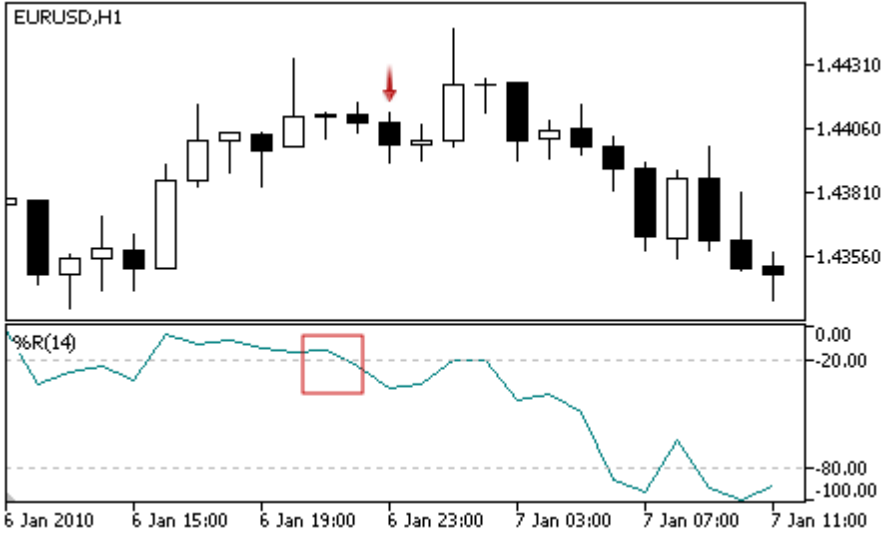
Signals of the Oscillator Williams Percent Range

This module of signals is based on the market models of the oscillator [Williams Percent Range](#). The mechanism of making trade decisions based on signals obtained from the modules is described in a [separate section](#).

Conditions of Generation of Signals

Below you can find the description of conditions when the module passes a signal to an Expert Advisor.

Signal Type	Description of Conditions
For buying	<ul style="list-style-type: none"> Reverse behind the oversold level - the oscillator turned upwards and its value at the analyzed bar is behind the oversold level (default value is -20) .  <ul style="list-style-type: none"> Divergence - the first analyzed bottom of the oscillator is higher than the previous one, and the corresponding price bottom is lower than the previous one. 

Signal Type	Description of Conditions
For selling	<div><ul style="list-style-type: none">Reverse behind the overbought level - the oscillator turned downwards and its value at the analyzed bar is behind the overbought level (default value is -80) .<div></div><ul style="list-style-type: none">Divergence - the first analyzed peak of the oscillator is lower than the previous one, and the corresponding price peak is higher than the previous peak.<div></div></div>
No objections to buying	Value of the oscillator grows at the analyzed bar.
No objections to selling	Value of the oscillator falls at the analyzed bar.

Note

Depending on the mode of operation of an Expert Advisor ("Every tick" or "Open prices only") an analyzed bar is either the current bar (with index 0) , or the last formed bar (with index 1) .

Remember that the oscillator Williams Percent Range has a reversed scale. Its maximum value is -100,

and minimum is 0.

Adjustable Parameters

This module has the following adjustable parameters:

Parameter	Description
Weight	Weight of signal of the module in the interval 0 to 1.
PeriodWPR	Period of calculation of the oscillator.

Trailing Stop classes

This section contains technical details of working with trailing stop classes and description of the relevant components of the MQL5 standard library.

The use of these classes will save time when creating (and testing) of trading strategies.

MQL5 Standard Library (in terms of trading strategies) is placed in the terminal directory, in the Include\Expert\Trailing folder.

Class	Description
<u>CTrailingFixedPips</u>	This class implements Trailing Stop algorithm, based on fixed points
<u>CTrailingMA</u>	This class implements Trailing Stop algorithm, based on the values of Moving Average indicator
<u>CTrailingNone</u>	A gag class, it doesn't uses any Trailing Stop algorithm
<u>CTrailingPSAR</u>	This class implements Trailing Stop algorithm, based on the values of Parabolic SAR indicator

CTrailingFixedPips

CTrailingFixedPips is a class with implementation of Trailing Stop algorithm, based on fixed points trailing.

If position has Stop Loss price, it checks the minimal allowed Stop Loss distance to the current price. If its value is lower, that Stop Loss level, it suggests to set new Stop Loss price. For this case if position has Take Profit price, it suggests to set new Take Profit price.

