```
//+------------------------------------------------------------------+
//|                                          C_OPTIMIZATION_LOG.mqh |
//|                                                        M Wilson |
//|                                          https://www.mql5.com |
//+------------------------------------------------------------------+
#property copyright "M Wilson"
#property link      "https://www.mql5.com"
#property version   "1.00"
#property strict
//+------------------------------------------------------------------+
//|The C_OPTIMIZATION_LOG class is used for logging Optimization     |
//|results.                                                          |
//+------------------------------------------------------------------+

//+------------------------------------------------------------------+
//| C_OPTIMIZATION_LOG Class                                         |
//+------------------------------------------------------------------+
const string C_FileName="OptimizationLog.csv";


//+------------------------------------------------------------------+
//| C_OPTIMIZATION_LOG Class                                         |
//+------------------------------------------------------------------+
class C_OPTIMIZATION_LOG
{
private:
   //Private Functions.
   void AppendStringToLog(const string strInput);
   void WriteTitlesToLogFile();
   void UpdateTradeData();
protected:
   //Protected Variables
   int m_intMagicNumber;
   string m_strSymbol;
   datetime m_dtStartDate;
   datetime m_dtEndDate;
   datetime m_dtMidDate;
   datetime m_dtFirstQuarterDate;
   datetime m_dtLastQuarterDate;
   int m_intNoBuy;
   int m_intNoSell;
   double m_dblFirstQuarterPandL;
   double m_dblSecondQuarterPandL;
   double m_dblThirdQuarterPandL;
   double m_dblFourthQuarterPandL;
   double m_dblAtRisk;
   int m_intNoGapRiskLimitedTrades;
public:
   //Public Variables
   string m_strWhereIsTheLog;
   string m_strWhereIsTheStrategyTesterLog;
   //Constructor and Destructor
   C_OPTIMIZATION_LOG();
   C_OPTIMIZATION_LOG(const int intMagicNumber=0, const string strSymbol="");
   ~C_OPTIMIZATION_LOG();
   //Public Functions
   void StartDate(const datetime dtInput){if(this.m_dtStartDate<1)   this.m_dtStartDate=
dtInput;};
   void EndDate(const datetime dtInput){this.m_dtEndDate=dtInput;};
   void AddToAmountAtRisk(const double dblRiskToAdd=0){this.m_dblAtRisk+=dblRiskToAdd;};
   void IncrementNoGAPRiskLimitedTrades(){this.m_intNoGapRiskLimitedTrades++;};
   void PrintLocationOfLogFiles();
   void RemoveLogFile();
   void ArchiveLogFile();
   bool FileExists();
   void WriteSummaryToLogFile(const string strFinalAppend="");
};
//+------------------------------------------------------------------+
//|  Constructor                                                     |
//+------------------------------------------------------------------+
C_OPTIMIZATION_LOG::C_OPTIMIZATION_LOG()
```

```
{
   this.m_strWhereIsTheLog="";
   this.m_strWhereIsTheStrategyTesterLog="";
   this.m_dtStartDate=0;
   this.m_dtEndDate=0;
   this.m_dblAtRisk=0;
   this.m_intNoGapRiskLimitedTrades=0;
}
C_OPTIMIZATION_LOG::C_OPTIMIZATION_LOG(const int intMagicNumber=0, const string
strSymbol="")
{
   //Initiate Variables
   this.m_intMagicNumber=intMagicNumber;
   this.m_strSymbol=strSymbol;


//Update WhereIsTheLog so that the developer can find out where the actual log file should b
   string strTerminalPath = TerminalInfoString(TERMINAL_DATA_PATH);
   this.m_strWhereIsTheLog=strTerminalPath+"\\MQL4\\Files\\"+C_FileName;
   this.m_strWhereIsTheStrategyTesterLog=strTerminalPath+"\\tester\\files\\"+C_FileName;

   //If the file does not exist, create the file and update the titles.
