Fundamental Computer Programming - C++ Lab(II)

Lab 2: Basics of Class

03/04/2021

Rung-Bin Lin

International Bachelor Program in Informatics
Yuan Ze University

Purposes of this Lab

- **Get you familiar with the basics of class**
  - ✓ **Concept of object-oriented programming**
  - ✓ **Class definition**
    - ➢ **Class declaration**
    - ➢ **Class implementation**
  - ✓ **Object instantiation**
  - ✓ **Constructor**
  - ✓ **Class Scope**

Time Class Definition

- **Class declaration means to create a class type that can be used to create a class object.**

```
// Fig. 9.1: fig09_01.cpp
// Time class.
#include <iostream>
#include <iomanip>
using namespace std;

// Time class declaration
class Time
{
public:
   Time(); // constructor, the name must be the same as class name
   void setTime( int, int, int ); // set hour, minute and second
   void printUniversal(); // print time in universal-time format
   void printStandard(); // print time in standard-time format
private:
   int hour; // 0 - 23 (24-hour clock format)
   int minute; // 0 - 59
   int second; // 0 - 59
}; // end class Time
```

- You may notice that printUniversal() and printStandard() do not have any parameters. So, which object's time is printed?
- The public functions as a whole are also called public interface of the class.

Class Implementation for Member Functions

```
// Time constructor initializes each data member to zero.
// Ensures all Time objects start in a consistent state.
Time::Time() {  // :: is called scope operator
   hour = minute = second = 0;
} // end Time constructor

// set new Time value using universal time; ensure that
// the data remains consistent by setting invalid values to zero
void Time::setTime( int h, int m, int s ) {
   hour = ( h >= 0 && h < 24 ) ? h : 0; // validate hour
   minute = ( m >= 0 && m < 60 ) ? m : 0; // validate minute
   second = ( s >= 0 && s < 60 ) ? s : 0; // validate second
} // end function setTime

// print Time in universal-time format (HH:MM:SS)
void Time::printUniversal() {
   cout << setfill( '0' ) << setw( 2 ) << hour << ":"  << setw( 2 ) << minute << ":" << setw( 2 ) << second;
} // end function printUniversal

// print Time in standard-time format (HH:MM:SS AM or PM)
void Time::printStandard()
   cout << ( ( hour == 0 || hour == 12 ) ? 12 : hour % 12 ) << ":“ << setfill( '0' ) << setw( 2 )
   << minute << ":" << setw( 2 ) << second << ( hour < 12 ? " AM" : " PM" );
} // end function printStandard
```

main() with Object Instantiation (Creation)

```
int main()
{
   Time t; // instantiate object t of class Time

   // output Time object t's initial values
   cout << "The initial universal time is ";
   t.printUniversal(); // 00:00:00
   cout << "\nThe initial standard time is ";
   t.printStandard(); // 12:00:00 AM
   t.setTime( 13, 27, 6 ); // change time
   // output Time object t's new values
   cout << "\n\nUniversal time after setTime is ";
   t.printUniversal(); // 13:27:06
   cout << "\nStandard time after setTime is ";
   t.printStandard(); // 1:27:06 PM
   t.setTime( 99, 99, 99 ); // attempt invalid settings
   // output t's values after specifying invalid values
   cout << "\n\nAfter attempting invalid settings:“ << "\nUniversal time: ";
   t.printUniversal(); // 00:00:00
   cout << "\nStandard time: ";
   t.printStandard(); // 12:00:00 AM
   cout << endl;
} // end main
```

- A client of a class is a program that uses the class in the program.

Object Creation and Object's Handles

- By declaration

  Time t;
- By new

  Time *tPtr;

  tPtr = new Time;
- Object's handles: used to get access to object's members
  - **Name of an object**

    t.setTime(10, 10, 10);
  - **Pointer to an object**

    tPtr→setTime(10,10,10);
  - **Reference to an object**

    Time &tRef = t;

    tRef.setTime(10, 10, 10);
- Member selection operators
  - . and →

Member Function to Set an Object's Time to Another Object's Time

- A member declaration setT1toT2(Time t2) to Time class
- Implement setT1toT2(Time t2)

```
void Time::setT1toT2(Time t2){
    hour = t2.hour;
    minute = t2.minute;
    second = t2.second;
}
```

- Use setT1toT2(Time)

```
Time t1;
Time t2;
t1.setTime(10, 10, 10);
t2.setTime(11, 11, 11);
t1.setT1toT2(t2);  // t1 = t2;
t1.printStandard(); // 11:11:11 AM is printed
```