If Expert Advisor has been [initialized](#) with the flag `every_tick=false`, it will perform all operations (trading, trailing, etc) only at the new bar. For this case Take profit level can be used. It will allow you to close opened position at Take Profit price before the new bar will be completed.

Description

CTrailingFixedPips implements the Trailing Stop algorithm, based on positions trailing with the fixed points.

Declaration

```
class CTrailingFixedPips: public CExpertTrailing
```

Title

```
#include <Expert\Trailing\CTrailingFixedPips.mqh>
```

Class Methods

Initialization	
StopLevel	Sets the value of Stop Loss level
ProfitLevel	Sets the value of Take Profit level
virtual ValidationSettings	Checks the settings
Check Trailing Methods	
virtual CheckTrailingStopLong	Check Trailing Stop conditions of long position
virtual CheckTrailingStopShort	Check Trailing Stop conditions of short position

StopLevel

Sets the value Stop Loss level (in points) .

```
void StopLevel(  
    int stop_level    // Stop Loss level  
)
```

Parameters

stop_loss

[in] The value of Stop Loss level (in conventional 2/4-digit points) .

Note

If Stop Loss level is equal to 0, the Trailing Stop is not used.

ProfitLevel

Sets the value of Take Profit level (in points) .

```
void ProfitLevel(  
    int profit_level    // Take profit level  
)
```

Parameters

profit_level

[in] The value of Take Profit level (in conventional 2/4-digit points) .

Note

If profit level is equal to 0, the Trailing Stop is not used.

ValidationSettings

Checks the settings.

```
virtual bool ValidationSettings()
```

Returned value

true if successful, otherwise false.

Note

The function checks Take Profit and Stop Loss levels. The correct values are 0 and values, greater than minimal stop for stop orders for the symbol.

CheckTrailingStopLong

Checks Trailing Stop conditions of long position.

```
virtual bool CheckTrailingStopLong (
    CPositionInfo* position,      // CPositionInfo object pointer
    double& sl,                  // Stop Loss price
    double& tp                    // Take Profit price
)
```

Parameters

position

[in] Pointer to [CPositionInfo](#) object.

sl

[in][out] Variable for Stop Loss price.

tp

[in][out] Variable for Take Profit price.

Returned value

true if conditions are satisfied, otherwise false.

Note

If Stop Loss level is equal to 0, the Trailing Stop is not used. If position already has Stop Loss price, its value is assumed as a base price, otherwise the position open price is assumed as a base price.

If the current Bid price is higher than base price+stop loss level, it suggests to set new Stop Loss price. In this case, If position already has Take Profit price, it suggests to set new Take Profit price equal to Bid price+take profit level.

CheckTrailingStopShort

Checks Trailing Stop conditions of short position.

```
virtual bool CheckTrailingStopShort(  
    CPositionInfo* position,      // CPositionInfo object pointer  
    double& sl,                  // Stop Loss price  
    double& tp                    // Take Profit price  
)
```

Parameters

position

[in] Pointer to [CPositionInfo](#) object.

sl

[in][out] Variable for Stop Loss price.

tp

[in][out] Variable for Take Profit price.

Returned value

true if conditions are satisfied, otherwise false.

Note

If Stop Loss level is equal to 0, the Trailing Stop is not used. If position already has Stop Loss price, its value is assumed as a base price, otherwise the position open price is assumed as a base price.

If the current Ask price is lower than base price-stop loss level, it suggests to set new Stop Loss price. In this case, If position already has Take Profit price, it suggests to set new Take Profit price equal to Ask price-take profit level.

CTrailingMA

CTrailingMA is a class with implementation of Trailing Stop algorithm, based on the values of moving average indicator.

Description

CTrailingMA class implements Trailing Stop algorithm, based on the values of moving average indicator of the previous (completed) bar.

Declaration

```
class CTrailingMA: public CExpertTrailing
```

Title

```
#include <Expert\Trailing\TrailingMA.mqh>
```

Class Methods

Initialization	
Period	Sets period of moving average
Shift	Sets shift of moving average
Method	Sets smoothing method of moving average
Applied	Sets applied price of moving average
virtual InitIndicators	Initializes indicators and time series
virtual ValidationSettings	Checks the settings
Check Trailing Methods	
virtual CheckTrailingStopLong	Check Trailing Stop conditions of long position
virtual CheckTrailingStopShort	Check Trailing Stop conditions of short position

Period

Sets period of moving average.

```
void Period(  
    int period    // Smoothing period  
)
```

Parameters

period

[in] Period of moving average.

