```
//+------------------------------------------------------------------+
//|                                                   C_Calendar.mqh |
//|                                Copyright 2017, M Wilson |
//|                                https://www.algotrader.blog |
//+------------------------------------------------------------------+
#include <C_Calendar_Data.mqh>

#property copyright "Copyright 2017, M Wilson"
#property link      "https://www.algotrader.blog"
#property version   "1.00"
#property strict

//+------------------------------------------------------------------+
//|                                                                  |
//+------------------------------------------------------------------+
class C_Calendar
  {
public:
   C_Calendar();
  ~C_Calendar();
  int ArraySizeCalendar();
  int ArraySizeBankHols();
  int ArraySizeAllDayEvents();
  void AddCalendarData(const C_Calendar_Data &srcC_Calendar_Data);
  void AddBankHoliday(const C_Calendar_Data &srcC_Calendar_Data);
  void AddAllDayEvent(const C_Calendar_Data &srcC_Calendar_Data);
  C_Calendar_Data *GetCalendarData(const int intIndex);
  bool Exists(const C_Calendar_Data &srcC_Calendar_Data);
  bool HighestImpactEventBetweenDates(const datetime dtStart, const datetime dtEnd,
NewsImpact &eImpactLevel);
  bool HighestImpactAllDayEvent(const datetime dtToday,NewsImpact &eImpactLevel);
  bool TodayIsBankHoliday(const datetime dtNow, string strCurrency, string strCountry);
  bool TodayIsBankHoliday(const datetime dtNow,string &strCurrency[]);
  string AsString();
protected:
   void ClearArray();
private:
   C_Calendar_Data *m_objC_Calendar_Data[];
   C_Calendar_Data *m_objC_BankHolidays[];
   C_Calendar_Data *m_objC_AllDayEvents[];
  };
//+------------------------------------------------------------------+
//| Public Constructor/Destructor                                    |
//+------------------------------------------------------------------+
C_Calendar::C_Calendar()
  {
  }
C_Calendar::~C_Calendar()
  {
  this.ClearArray();
  }
//+------------------------------------------------------------------+
//+------------------------------------------------------------------+
//| Public Functions                                                 |
//+------------------------------------------------------------------+
int C_Calendar::ArraySizeCalendar()
{
   int intRet=ArraySize(this.m_objC_Calendar_Data);
   return intRet;
}
int C_Calendar::ArraySizeBankHols()
{
   int intRet=ArraySize(this.m_objC_BankHolidays);
   return intRet;
}
int C_Calendar::ArraySizeAllDayEvents()
{
   int intRet=ArraySize(this.m_objC_AllDayEvents);
   return intRet;
}
void C_Calendar::AddCalendarData(const C_Calendar_Data &srcC_Calendar_Data)
```

```
{
   int intCount=this.ArraySizeCalendar();
   ArrayResize(this.m_objC_Calendar_Data,intCount+1);
   this.m_objC_Calendar_Data[intCount]=new C_Calendar_Data(srcC_Calendar_Data);
}
void C_Calendar::AddBankHoliday(const C_Calendar_Data &srcC_Calendar_Data)
{
   int intCount=this.ArraySizeBankHols();
   ArrayResize(this.m_objC_BankHolidays,intCount+1);
   this.m_objC_BankHolidays[intCount]=new C_Calendar_Data(srcC_Calendar_Data);
}
void C_Calendar::AddAllDayEvent(const C_Calendar_Data &srcC_Calendar_Data)
{
   int intCount=this.ArraySizeAllDayEvents();
   ArrayResize(this.m_objC_AllDayEvents,intCount+1);
   this.m_objC_AllDayEvents[intCount]=new C_Calendar_Data(srcC_Calendar_Data);
}
C_Calendar_Data *C_Calendar::GetCalendarData(const int intIndex)
{
   if(intIndex<0 || intIndex>=this.ArraySizeCalendar())
   {
      return NULL;
   }
   else
   {
      return  this.m_objC_Calendar_Data[intIndex];