   if(!this.FileExists())
   {
      this.WriteTitlesToLogFile();
   }

}
//+------------------------------------------------------------------+
//|  Destructor                                                      |
//+------------------------------------------------------------------+
C_OPTIMIZATION_LOG::~C_OPTIMIZATION_LOG()
{
}
//+------------------------------------------------------------------+
//+------------------------------------------------------------------+
//|  Public Functions                                                |
//+------------------------------------------------------------------+
void C_OPTIMIZATION_LOG::PrintLocationOfLogFiles()
{

//Call this function at the end of an EA to let the developer/user know where the log file i
   Print(this.m_strWhereIsTheStrategyTesterLog);
   Print("Location of Strategy Tester Log File:");
   Print(this.m_strWhereIsTheLog);
   Print("Location of Standard Log File:");

   return;
}
void C_OPTIMIZATION_LOG::RemoveLogFile()
{
   //This just deletes the log file
   FileDelete(C_FileName);
   return;
}
void C_OPTIMIZATION_LOG::ArchiveLogFile()
{

//If there is not a BUP directory it creates it.   It then deletes any BUP versions of the f
   //and copys the file over it.   A new log file is then created.

//It is best if this is called during the Init stage of an EA so that we have prepaired a ne

   if(!FileIsExist("Bup"))
   {
      FolderCreate("Bup");
   }

   FileDelete("Bup\\"+C_FileName);
   FileCopy(C_FileName,0,"BUP\\"+C_FileName,FILE_REWRITE);
   FileDelete(C_FileName);
```

```
   return;

}
void C_OPTIMIZATION_LOG::WriteTitlesToLogFile()
{
   //Add titles to log file

   //Now Create a string containing the titles of data
   string strTitles=
"MagicNo;Symbol;StartDate;MidDate;EndDate;EndAcctBalance;EndAcctEquity;EndAcctProf;NoBuy;NoS
;
   this.AppendStringToLog(strTitles);

   return;

}
bool C_OPTIMIZATION_LOG::FileExists()
{
   bool boolRet=FileIsExist(C_FileName);
   return boolRet;
}
void C_OPTIMIZATION_LOG::WriteSummaryToLogFile(const string strFinalAppend="")
{
   //Update the trade data before writing the file
   this.UpdateTradeData();

   //Get Basic Log Data
   string strWrite=IntegerToString(this.m_intMagicNumber)+";";
   strWrite+=this.m_strSymbol+";";
   strWrite+=TimeToString(this.m_dtStartDate)+";";
   strWrite+=TimeToString(this.m_dtMidDate)+";";
   strWrite+=TimeToString(this.m_dtEndDate)+";";
   //Get Account Results
   strWrite+=DoubleToString(AccountInfoDouble(ACCOUNT_BALANCE))+";";
   strWrite+=DoubleToString(AccountInfoDouble(ACCOUNT_EQUITY))+";";
   strWrite+=DoubleToString(AccountInfoDouble(ACCOUNT_PROFIT))+";";
   strWrite+=IntegerToString(this.m_intNoBuy)+";";
   strWrite+=IntegerToString(this.m_intNoSell)+";";
   strWrite+=DoubleToString(this.m_dblFirstQuarterPandL)+";";
   strWrite+=DoubleToString(this.m_dblSecondQuarterPandL)+";";
   strWrite+=DoubleToString(this.m_dblThirdQuarterPandL)+";";
   strWrite+=DoubleToString(this.m_dblFourthQuarterPandL)+";";
   strWrite+=DoubleToString(this.m_dblFirstQuarterPandL+this.m_dblSecondQuarterPandL+
this.m_dblThirdQuarterPandL+this.m_dblFourthQuarterPandL)+";";
   strWrite+=DoubleToString(this.m_dblAtRisk)+";";
   strWrite+=DoubleToString((this.m_dblFirstQuarterPandL+this.m_dblSecondQuarterPandL+
this.m_dblThirdQuarterPandL+this.m_dblFourthQuarterPandL)/this.m_dblAtRisk)+";";
   strWrite+=IntegerToString(this.m_intNoGapRiskLimitedTrades)+";";

   //Append the final bit of information
   strWrite+=strFinalAppend;

   this.AppendStringToLog(strWrite);
}
//+------------------------------------------------------------------+
//|  Private Functions                                               |
//+------------------------------------------------------------------+
void C_OPTIMIZATION_LOG::AppendStringToLog(const string strInput)
{

//This function opens the Log File, moves to the end of the file and then appends the input

//string to the file.   Returns (ie \r\n) are added by this routine onto the end of the stri

   //Reset the last error
   ResetLastError();

   //Open the file - must be read and write for appending to files.