Shift

Sets shift of moving average.

```
void Shift(  
    int shift    // Shift  
)
```

Parameters

shift

[in] Shift of moving average.

Method

Sets smoothing method of moving average.

```
void Method(  
    ENUM_MA_METHOD method      // Smoothing method  
)
```

Parameters

method

[in] [Smoothing method](#) of moving average indicator.

Applied

Sets applied price of moving average.

```
void Applied(  
    ENUM_APPLIED_PRICE applied // Applied price  
)
```

Parameters

applied

[in] [Applied price](#) of moving average.

InitIndicators

Initializes indicators and time series.

```
virtual bool InitIndicators(  
    CIndicators* indicators // CIndicators collection pointer  
)
```

Parameters

indicators

[in] Pointer to indicators and time series collection ([CExpert](#) class member) .

Returned value

true if successful, otherwise false.

ValidationSettings

Checks the settings.

```
virtual bool ValidationSettings()
```

Returned value

true if successful, otherwise false.

Note

The function checks the period of moving average, the correct values are positive.

CheckTrailingStopLong

Checks Trailing Stop conditions of long position.

```
virtual bool CheckTrailingStopLong (
    CPositionInfo* position,      // CPositionInfo object pointer
    double& sl,                  // Stop Loss price
    double& tp                    // Take Profit price
)
```

Parameters

position

[in] Pointer to [CPositionInfo](#) object.

sl

[in][out] Variable for Stop Loss price.

tp

[in][out] Variable for Take Profit price.

Returned value

true if conditions are satisfied, otherwise false.

Note

The first it calculates the maximal allowed Stop Loss price, closest to the current price and calculates Stop Loss price using the values of moving average indicator of the previous (completed) bar.

If position already has Stop Loss price, its value is assumed as a base price, otherwise the base price is the open price of the position.

If the calculated Stop Loss price is higher than base price and lower than maximal allowed Stop Loss price, it suggests to set new Stop Loss price.

CheckTrailingStopShort

Checks Trailing Stop conditions of short position.

```
virtual bool CheckTrailingStopShort(  
    CPositionInfo* position,      // CPositionInfo object pointer  
    double& sl,                  // Stop Loss price  
    double& tp                    // Take Profit price  
)
```

Parameters

position

[in] Pointer to [CPositionInfo](#) object.

sl

[in][out] Variable for Stop Loss price.

tp

[in][out] Variable for Take Profit price.

Returned value

true if conditions are satisfied, otherwise false.

Note

The first it calculates the minimal allowed Stop Loss price, closest to the current price and calculates Stop Loss price using the values of moving average indicator of the previous (completed) bar.

If position already has Stop Loss price, its value is assumed as a base price, otherwise the base price is the open price of the position.

If the calculated Stop Loss price is higher than base price and lower than minimal allowed Stop Loss price, it suggests to set new Stop Loss price.

CTrailingNone

CTrailingNone is a gag class. This class should be used at initialization of Trailing object if your strategy doesn't use Trailing Stop.

Description

CTrailingNone class doesn't implement any Trailing Stop algorithms. The methods of checking Trailing Stop conditions always return false.

Declaration

```
class CTrailingNone: public CExpertTrailing
```

Title

```
#include <Expert\Trailing\TrailingNone.mqh>
```

Class Methods

Check Trailing Methods	
virtual CheckTrailingStopLong	A gag method for check Trailing Stop conditions of long position
virtual CheckTrailingStopShort	A gag method for check Trailing Stop conditions of short position

CheckTrailingStopLong

Checks Trailing Stop conditions of long position.

```
virtual bool CheckTrailingStopLong (  
    CPositionInfo* position,      // CPositionInfo object pointer  
    double& sl,                  // Stop Loss price  
    double& tp                    // Take Profit price  
)
```

Parameters

position

[in] Pointer to [CPositionInfo](#) object.

sl

[in][out] Variable for Stop Loss price.

tp

[in][out] Variable for Take Profit price.

Returned value

true if conditions are satisfied, otherwise false.

