**C/C++ ADT Class Digital Time LAB REPORT**

**1) Enter your name, student ID, platform (Mac or PC) and date**

Name and Student ID: Samuel Indurkar, 0888068

Class: CIS054 C/C++ Programming

Platform (Mac or PC): MAC and Eclipse  
Date: 7/23/2017

**DESCRIPTION:**

Update the program that is already provided for the Digital Time class to add a new member function that computes the difference in time between two different DigitalTime objects.

Three files are already provided for you on the class website:

dtime.h a header file that defines the digital time class  
 dtime.cpp the implementation code for the digital time class

C++DigitalTimeApplication.cpp sample code that tests the digital time class

**LAB ASSIGNMENT:**2) Refer to practice problem #1 (8th Edition page 718 , 9th Edition page 736) for additional information.   
 *void* DigitalTime::interval\_since(*const* DigitalTime& a\_previous\_time,  
 *int*& hours\_in\_inteval, *int*& minutes\_in\_interval) *const*

where: hours\_in\_interval and minutes\_in\_interval are values returned using PassByReference.

Refer to  **C++DigitalTimeApplication.cpp** to see a sample run of the program. The project is not complete until the code for the **interval\_since** member function has been written. Add the the **interval\_since** member function and test it a) when the first argument is earlier in the day than the second argument and b) when the first argument is later in the day than the first argument.

**DISCUSSION:**

**3) Complete the DISCUSSION section. It does not need to be long, but it needs to be complete.**3a) What did you do to develop the program? ("Followed the Directions" is not a complete description)

I added the interval\_since() member function which computes the time difference between two objects. First, I converted the time into minutes. The entire time is expressed as minutes. Then I compare the two times( expressed in minutes). For the first simple case, I just subtract the two times and then I re-convert the difference-in-minutes back to hours-and-minutes

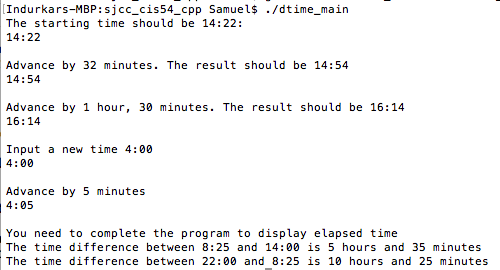
3b) What problems did you have and how did you overcome the problems?

I had the “previous\_time” flipped so it was displaying negative numbers. But I put some debug prints and figured it out.

**PROGRAM OUTPUT:**

**4) Show screen shots for** a) when the first argument is earlier in the day than the second argument and b) when the first argument is later in the day than the first argument.

Refer to previous lab assignments for instructions on how to capture a screen or portions of a screen for either the PC or a Mac

****

**PROGRAM LISTING:**

**5) Copy and paste the code that YOU typed to make the program work. Your program should include a comment block at the top that shows the name of the program, date, version and your name.**

/\*

\* dtime.h

\*

\* Created on: Jul 23, 2017

\* Author: Samuel

\*/

#ifndef DTIME\_H\_

#define DTIME\_H\_

#pragma once

//DISPLAY 12.2 Implementation File for DigitalTime

//Implementation file dtime.cpp (Your system may require some

//suffix other than .cpp): This is the IMPLEMENTATION of the ADT DigitalTime.

//The interface for the class DigitalTime is in the header file dtime.h.

//DISPLAY 12.1 Interface File for DigitalTime

//Header file dtime.h: This is the INTERFACE for the class DigitalTime.

//Values of this type are times of day. The values are input and output in

//24-hour notation, as in 9:30 for 9:30 AM and 14:45 for 2:45 PM.

#include <iostream>

using namespace std;

class DigitalTime

{

public:

friend bool operator ==(const DigitalTime& time1, const DigitalTime& time2);

//Returns true if time1 and time2 represent the same time;

//otherwise, returns false.

DigitalTime(int the\_hour, int the\_minute);

//Precondition: 0 <= the\_hour <= 23 and 0 <= the\_minute <= 59.

//Initializes the time value to the\_hour and the\_minute.

DigitalTime( );

//Initializes the time value to 0:00 (which is midnight).

void advance(int minutes\_added);

//Precondition: The object has a time value.