Lab 2: Extend Time Class

- Add setT1toT2(Time) to Time class
- Add a member function compareTime(Time t2) to Time class to see whether t2's time is later than the object's time being compared. If t2's time is later, print out "Later". If t2's time is early, print out "Earlier". Otherwise, print "Same".
- The main() is given but is not complete and contains syntax bugs. You have to remove the bugs so that the main function can be correctly compiled. You also have to add some statements to generate the output exactly as shown in this document. Note that you can only add or modify a statement, but you cannot delete any statements or remove any comments in the main() function.
  - ✓ If a statement (line) is added, put a comment //#################### behind the statement. For example,
    Time tx; //####################
  - ✓ If a statement is modified, put a comment //********** behind the statement. For example,
    tPtr.setTime(0, 0, 8); //********

main()

```
  int main()
{
   Time t; // instantiate object t of class
Time

   // output Time object t's initial values
   t.printUniversal(); // 00:00:00
   t.printStandard(); // 12:00:00 AM
   // output Time object t's new values
using object name as a handle
   t.printUniversal();
   t.printStandard();
   // output Time object t's new values
using object reference as a handle
   tRef.printUniversal();
   tRef.printStandard();
   // output Time object t's new values
using object pointer as a handle
   tPtr->printUniversal();
   tPtr->printStandard();
```

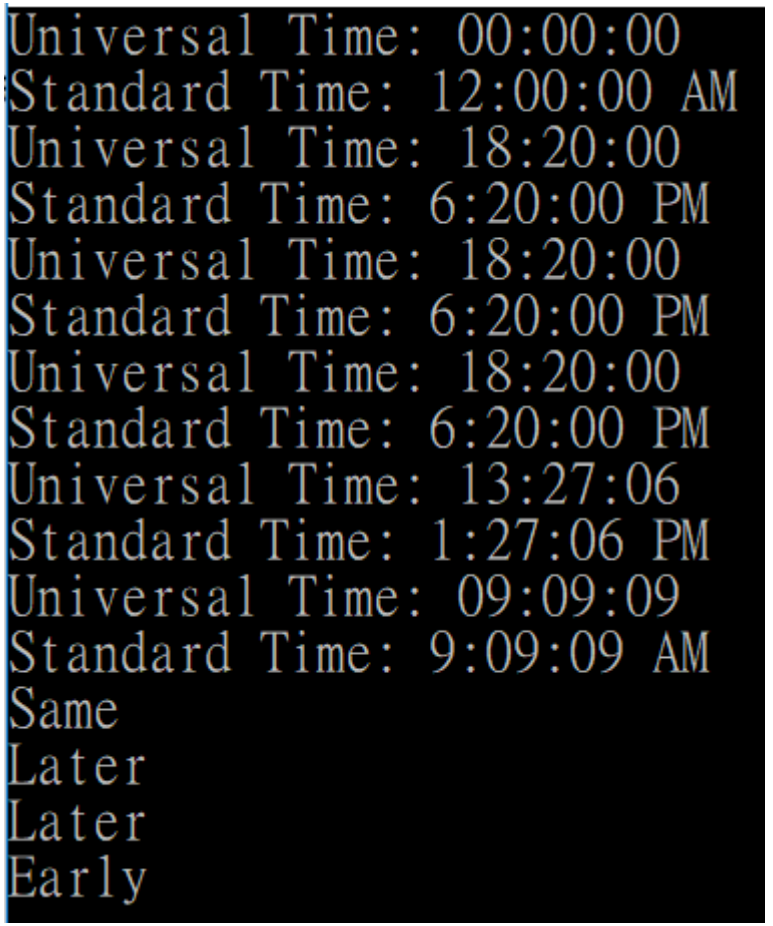
```
 t1.setTime( 13, 27, 6 ); // set a new time
   // output Time object t1's new values
   t1.printUniversal(); // 13:27:06
   t1.printStandard(); // 1:27:06 PM

   t2->setTime( 9, 9, 9);
// output t2's values after specifying invalid
values
   t2->printUniversal();
   t2->printStandard();
   Time t3;
    t3.setT1toT2(t2);
    t3.compareTime(t2);
    t2.compareTime(t1);
    t1.compareTime(t);
    t.compareTime(t1);

} // end main
```

Output

- Your program should exactly generate the following output. Note that printStandard() and printUniversal() may have to be modified to generate desired output.



Key Points for Grading

- The output should be correct.
- There are five lines added and three lines modified.
- A line added should have a comment //########
- A line modified should have a comment //*********
- setT1toT2(Time) and compareTime(Time) should be added.
- The calls to setT1toT2(Time) and compareTime(Time) should remain in the main().