Note

The function always returns false.

CheckTrailingStopShort

Checks Trailing Stop conditions of short position.

```
virtual bool CheckTrailingStopShort(  
    CPositionInfo* position,      // CPositionInfo object pointer  
    double& sl,                  // Stop Loss price  
    double& tp                   // Take Profit price  
)
```

Parameters

position

[in] Pointer to [CPositionInfo](#) object.

sl

[in][out] Variable for Stop Loss price.

tp

[in][out] Variable for Take Profit price.

Returned value

true if conditions are satisfied, otherwise false.

Note

The function always returns false.

CTrailingPSAR

CTrailingPSAR is a class with implementation of Trailing Stop algorithm, based on the values of of Parabolic SAR indicator.

Description

CTrailingPSAR class implements the Trailing Stop algorithm, based on the values of Parabolic SAR indicator of the previous (completed) bar.

Declaration

```
class CTrailingPSAR: public CExpertTrailing
```

Title

```
#include <Expert\Trailing\TrailingParabolicSAR.mqh>
```

Class Methods

Initialization	
Step	Sets the value of step of Parabolic SAR indicator
Maximum	Sets the value of maximum of Parabolic SAR indicator
virtual InitIndicators	Initializes indicators and time series
Check Trailing Methods	
virtual CheckTrailingStopLong	Check conditions of trailing stop of long position
virtual CheckTrailingStopShort	Check conditions of trailing stop of short position

Step

Sets the value of step of Parabolic SAR indicator.

```
void Step(  
    double step    // Step  
)
```

Parameters

step

[in] The value of Step of Parabolic SAR indicator.

Maximum

Sets the value of maximum of Parabolic SAR indicator.

```
void Maximum(  
    double maximum    // Maximum  
)
```

Parameters

maximum

[in] The value of maximum of Parabolic SAR indicator.

InitIndicators

Initializes indicators and time series.

```
virtual bool InitIndicators(  
    CIndicators* indicators // CIndicators collection pointer  
)
```

Parameters

indicators

[in] Pointer to indicators and time series collection ([CExpert](#) class member) .

Returned value

true if successful, otherwise false.

CheckTrailingStopLong

Checks Trailing Stop conditions of long position.

```
virtual bool CheckTrailingStopLong (
    CPositionInfo* position,      // указатель
    double& sl,                  // ссылка
    double& tp                    // ссылка
)
```

Parameters

position

[in] Pointer to [CPositionInfo](#) object.

sl

[in][out] Variable for Stop Loss price.

tp

[in][out] Variable for Take Profit price.

Returned value

true if conditions are satisfied, otherwise false.

Note

The first it calculates the maximal allowed Stop Loss price, closest to the current price and calculates Stop Loss price using the values of Parabolic SAR indicator of the previous (completed) bar.

If position already has Stop Loss price, its value is assumed as a base price, otherwise the position open price is assumed as a base price.

If the calculated Stop Loss price is higher than base price and lower than maximal allowed Stop Loss price, it suggests to set new Stop Loss price.

CheckTrailingStopShort

Checks Trailing Stop conditions of short position.

```
virtual bool CheckTrailingStopShort (  
    CPositionInfo* position,      // указатель  
    double& sl,                  // ссылка  
    double& tp                    // ссылка  
)
```

Parameters

position

[in] Pointer to [CPositionInfo](#) object.

sl

[in][out] Variable for Stop Loss price.

tp

[in][out] Variable for Take Profit price.

Returned value

true if conditions are satisfied, otherwise false.

Note

The first it calculates the minimal allowed Stop Loss price, closest to the current price and calculates Stop Loss price using the values of Parabolic SAR indicator of the previous (completed) bar.

If position already has Stop Loss price, its value is assumed as a base price, otherwise the position open price is assumed as a base price.

If the calculated Stop Loss price is higher than base price and lower than minimal allowed Stop Loss price, it suggests to set new Stop Loss price.

Money Management classes

This section contains technical details of working with money and risk management classes and description of the relevant components of the MQL5 standard library.

The use of these classes will save time when creating (and testing) of trading strategies.

MQL5 Standard Library (in terms of money and risk management classes) is placed in the terminal directory, in the Include\Expert\Money\ folder.

