```
//+------------------------------------------------------------------+
//|                                  Test Moving Average.mq4 |
//+------------------------------------------------------------------+
// Модификация советника Moving Average от MetaQuotes Software Corp.(http://www.metaquotes.net/).
// Позволяет проводить оптимизацию по критерию Прибыль/Просадка.
// Предназначен исключительно для демонстрации подхода к контролю процесса оптимизации в тестере
// Внимание:
// Произведение MovingPeriodStepsNumber на MovingShiftStepsNumber ДОЛЖНО быть равно TestsNumber.
// В настройках параметров оптимизации необходимо задать диапазон ТОЛЬКО для Counter,
// причём он ДОЛЖЕН быть от 1 до TestsNumber с шагом 1.
// Генетический алгоритм должен быть отключён.
// Ещё одна возможная неприятность связана с кешированием результатов предыдущих тестирований.
// Когда результаты берутся из кеша, функции init() и deinit() не вызываются.
// Это может нарушить работу логики советника. Автору известны два способа перестраховки:
// перекомпиляция советника или закрытие/пауза/открытие окна тестера.
// Вмешательство кеша можно детектировать с помощью независимого подсчёта проходов.
// Код для такого подсчёта работал в первой версии советника, в этой версии он закомментирован.
#define MAGICMA  20050610
extern int Counter                = 1;    // Счётчик проходов тестера
extern int TestsNumber            = 200;  // Контрольная цифра - общее число проходов
extern int MovingPeriodStepsNumber    = 20;   // Число шагов оптимизации для MovingPeriod
extern int MovingShiftStepsNumber     = 10;   // Число шагов оптимизации для MovingShift
extern double MovingPeriodLow         = 150;  // Нижняя граница диапазона оптимизации для MovingPeriod
extern double MovingShiftLow          = 1;    // Нижняя граница диапазона оптимизации для MovingShift
extern double MovingPeriodStep        = 1;    // Шаг оптимизации для MovingPeriod
extern double MovingShiftStep         = 1;    // Шаг оптимизации для MovingShift
// Далее параметры штатного Moving Average
extern double MovingPeriod          = 12;
extern double MovingShift           = 6;
extern double Lots                  = 0.1;
extern double MaximumRisk           = 0.02;
extern double DecreaseFactor        = 3;
//int TestsCnt;             // Независимый счётчик проходов
int FilePtr;              //  Положение файлового указателя
double Criterion;         //  Критерий оптимизации
int h;                    //  хэндл файла
double Data[][4];         //  Массив для данных по тестированиям
double StartBalance;      //  Стартовый баланс
double MaxEqu;            //  Максимум средств
double MaxDD;             //  Максимальная просадка
//+------------------------------------------------------------------+
//| expert initialization function                                   |
```

```
//+------------------------------------------------------------------+
int init() {
// Пусть наша добавка к коду будет включаться только в тестере
// и при условии отличного от нуля количества проходов.
  if (IsTesting() && TestsNumber > 0) {
// Для ускорения процесса будем поддерживать сквозной указатель файловой позиции
// с помощью глобальной переменной FilePtr
// На первом проходе инициализируем её нулём.
    if (GlobalVariableCheck("FilePtr")==false || Counter == 1) {
      FilePtr = 0;
      GlobalVariableSet("FilePtr",0);
    } else {
      FilePtr = GlobalVariableGet("FilePtr");
    }
/*
// Код независимого счётчика проходов
    if (GlobalVariableCheck("TestsCnt")==false || Counter == 1) {
      TestsCnt = 0;
      GlobalVariableSet("TestsCnt",0);
    } else {
      TestsCnt = GlobalVariableGet("TestsCnt");
    }
*/
// Поскольку оптимизация будет производиться по счётчику, необходимо обеспечить
перебор параметров
    MovingPeriod = MovingPeriodLow+((Counter-1)/
MovingShiftStepsNumber)*MovingPeriodStep;
    MovingShift =
MovingShiftLow+((Counter-1)%MovingShiftStepsNumber)*MovingShiftStep;
// Наконец, инициализируем переменные, необходимые для расчёта критерия
оптимизации
    StartBalance = AccountBalance();
    MaxEqu = 0;
    MaxDD = 0;
  }
  return(0);
}
//+------------------------------------------------------------------+
//| expert deinitialization function                                 |
//+------------------------------------------------------------------+
int deinit() {
// Здесь нам небходимо подвести итог тестирования, запомнить его,
// а также принять меры для передачи информации следующей реинкарнации
советника.