//Postcondition: The time has been changed to minutes\_added minutes later.

void advance(int hours\_added, int minutes\_added);

//Precondition: The object has a time value.

//Postcondition: The time value has been advanced

//hours\_added hours plus minutes\_added minutes.

friend istream& operator >>(istream& ins, DigitalTime& the\_object);

//Overloads the >> operator for input values of type DigitalTime.

//Precondition: If ins is a file input stream, then ins has already been

//connected to a file.

friend ostream& operator <<(ostream& outs, const DigitalTime& the\_object);

//Overloads the << operator for output values of type DigitalTime.

//Precondition: If outs is a file output stream, then outs has already been

//connected to a file.

void interval\_since(const DigitalTime& previous\_time,

int& hours\_in\_interval, int& minutes\_in\_interval) const;

//Precondition: The object has a time value.

//Precondition: The previous\_time object has a time value

//Postcondition: The hours\_in\_interval indicates the number of hours that have passed

//Postcondition: The minutes\_in\_interval indicates the number of minutes that have passed

private:

int hour;

int minute;

};

#endif /\* DTIME\_H\_ \*/

/\*

\* dtime.cpp

\*

\* Created on: Jul 23, 2017

\* Author: Samuel

\*/

// This is the IMPLEMENTATION of the ADT DigitalTime.

//The interface for the class DigitalTime is in the header file dtime.h.

#include <iostream>

#include <cctype>

#include <cstdlib>

#include "dtime.h"

using namespace std;

//These FUNCTION DECLARATIONS are for use in the definition of

//the overloaded input operator >>:

void read\_hour(istream& ins, int& the\_hour);

//Precondition: Next input in the stream ins is a time in 24-hour notation,

//like 9:45 or 14:45.

//Postcondition: the\_hour has been set to the hour part of the time.

//The colon has been discarded and the next input to be read is the minute.

void read\_minute(istream& ins, int& the\_minute);

//Reads the minute from the stream ins after read\_hour has read the hour.

int digit\_to\_int(char c);

//Precondition: c is one of the digits '0' through '9'.

//Returns the integer for the digit; for example, digit\_to\_int('3') returns 3.

bool operator ==(const DigitalTime& time1, const DigitalTime& time2)

{

return (time1.hour == time2.hour && time1.minute == time2.minute);

}

//Uses iostream and cstdlib:

DigitalTime::DigitalTime(int the\_hour, int the\_minute)

{

if (the\_hour < 0 || the\_hour > 23 || the\_minute < 0 || the\_minute > 59)

{

cout << "Illegal argument to DigitalTime constructor.";

exit(1);

}

else

{

hour = the\_hour;

minute = the\_minute;

}

}

DigitalTime::DigitalTime( ) : hour(0), minute(0)

{

//Body intentionally empty.

}

void DigitalTime::advance(int minutes\_added)

{

int gross\_minutes = minute + minutes\_added;

minute = gross\_minutes%60;

int hour\_adjustment = gross\_minutes/60;

hour = (hour + hour\_adjustment)%24;

}

void DigitalTime::advance(int hours\_added, int minutes\_added)

{

hour = (hour + hours\_added)%24;

advance(minutes\_added);

}

//Uses iostream:

ostream& operator <<(ostream& outs, const DigitalTime& the\_object)

{

outs << the\_object.hour << ':';

if (the\_object.minute < 10)

outs << '0';

outs << the\_object.minute;

return outs;

}

//Uses iostream:

istream& operator >>(istream& ins, DigitalTime& the\_object)

{

read\_hour(ins, the\_object.hour);

read\_minute(ins, the\_object.minute);

return ins;

}

int digit\_to\_int(char c)

{

return ( static\_cast<int>(c) - static\_cast<int>('0') );

}

//Uses iostream, cctype, and cstdlib:

void read\_minute(istream& ins, int& the\_minute)

{

char c1, c2;

ins >> c1 >> c2;

if (!(isdigit(c1) && isdigit(c2)))

{

cout << "Error illegal input to read\_minute\n";

exit(1);