Class	Description
<u>CMoneyFixedLot</u>	This class implements money management algorithm, based on trading with predefined fixed lot size.
<u>CMoneyFixedMargin</u>	This class implements money management algorithm, based on trading with predefined fixed margin.
<u>CMoneyFixedRisk</u>	This class implements money management algorithm, based on trading with predefined risk.
<u>CMoneyNone</u>	This class implements money management algorithm, based on trading with minimal allowed lot size.
<u>CMoneySizeOptimized</u>	This class implements money management algorithm, based on trading with variable lot size, depending on results of the previous deals.

CMoneyFixedLot

CMoneyFixedLot is the class money management algorithm, based on trading with predefined fixed lot size.

Description

CMoneyFixedLot implements money management algorithm, based on trading with predefined fixed lot size.

Declaration

```
class CMoneyFixedLot: public CExpertMoney
```

Title

```
#include <Expert\Money\MoneyFixedLot.mqh>
```

Class Methods

Initialization	
Lots	Sets trading volume
virtual ValidationSettings	Checks the settings
Money and Risk Management Methods	
virtual CheckOpenLong	Gets trade volume for long position
virtual CheckOpenShort	Gets trade volume for short position

Lots

Sets trading volume (in lots) .

```
void Lots(  
    double lots    // Lots  
)
```

Parameters

lots

[in] Trading volume (in lots) .

ValidationSettings

Checks the settings.

```
virtual bool ValidationSettings()
```

Returned value

true if successful, otherwise false.

Note

Checks the specified trading volume for correctness.

CheckOpenLong

Gets trade volume for long position.

```
virtual double CheckOpenLong(  
    double price,      // Price  
    double sl          // Stop Loss price  
)
```

Parameters

price

[in] Price.

sl

[in] Stop Loss price.

Returned value

Trade volume for long position.

Note

The function always returns the fixed trade volume, defined by [Lots](#) method.

CheckOpenShort

Gets trade volume for short position.

```
virtual double CheckOpenShort(  
    double price,      // Price  
    double sl          // Stop Loss price  
)
```

Parameters

price

[in] Price.

sl

[in] Stop Loss price.

Returned value

Trade volume for short position.

Note

The function always returns the fixed trade volume, defined by [Lots](#) method.

CMoneyFixedMargin

CMoneyFixedMargin is the class money management algorithm, based on trading with predefined fixed margin.

Description

CMoneyFixedMargin implements money management algorithm, based on trading with predefined fixed margin.

Declaration

```
class CMoneyFixedMargin: public CExpertMoney
```

Title

```
#include <Expert\Money\MoneyFixedMargin.mqh>
```

Class Methods

Money and Risk Management Methods	
virtual CheckOpenLong	Gets trade volume for long position
virtual CheckOpenShort	Gets trade volume for short position

CheckOpenLong

Gets trade volume for long position.

```
virtual double CheckOpenLong(  
    double price,      // Price  
    double sl          // Stop Loss price  
)
```

Parameters

price

[in] Price.

sl

[in] Stop Loss price.

Returned value

Trade volume for long position.

Note

The function returns trade volume for long position, it uses the fixed margin. The margin is defined by Percent parameter of [CExpertMoney](#) base class.

CheckOpenShort

Gets trade volume for short position.

```
virtual double CheckOpenShort(  
    double price,      // Price  
    double sl          // Stop Loss price  
)
```

Parameters

price

[in] Price.

sl

[in] Stop Loss price.

Returned value

Trade volume for short position.

Note

The function returns trade volume for short position, it uses the fixed margin. The margin is defined by Percent parameter of [CExpertMoney](#) base class.

CMoneyFixedRisk

CMoneyFixedRisk is a class with implementation of money management algorithm with fixed predefined risk.

Description

CMoneyFixedRisk class implements the money management algorithm with fixed predefined risk.

Declaration

```
class CMoneyFixedRisk: public CExpertMoney
```

Title

```
#include <Expert\Money\MoneyFixedRisk.mqh>
```

Class Methods

Money and Risk Management Methods	
virtual CheckOpenLong	Gets trade volume for long position
virtual CheckOpenShort	Gets trade volume for short position

CheckOpenLong

Gets trade volume for long position.

```
virtual double CheckOpenLong(  
    double price,      // Price  
    double sl          // Stop Loss price  
)
```

Parameters

price

[in] Price.

sl

[in] Stop Loss price.

Returned value

Trade volume for long position.