   int intFileHandle=FileOpen(C_FileName,FILE_READ|FILE_WRITE|FILE_TXT);

   //If the file has been opened successfully, write to it
```

```
   if(intFileHandle!=INVALID_HANDLE)
   {
      //Find the End of the file
      if(!FileSeek(intFileHandle,0,SEEK_END))   Print(__FUNCTION__,"File Seek Error ",
GetLastError());

      //Write the String
      if(FileWriteString(intFileHandle,strInput+"\r\n")<=0) Print(__FUNCTION__,
"File Write Error ",GetLastError());

      //Close the file
      FileClose(intFileHandle);
   }
   else
   {
      Print(__FUNCTION__,"Failed to open file ",C_FileName," ",GetLastError());
   }

   return;
}
void C_OPTIMIZATION_LOG::UpdateTradeData()
{
   //Set the mid date
   long lngNoSec=(this.m_dtEndDate-this.m_dtStartDate)/2;
   this.m_dtMidDate=(datetime)(this.m_dtStartDate+lngNoSec);
   this.m_dtFirstQuarterDate=(datetime)(this.m_dtStartDate+lngNoSec/2);
   this.m_dtLastQuarterDate=(datetime)(this.m_dtMidDate+lngNoSec/2);

   //Scan through the trades working out the results
   this.m_intNoBuy=0;
   this.m_intNoSell=0;
   this.m_dblFirstQuarterPandL=0;
   this.m_dblSecondQuarterPandL=0;
   this.m_dblThirdQuarterPandL=0;
   this.m_dblFourthQuarterPandL=0;

   int intCount=OrdersHistoryTotal();
   for(int i=0;i<intCount;i++)
   {
      if(OrderSelect(i,SELECT_BY_POS,MODE_HISTORY))
      {
         if(OrderType()==OP_BUY)
         {
            this.m_intNoBuy++;
         }
         else if(OrderType()==OP_SELL)
         {
            this.m_intNoSell++;
         }

         if(OrderOpenTime()<this.m_dtFirstQuarterDate && OrderOpenTime()>=this
.m_dtStartDate)
         {
            this.m_dblFirstQuarterPandL+=(OrderProfit()+OrderCommission()+OrderSwap());
         }
         else if(OrderOpenTime()<this.m_dtMidDate && OrderOpenTime()>=this.m_dtStartDate
)
         {
            this.m_dblSecondQuarterPandL+=(OrderProfit()+OrderCommission()+OrderSwap());
         }
         else if(OrderOpenTime()<this.m_dtLastQuarterDate && OrderOpenTime()>=this
.m_dtStartDate)
         {
            this.m_dblThirdQuarterPandL+=(OrderProfit()+OrderCommission()+OrderSwap());
         }
         else if(OrderOpenTime()<=this.m_dtEndDate && OrderOpenTime()>=this
.m_dtStartDate)
         {
            this.m_dblFourthQuarterPandL+=(OrderProfit()+OrderCommission()+OrderSwap());
         }
      }
```

```
      else
      {
         Print(__FILE__+" : "+__FUNCTION__," Could not select historic trade position ",
i);
      }
   }

}
```