   }
}
bool C_Calendar::Exists(const C_Calendar_Data &srcC_Calendar_Data)
{
   bool boolRet=False;
   for(int i=0;i<this.ArraySizeCalendar();i++)
   {
      if(this.m_objC_Calendar_Data[i].IsEqual(srcC_Calendar_Data))
      {
         boolRet=True;
         break;
      }
   }
   return boolRet;
}
bool C_Calendar::HighestImpactEventBetweenDates(const datetime dtStart, const datetime
dtEnd, NewsImpact &eImpactLevel)
{

//Function returns True if an event is between dtStart and dtEnd.   It then populates the hi

//Assume that the enum NewsImpact has the lowest integer for LowImpact and highest integer f
   bool boolRet=False,boolFound=False;
   int intCount=this.ArraySizeCalendar();
   for(int i=0;i<intCount;i++)
   {
      if(this.m_objC_Calendar_Data[i].m_dtTime>=dtStart && this.m_objC_Calendar_Data[i]
.m_dtTime<dtEnd)
      {
         boolRet=True;
         if(!boolFound)
         {
            eImpactLevel=this.m_objC_Calendar_Data[i].m_eImpact;
            boolFound=True;
         }
         else if(this.m_objC_Calendar_Data[i].m_eImpact>eImpactLevel)
         {
            eImpactLevel=this.m_objC_Calendar_Data[i].m_eImpact;
         }
      }
   }
   return boolRet;
}
bool C_Calendar::HighestImpactAllDayEvent(const datetime dtToday,NewsImpact &
eImpactLevel)
```

```
{

//Function returns True if an all day event exists for dtToday.   It then populates the high

//Assume that the enum NewsImpact has the lowest integer for LowImpact and highest integer f
   bool boolRet=False,boolFound=False;
   int intCount=this.ArraySizeAllDayEvents();
   for(int i=0;i<intCount;i++)
   {
      datetime dtEvent=this.m_objC_AllDayEvents[i].m_dtTime;
      if(TimeYear(dtEvent)==TimeYear(dtToday) && TimeMonth(dtEvent)==TimeMonth(dtToday)
&& TimeDay(dtEvent)==TimeDay(dtToday))
      {
         boolRet=True;
         if(!boolFound)
         {
            eImpactLevel=this.m_objC_AllDayEvents[i].m_eImpact;
            boolFound=True;
         }
         else if(this.m_objC_AllDayEvents[i].m_eImpact>eImpactLevel)
         {
            eImpactLevel=this.m_objC_AllDayEvents[i].m_eImpact;
         }
      }
   }
   return boolRet;
}
bool C_Calendar::TodayIsBankHoliday(const datetime dtNow,string strCurrency, string
strCountry)
{

//This routine is a bit simplistic.   It loos for a calendar event that has been loaded into

//as the input date dtNow.   Of course, the US days start at a different time as UK days, so
   //so I will keep the comparison simple.

//You can specify a currency to search for and in the case of EUR, a country string.   If th

   //Get ucase, trim of country and currency
   string strCurrTUP=StringTrimLeft(StringTrimRight(strCurrency));
   string strCountryTUP=StringTrimLeft(StringTrimRight(strCountry));
   StringToUpper(strCurrTUP);
   StringToUpper(strCountryTUP);

   bool boolRet=False;
   int intCount=this.ArraySizeBankHols();
   for(int i=0;i<intCount;i++)
   {
      //If we find a bank holiday record with the same day as today, then return true
      datetime dtBH=this.m_objC_BankHolidays[i].m_dtTime;
      if(TimeDay(dtBH)==TimeDay(dtNow) && TimeMonth(dtBH)==TimeMonth(dtNow) && TimeYear(
dtBH)==TimeYear(dtNow))
      {
         bool boolCurrencyOK=True, boolCountryOK=True;


//If an input currency has been provided, do a string comparison on the country.
         if(StringLen(strCurrTUP)>0)
         {
            string strC=StringTrimLeft(StringTrimRight(this.m_objC_BankHolidays[i]
.m_strCountry));
            StringToUpper(strC);
            if(strCurrTUP!=strC)
            {  //Currency is different to input, do not process.