  if (IsTesting() && TestsNumber > 0) {
/*
// Код независимого счётчика проходов
    TestsCnt++;
    GlobalVariableSet("TestsCnt",TestsCnt);
*/
//  Рассчитаем критерий оптимизации
    if (AccountEquity() > MaxEqu) MaxEqu = AccountEquity();
    if (MaxEqu-AccountEquity() > MaxDD) MaxDD = MaxEqu-AccountEquity();
    Criterion = (AccountBalance()-StartBalance)/MaxDD;
```

```
// Теперь займёмся передачей информации. Помимо критерия оптимизации будем
записывать
// значения оптимизируемых параметров и номер прохода тестера.
// Мы должны обработать три ситуации: первый запуск, последний запуск и всё
остальное.
// Для их разделения будем использовать счётчик проходов тестера (Counter).
   if (Counter == 1) {
// Первый проход, создаём/обнуляем файл данных.
     h=FileOpen("test.txt",FILE_CSV|FILE_WRITE,';');
     FileWrite(h,Criterion,MovingPeriod,MovingShift,Counter);
// Запомним в глобальной переменной положение файлового указателя после записи
     FilePtr = FileTell(h);
     GlobalVariableSet("FilePtr",FilePtr);
     FileClose(h);
   } else {
//  После того как первый запуск обработан, данные в файл будем дописывать
     h=FileOpen("test.txt",FILE_CSV|FILE_READ|FILE_WRITE,';');
// Пришло время воспользоваться записанным в глобальной переменной файловым
указателем
     FilePtr = GlobalVariableGet("FilePtr");
     FileSeek(h,FilePtr, SEEK_SET);
     FileWrite(h,Criterion,MovingPeriod,MovingShift,Counter);
// И снова запомним положение файлового указателя
     FilePtr = FileTell(h);
     GlobalVariableSet("FilePtr",FilePtr);
// Теперь вспомним о последнем запуске
     if (Counter == TestsNumber) {
// Нужно обработать результаты оптимизации
// Подготовим массив для данных
       ArrayResize(Data,TestsNumber);
// Возвращаем файловый указатель в начало
       FileSeek(h,0,SEEK_SET);
// Читаем результаты всех тестирований из файла
       int i = 0;
       while (i<TestsNumber && FileIsEnding(h)== false) {
         for (int j=0;j<4;j++) {
           Data[i][j]=FileReadNumber(h);
         }
         i++;
       }
// И сортируем массив по нашему критерию оптимизации
       ArraySort(Data,WHOLE_ARRAY,0,MODE_DESCEND);
// Пожалуй немного оформим результат. Для этого придётся переоткрыть файл
       FileClose(h);
       h=FileOpen("test.txt",FILE_CSV|FILE_WRITE,' ');
       FileWrite(h,"  Критерий","     MovingPeriod"," MovingShift"," Счётчик");
       for (i=0;i<TestsNumber;i++) {
         FileWrite(h,DoubleToStr(Data[i][0],10),"        ",Data[i][1],"        ",Data[i][2],"
",Data[i][3]);
       }
// Осталось убрать за собой
       GlobalVariableDel("FilePtr");
/*
// Код независимого счётчика проходов
```

```
      GlobalVariableDel("TestsCnt");
*/
     }
     FileClose(h);
    }
  }
  return(0);
}
//+------------------------------------------------------------------+
//| Calculate open positions                                         |
//+------------------------------------------------------------------+
int CalculateCurrentOrders(string symbol)
  {
   int buys=0,sells=0;
//----
   for(int i=0;i<OrdersTotal();i++)
     {
      if(OrderSelect(i,SELECT_BY_POS,MODE_TRADES)==false) break;
      if(OrderSymbol()==Symbol() && OrderMagicNumber()==MAGICMA)
        {
         if(OrderType()==OP_BUY)  buys++;
         if(OrderType()==OP_SELL) sells++;
        }
     }
//---- return orders volume
   if(buys>0) return(buys);
   else       return(-sells);
  }
//+------------------------------------------------------------------+
//| Calculate optimal lot size                                       |