}

the\_minute = digit\_to\_int(c1)\*10 + digit\_to\_int(c2);

if (the\_minute < 0 || the\_minute > 59)

{

cout << "Error illegal input to read\_minute\n";

exit(1);

}

}

//Uses iostream, cctype, and cstdlib:

void read\_hour(istream& ins, int& the\_hour)

{

char c1, c2;

ins >> c1 >> c2;

if ( !( isdigit(c1) && (isdigit(c2) || c2 == ':' ) ) )

{

cout << "Error illegal input to read\_hour\n";

exit(1);

}

if (isdigit(c1) && c2 == ':')

{

the\_hour = digit\_to\_int(c1);

}

else //(isdigit(c1) && isdigit(c2))

{

the\_hour = digit\_to\_int(c1)\*10 + digit\_to\_int(c2);

ins >> c2;//discard ':'

if (c2 != ':')

{

cout << "Error illegal input to read\_hour\n";

exit(1);

}

}

if ( the\_hour < 0 || the\_hour > 23 )

{

cout << "Error illegal input to read\_hour\n";

exit(1);

}

}

void DigitalTime::interval\_since(const DigitalTime& previous\_time,

int& hours\_in\_interval, int& minutes\_in\_interval) const

{

// you need to complete this part of the program to compute the

// a) hours\_in\_interval, and

// b) minutes\_in\_interval

int temp\_minutes;

int current\_time\_in\_min;

int prev\_time\_in\_min;

prev\_time\_in\_min = (previous\_time.hour \*60) + previous\_time.minute;

//cout << "Prev inn min " << prev\_time\_in\_min << endl;

current\_time\_in\_min = (hour \*60) + minute;

//cout << "current in min" << current\_time\_in\_min<<endl;

if (prev\_time\_in\_min < current\_time\_in\_min)

{

temp\_minutes = current\_time\_in\_min - prev\_time\_in\_min;

//cout << temp\_minutes;

hours\_in\_interval = temp\_minutes/60;

minutes\_in\_interval = temp\_minutes%60;

}

else

{

temp\_minutes = 1440 - prev\_time\_in\_min;

temp\_minutes += current\_time\_in\_min;

hours\_in\_interval = temp\_minutes/60;

minutes\_in\_interval = temp\_minutes%60;

}

return;

}

/\*

\* dtime\_main.cpp

\*

\* Created on: Jul 23, 2017

\* Author: Samuel

\*/

#include "dtime.h"

#include <iostream>

using namespace std;

int main(int argc, char\* argv[])

{

// create a DigitalTime object and name it time1

DigitalTime time1(14,22); // initialize time to 14:22 (2:44pm)

cout << "The starting time should be 14:22:" << endl;

cout << time1 << endl << endl; // display the time

cout << "Advance by 32 minutes. The result should be 14:54" << endl;

time1.advance(32); // add 32 minutes. Should be 14:54

cout << time1 << endl << endl; // display the time

cout << "Advance by 1 hour, 30 minutes. The result should be 16:14" << endl;

time1.advance(1,20); // add 1 hour, 30 minutes. Should be 16:14

cout << time1 << endl << endl; // display the time

cout << "Input a new time ";

cin >> time1;

cout << time1 << endl << endl; // display the time

cout << "Advance by 5 minutes" << endl;

time1.advance(5); // advance 5 minutes

cout << time1 << endl << endl; // display the time

//

// You need to complete the member function interval\_since before the project is finished

//

cout << "You need to complete the program to display elapsed time" << endl;

int hours, minutes;

DigitalTime time2(8, 25);

DigitalTime time3(14, 0);

DigitalTime time4(22, 0);

// the time difference between 8:25 and 14:00 (2:00pm) should be 5 hours, 35 minutes

time3.interval\_since(time2, hours, minutes);

cout << "The time difference between " << time2 << " and " << time3 << " is "

<< hours << " hours and " << minutes << " minutes" << endl;

// the time difference between 22:00(10pm) and 8:25 is 10 hours 25 minutes

time2.interval\_since(time4, hours, minutes);

cout << "The time difference between " << time4 << " and " << time2 << " is "

<< hours << " hours and " << minutes << " minutes" << endl;

}