Note

The function returns trade volume for long position, it uses the fixed risk. The risk is defined by Percent parameter of [CExpertMoney](#) base class.

CheckOpenShort

Gets trade volume for short position.

```
virtual double CheckOpenShort(  
    double price,      // Price  
    double sl          // Stop Loss price  
)
```

Parameters

price

[in] Price.

sl

[in] Stop Loss price.

Returned value

Trade volume for short position.

Note

The function returns trade volume for short position, it uses the fixed risk. The risk is defined by Percent parameter of [CExpertMoney](#) base class.

CMoneyNone

CMoneyNone is a class with implementation of trading algorithm with minimal allowed lot.

Description

CMoneyNone class implements trading with minimal allowed lot.

Declaration

```
class CMoneyNone: public CExpertMoney
```

Title

```
#include <Expert\Money\MoneyNone.mqh>
```

Class Methods

Initialization	
virtual ValidationSettings	Checks the settings
Money and Risk Management Methods	
virtual CheckOpenLong	Gets trade volume for long position
virtual CheckOpenShort	Gets trade volume for short position

ValidationSettings

Checks the settings.

```
virtual bool ValidationSettings()
```

Returned value

true if successful, otherwise false.

Note

The function always returns true.

CheckOpenLong

Gets trade volume for long position.

```
virtual double CheckOpenLong(  
    double price,      // Price  
    double sl          // Stop Loss price  
)
```

Parameters

price

[in] Price.

sl

[in] Stop Loss price.

Returned value

Trade volume for long position.

Note

The function always returns the minimal lot size.

CheckOpenShort

Gets trade volume for long position.

```
virtual double CheckOpenShort(  
    double price,      // Price  
    double sl          // Stop Loss price  
)
```

Parameters

price

[in] Price.

sl

[in] Stop Loss price.

Returned value

Trade volume for short position.

Note

The function always returns the minimal lot size.

CMoneySizeOptimized

CMoneySizeOptimized is a class with implementation of money management algorithm, based on trading with variable lot size, depending on results of the previous deals.

Description

CMoneySizeOptimized implements money management algorithm, based on trading with variable lot size, depending on results of the previous deals.

Declaration

```
class CMoneySizeOptimized: public CExpertMoney
```

Title

```
#include <Expert\Money\MoneySizeOptimized.mqh>
```

Class Methods

Initialization	
DecreaseFactor	Sets the value of decrease factor
virtual ValidationSettings	Checks the settings
Money and Risk Management Methods	
virtual CheckOpenLong	Gets trade volume for long position
virtual CheckOpenShort	Gets trade volume for short position

DecreaseFactor

Sets the value of decrease factor.

```
void DecreaseFactor(  
    double decrease_factor    // Decrease factor  
)
```

Parameters

decrease_factor

[in] Decrease factor.

Note

The DecreaseFactor defines the volume decreasing coefficient (compared with the volume of previous position) for the case of consecutive loss trades.

ValidationSettings

Checks the settings.

```
virtual bool ValidationSettings()
```

Returned value

true if successful, otherwise false.

Note

If the value of decrease factor is negative, it returns false, otherwise it returns true.

CheckOpenLong

Gets trade volume for long position.

```
virtual double CheckOpenLong(  
    double price,      // Price  
    double sl          // Stop Loss price  
)
```

Parameters

price

[in] Price.

sl

[in] Stop Loss price.

Returned value

Trade volume for long position.

Note

The function returns trade volume for long position, the volume dependent on results of the previous deals.

CheckOpenShort

Gets trade volume for short position.

```
virtual double CheckOpenShort(  
    double price,      // Price  
    double sl          // Stop Loss price  
)
```

Parameters

price

[in] Price.

sl

[in] Stop Loss price.

Returned value

Trade volume for long position.

Note

The function returns trade volume for short position, the volume dependent on results of the previous deals.