//+------------------------------------------------------------------+
double LotsOptimized()
  {
   double lot=Lots;
   int    orders=HistoryTotal();     // history orders total
   int    losses=0;                  // number of losses orders without a break
//---- select lot size
   lot=NormalizeDouble(AccountFreeMargin()*MaximumRisk/1000.0,1);
//---- calcuulate number of losses orders without a break
   if(DecreaseFactor>0)
     {
      for(int i=orders-1;i>=0;i--)
        {
         if(OrderSelect(i,SELECT_BY_POS,MODE_HISTORY)==false) { Print("Error in
history!"); break; }
         if(OrderSymbol()!=Symbol() || OrderType()>OP_SELL) continue;
         //----
         if(OrderProfit()>0) break;
         if(OrderProfit()<0) losses++;
        }
      if(losses>1) lot=NormalizeDouble(lot-lot*losses/DecreaseFactor,1);
     }
//---- return lot size
   if(lot<0.1) lot=0.1;
```

```
   return(lot);
  }
//+------------------------------------------------------------------+
//| Check for open order conditions                                  |
//+------------------------------------------------------------------+
void CheckForOpen()
  {
   double ma;
   int    res;
//---- go trading only for first tiks of new bar
   if(Volume[0]>1) return;
//---- get Moving Average
   ma=iMA(NULL,0,MovingPeriod,MovingShift,MODE_SMA,PRICE_CLOSE,0);
//---- sell conditions
   if(Open[1]>ma && Close[1]<ma)
     {
      res=OrderSend(Symbol(),OP_SELL,LotsOptimized(),Bid,3,0,0,"",MAGICMA,0,Red);
      return;
     }
//---- buy conditions
   if(Open[1]<ma && Close[1]>ma)
     {
      res=OrderSend(Symbol(),OP_BUY,LotsOptimized(),Ask,3,0,0,"",MAGICMA,0,Blue);
      return;
     }
//----
  }
//+------------------------------------------------------------------+
//| Check for close order conditions                                 |
//+------------------------------------------------------------------+
void CheckForClose()
  {
   double ma;
//---- go trading only for first tiks of new bar
   if(Volume[0]>1) return;
//---- get Moving Average
   ma=iMA(NULL,0,MovingPeriod,MovingShift,MODE_SMA,PRICE_CLOSE,0);
//----
   for(int i=0;i<OrdersTotal();i++)
     {
      if(OrderSelect(i,SELECT_BY_POS,MODE_TRADES)==false)       break;
      if(OrderMagicNumber()!=MAGICMA || OrderSymbol()!=Symbol()) continue;
      //---- check order type
      if(OrderType()==OP_BUY)
        {
         if(Open[1]>ma && Close[1]<ma) OrderClose(OrderTicket(),OrderLots(),Bid,3,White);
         break;
        }
      if(OrderType()==OP_SELL)
        {
         if(Open[1]<ma && Close[1]>ma) OrderClose(OrderTicket(),OrderLots(),Ask,3,White);
         break;
        }
     }
```

```
//----
  }
//+------------------------------------------------------------------+
//| Start function                                                   |
//+------------------------------------------------------------------+
void start()
  {
  if (IsTesting() && TestsNumber > 0) {
// Для расчёта критерия оптимизации рассчитываем максимум средств и максимальную просадку
    if (AccountEquity() > MaxEqu) MaxEqu = AccountEquity();
    if (MaxEqu-AccountEquity() > MaxDD) MaxDD = MaxEqu-AccountEquity();
  }
//---- check for history and trading
   if(Bars<100 || IsTradeAllowed()==false) return;
//---- calculate open orders by current symbol
   if(CalculateCurrentOrders(Symbol())==0) CheckForOpen();
   else                                    CheckForClose();
//----
  }
//+------------------------------------------------------------------+
```