CSCI-1200 Data Structures — Spring 2014
Lab 2 — C++ Classes

In this lab, we will implement a simple C++ class named `Time`. It represents all possible times in a 24-hour period, including hours, minutes and seconds. An immediate representation issue is how to handle morning (am) and afternoon (pm) times. We could have a separate `bool` indicating whether the time is am or pm. It is easier, however, to represent the hours in *military time*. This means that the hours of the day are numbered from 0 to 23, with 13 being 1 pm, 14 being 2 pm, etc.

Your notes from Lecture 3 with the example `Date` class will be helpful in completing this lab.

Checkpoint 1

In the first checkpoint you will get started by implementing the initial class design, several member functions, and a simple main program to test your class. The instructions below describe how to build your executable using from the command line using g++ (using Cygwin or another UNIX-like terminal) and using Visual Studio. Even if you plan to use Visual Studio or another IDE for the bulk of your work this semester, you are required to also show that you can successfully build and run this lab using g++ on your own machine.

- If you are working in Visual Studio, create a new "Empty" project. You will create three files within the project. These files will be called `time.h`, `time.cpp` and `main.cpp`. Note that in C++ the name of the header and implementation file are not required to exactly match the name of the class, but it is good coding style to do so.

- Begin work on `time.h`. In Visual Studio, create this file by selecting the "Add New Item..." option in the Project Menu (or use Ctrl+Shift+A). Select which type of new file you want to add (in this case a Header file) and give it the name `Time`. You do not need to include the extension (in this case .h) in the name because it will be added for you. Adding the extension though will have the same effect. (Aside: when there is more than one project in the Solution you need to make sure that the correct project is highlighted in the Solution Explorer before carrying out the above.) Now click OK.

  Within the file, declare a class called `Time`. Follow the form and syntax of the `Date` class from Lecture 3. Read the syntax carefully (such as the semi-colon at the end of the class declaration). Add private member variables for the `hour`, `minute` and `second`. In the public area of the class, declare two constructors: one, the default constructor, should initialize each of the member variables to 0; the other, having three arguments, accepts initial values for the hour, minute and second as function call arguments. Declare member functions to access the values of the hour, the minute and the second (three different member functions). It will be crucial for Checkpoint 3 to make these `const`. (Recall: a **const** member function can not change the member variables.)

  Don't write the body of all of the functions in the `time.h` file. Save the implementation for the `time.cpp` file.

- Switch to working on `main.cpp`. Create a new file within your project, as above. Be sure to add code to `#include "time.h"` in addition to including `#include <iostream>`. (Note: We use angle brackets for standard library includes and double quotes for our custom header files in the working directory.) Have the main program create two `Time` objects, one using each constructor. Show use of the functions that access the values of hour, minute and second by printing the two times.

  Note: There is a common confusion when creating a new variable using the default constructor:

```
Time t1(5,30,59);  // calls the non-default constructor w/ 3 integer arguments
Time t2();         // COMPILE ERROR - a buggy attempt to call the default constuctor
Time t3;           // the *correct* way to call the default constructor
```

- Now create and implement the file `time.cpp`. Don't forget to add the line to `#include "time.h"`. Any file that uses or implements `Time` functionality must include the `Time` header file. Implement the constructors and member functions.

- Now, compile your program and remove errors. Here's where the difference between compiling and linking matters.

  When **compiling using g++** on the command line, the two separate command lines:

  ```
  g++ -c main.cpp
  g++ -c time.cpp
  ```

  compile the source code to create two object code files called `main.o` and `time.o` separately. The `-c` means "compile only". Compiler errors will appear at this point. If there are errors in `main.cpp` (or `time.cpp`), then the files `main.o` (or `time.o`) will not be created. Note that we only compile `.cpp` files. We do not directly compile header files. Header files are compiled only indirectly when included in a `.cpp` file. Once you have driven out all of the compiler errors, you can "link" the program using the command:

  ```
  g++ main.o time.o -o time_test
  ```

  to create the executable called `time_test`. If you have not defined all of the necessary member functions in the `Time` class, then you would see "linking" errors at this point. You can combine all three command lines (compiling each of the 2 `.cpp` files to 2 object files and linking all object files) with this command:

  ```
  g++ main.cpp time.cpp -o time_test
  ```

  Or, equivalently, if those are the only two `.cpp` files in the current directory, you can compile and link using the command line wildcard:

  ```
  g++ *.cpp -o time_test
  ```

  Note that this will not create the intermediate .o files and will only proceed to the linking step if the two files compile cleanly.

  In **Visual Studio**, you can compile each of the two .cpp files individually (the .h file isn't compiled separately) by typing Ctrl-F7 when the pane containing that file is active. This will allow you to see and remove compiler errors for each file individually. Note that errors caused by the code in the .h file, `time.h` will appear when compiling either .cpp file. You can also compile and link multiple files at once using Build/Build project name. You will have to do this even after compiling each .cpp file individually because the Ctrl-F7 command does not link to create an executable program.

To complete this checkpoint: Show compilation of the program **using g++**, with all compiler errors removed and demonstrate correct execution of your program. *Yes, please show us you can compile from the terminal with g++, even if you plan to primarily use Visual Studio or another IDE for the rest of the semester.*

Checkpoint 2

Create and test a few more member functions. This will require modifications to all three of your files.

- `setHour`, `setMinute`, `setSecond`. Each should take a single integer argument and change the appropriate member variable. For now, do not worry about illegal values of these variables (such as setting the hour to 25 or the minute to -15). Assume whoever calls the functions does the right thing. In general, this is a bad assumption, but we will not worry about it here.

- `PrintAmPm` prints time in terms of am or pm, so that 13:24:39 would be output as 1:24:39 pm. This member function should have no arguments. Note that this requires some care so that 5 minutes and 4 seconds after 2 in the afternoon is output as 2:05:04 pm. The output should be to `cout`.

To complete this checkpoint: Show a TA your tested and debugged extensions.