ADT Time

Header file

```
#ifndef   TIMETYPE_H
#define   TIMETYPE_H

class TimeType
{
public:
   TimeType();
      // Postcondition:  Class object is constructed  &&  Time is 0:0:0

   TimeType( /* in */ int initHrs,
             /* in */ int initMins,
             /* in */ int initSecs );
      // Precondition:
      //     0 <= initHrs <= 23  &&  0 <= initMins <= 59  && 0 <= initSecs <= 59
      // Postcondition:
      //     Class object is constructed. Time is set according to the incoming parameters

   void Set( /* in */ int hours,
             /* in */ int minutes,
             /* in */ int seconds );
      // Precondition:
      //     0 <= hours <= 23  &&  0 <= minutes <= 59  && 0 <= seconds <= 59
      // Postcondition:
      //     Time is set according to the incoming parameters

   void Increment();
      // Postcondition:
      //     Time has been advanced by one second, with  23:59:59 wrapping around to 0:0:0

   void Write() const;
      // Postcondition:
      //     Time has been output in the form HH:MM:SS

   bool Equal( /* in */ TimeType otherTime ) const;
      // Postcondition:
      //     Function value == true,
      //   if this time equals otherTime;    == false, otherwise

   bool LessThan( /* in */ TimeType otherTime ) const;
      // Precondition:
      //     This time and otherTime represent times in the same day
      // Postcondition:
      //     Function value == true,
     // if this time is earlier in the day than otherTime;  == false, otherwise
private:
   int hrs;
   int mins;
   int secs;
};
#endif
```

Implementation file

```
#include "timetype.h"
#include <iostream>
using namespace std;

TimeType::TimeType()
{
   hrs = 0;
   mins = 0;
   secs = 0;
}

TimeType::TimeType( /* in */ int initHrs,
                    /* in */ int initMins,
                    /* in */ int initSecs )
{
   hrs = initHrs;
   mins = initMins;
   secs = initSecs;
}

void TimeType::Set( /* in */ int hours,
                    /* in */ int minutes,
                    /* in */ int seconds )
{
   hrs = hours;
   mins = minutes;
   secs = seconds;
}

void TimeType::Increment()
{
   secs++;
   if (secs > 59)
   {
      secs = 0;
      mins++;
      if (mins > 59)
      {
         mins = 0;
         hrs++;
         if (hrs > 23)
            hrs = 0;
      }
   }
}

void TimeType::Write() const
{
   if (hrs < 10)
      cout << '0';
```

```
    cout << hrs << ':';
    if (mins < 10)
        cout << '0';
    cout << mins << ':';
    if (secs < 10)
        cout << '0';
    cout << secs;
}


//************************************************
bool TimeType::Equal( /* in */ TimeType otherTime ) const

{
    return (hrs == otherTime.hrs && mins == otherTime.mins &&
            secs == otherTime.secs);
}

//************************************************
bool TimeType::LessThan( /* in */ TimeType otherTime ) const
{
    return (hrs < otherTime.hrs ||
            hrs == otherTime.hrs && mins < otherTime.mins ||
            hrs == otherTime.hrs && mins == otherTime.mins
                        && secs < otherTime.secs);
}
```

Client Program

```
#include "timetype.h"
#include <iostream>
using namespace std;

int main()
{
    TimeType time1(5, 30, 0);
    TimeType time2;
    int      count;

    time2 = time1;
    cout << "time1: ";
    time1.Write();
    cout << "  time2: ";
    time2.Write();
    cout << endl;

    if (time1.Equal(time2))
        cout << "Times are equal" << endl;
    else
        cout << "Times are NOT equal" << endl;

    time2.Increment();
    cout << "New time2: ";
```

```
    time2.Write();
    cout << endl;

    if (time1.Equal(time2))
       cout << "Times are equal" << endl;
    else
       cout << "Times are NOT equal" << endl;

    if (time1.LessThan(time2))
       cout << "time1 is less than time2" << endl;
    else
       cout << "time1 is NOT less than time2" << endl;

    if (time2.LessThan(time1))
       cout << "time2 is less than time1" << endl;
    else
       cout << "time2 is NOT less than time1" << endl;

    cout << "Incrementing time1:" << endl;
    time1.Set(23, 59, 55);
    for (count = 1; count <= 10; count++)
    {
       time1.Write();
       cout << ' ';
       time1.Increment();
    }
    return 0;
}
```