从 MQL4 到 MQL5

MQL5是前任MQL4语言的发展，其中编辑进去许多指标，脚本和EA交易。尽管事实上，新的程序语言最大化的与前一代相匹配，但在这些语言之间，仍有区别，当转换程序时，这些区别应该标注出来。

对于了解MQL4的程序员来说，新的MQL5语言，该章节涵盖了促进代码改编进化的所有信息。

首先应该标明的是：

- 新语言不包括函数start()，init() 和 deinit()；
- 指标缓冲区数量不限制；
- 在下载EA程序后（或者其他MQL5程序），dll立即被加载；
- 检测缩写的逻辑条件；
- 当数组超过限制时，结束当前操作（紧急的-错误输出）；
- 像C++一样运算符优先；
- 该语言提供隐式类型（甚至从字符串到数字）；
- 局部变量不能自动初始化（除了字符串）；
- 普通本地数组自动删除。

特殊函数 init, start and deinit

MQL4语言只包括3个预定函数可以操作指标，脚本或者EA交易（不包括files *.mqh和数据库文件）。在MQL5中没有那样的函数，但有与之相类似的函数，图表表明了近似函数。

MQL4	MQL5
init	OnInit
start	OnStart
deinit	OnDeinit

在MQL4中，函数 [OnInit](#) 和 [OnDeinit](#) 在 init 和 deinit 程序中执行了相同的任务-他们是为本地代码设计的，一定在MQL5程序的初始化过程中执行，可以因此重命名函数，或者以他们本身的形式呈现，但是在类似位置这些函数可以添加调用。

示例：

```
void OnInit()
{
    //--- 调用函数去初始化
    init();
}
void OnDeinit(const int reason)
{
    //--- 调用无法初始化函数
    deinit();
    //---
}
```

开始函数只在脚本中被[OnStart](#)代替，在EA交易和指标中应该分别重命名成[OnTick](#)和[OnCalculate](#)。在MQL5程序操作过程中，这些代码可以执行，并保存3个函数中：

mql5程序	主函数
脚本	OnStart
指标	OnCalculate
EA交易	OnTick

如果指标或者代码不包括主函数，或者函数名称与要求的不相符，该函数的调用就不能执行。这表示了，如果脚本的资源代码不包括OnStart，该代码会以一个EA交易编辑。

如果指标代码不包括OnCalculate函数，该指标的编辑不能完成。

预定义变量

在MQL5中有诸如Ask, Bid, Bars的预定义变量，变量点和数字的拼写有稍许不同：

MQL4	MQL5
数字	_ Digits
点	_ Point
	_ LastError
	_ Period
	_ Symbol
	_ StopFlag
	_ UninitReason

访问时间序列

MQL5中没有像Open [], High [], Low [], Close [], Volume [] and Time []这样预定义的时间序列。时间序列必要的深度能用相关[访问时间序列函数](#)来设定。

EA交易

MQL5的EA交易不需要强制存在处理新订单号收据[事件](#)的函数-OnTick，如在MQL4中一样（当接收新订单时执行MQL4的启动函数），因为在MQL5中EA交易可以包含几种类型的预定义处理器函数。

- [OnTick](#) – 新订单号收据；
- [OnTimer](#) – 时间数据；
- [OnTrade](#) - 交易事件；
 - [OnChartEvent](#) – 键盘鼠标输入事件，图解物件移动事件，完成文本编辑事件输入LabelEdit物件领域中的；
 - [OnBookEvent](#) – 市场深度状态更改事件。

自定义指标

在MQL4中，指标缓冲区的数量是有限的，不能超过8。而在MQL5中则没有这个限制，但是要记得每个指标缓冲区需要在程序端分配部分内存，所以，这个新功能也不能滥用。

MQL4只提供了6种自定义指标绘图；而MQL5现在提供了18种[绘画类型](#)。绘画类型的名称不变，但是指标的图解表示意义却显著变化。

指标缓冲区中索引趋势也并不相同。默认情况下，在MQL5中，所有指标缓冲区都有常用数组行为，例如0索引元素是历史记录中最古老的一个，随着指数增加，数据也从旧的换成新的。

受MQL4保护的工作[自定义指标](#)的唯一函数是[SetIndexBuffer](#)。但是调用改变；现在应该指定[存储在数组中的数据类型](#)，连接指标缓冲区。

自定义指标属性也已更改并发展。添加了[访问时间序列](#)的新函数，所以全部算法需要重新审议。

图解物件

在MQL5中图解物件的数量显著增加。此外，图解物件现在可以及时置于时间表的图表中且可精确到秒-现在物件定位点不会四舍五入到当前价格图表开盘时间柱上。对于箭头物件，可以指定文本和标签[绑定方式](#)，并且对于标签，可以设置按钮，图表，位图标签和编辑定位[定位物件的图表角](#)。