               boolCurrencyOK=False;
            }
            else if(StringLen(strCountryTUP)>0)
            {

//Currency is same and a country has been provided in the input.   See if we can find the co
```

```
//title string of the event, eg the currency is EUR and we are looking for 'French' (strCour
                string strCO=StringTrimLeft(StringTrimRight(this.m_objC_BankHolidays[i]
.m_strTitle));
                StringToUpper(strCO);
                if(StringFind(strCO,strCountryTUP,0)<0)
                {
                   boolCountryOK=False;
                }
             }
          }

          if(boolCurrencyOK && boolCountryOK)
          {
             boolRet=True;
             break;
          }
       }
    }
    return boolRet;
}
bool C_Calendar::TodayIsBankHoliday(const datetime dtNow,string &strCurrency[])
{

//This routine is a bit simplistic.   It loos for a calendar event that has been loaded intc

//as the input date dtNow.   Of course, the US days start at a different time as UK days, sc
    //so I will keep the comparison simple.
    //You can specify a currency array to search for.

    bool boolRet=False;
    int intCount=this.ArraySizeBankHols();
    for(int i=0;i<intCount;i++)
    {
       //If we find a bank holiday record with the same day as today, then return true
       datetime dtBH=this.m_objC_BankHolidays[i].m_dtTime;
       if(TimeDay(dtBH)==TimeDay(dtNow) && TimeMonth(dtBH)==TimeMonth(dtNow) && TimeYear(
dtBH)==TimeYear(dtNow))
       {
          bool boolCurrencyOK=False;
          for(int intC=0;intC<ArraySize(strCurrency);intC++)
          {
             //Get ucase, trim of currency from currency array
             string strCurrTUP=StringTrimLeft(StringTrimRight(strCurrency[intC]));
             StringToUpper(strCurrTUP);
             //Get ucase trim of currency from event.
             string strC=StringTrimLeft(StringTrimRight(this.m_objC_BankHolidays[i]
.m_strCountry));
             StringToUpper(strC);

//See if the event country equals one of the currencies in the input array, if they are the
             if(strC==strCurrTUP)
             {
                boolRet=True;
                break;
             }
          }
       }
    }
    return boolRet;
}
string C_Calendar::AsString()
{
    string strRet="Calendar:\n";
    int intCount=this.ArraySizeCalendar();
    for(int i=0;i<intCount;i++)
    {
       strRet+=this.m_objC_Calendar_Data[i].AsString()+"\n";
    }
    strRet+="\nBank Hols:\n";
    intCount=this.ArraySizeBankHols();
```

```
   for(int i=0;i<intCount;i++)
   {
      strRet+=this.m_objC_BankHolidays[i].AsString()+"\n";
   }
   strRet+="\nAll Day Events:\n";
   intCount=this.ArraySizeAllDayEvents();
   for(int i=0;i<intCount;i++)
   {
      strRet+=this.m_objC_AllDayEvents[i].AsString()+"\n";
   }
   return strRet;
}
//+------------------------------------------------------------------+
//| Protected Functions                                              |
//+------------------------------------------------------------------+
void C_Calendar::ClearArray()
{
   int intCount=this.ArraySizeCalendar();
   for(int i=0;i<intCount;i++)
   {
      delete this.m_objC_Calendar_Data[i];
   }
   ArrayFree(this.m_objC_Calendar_Data);
   intCount=this.ArraySizeBankHols();
   for(int i=0;i<intCount;i++)
   {
      delete this.m_objC_BankHolidays[i];
   }
   ArrayFree(this.m_objC_BankHolidays);
   intCount=this.ArraySizeAllDayEvents();
   for(int i=0;i<intCount;i++)
   {
      delete this.m_objC_AllDayEvents[i];
   }
   ArrayFree(this.m_objC_AllDayEvents);
}
//+------------------------------------------------------------------+
//| Private Functions                                                |
//+------------------------------------------------------------------+
```