Fundamental Computer Programming - C++ Lab(II)

Lab 2: Basics of Class

02/24/2022

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 →

Lab 2: Extend Time Class

- Add a member function void resetTime( ) to reset time of a time object to hour = 0, minute = 0, and second = 0.
- Add a member function void compareTime(Time t2) to compare the object's time with t2's time. If object's time is later, print out "Later". If object's time is earlier, print out "Earlier". Otherwise, print "Same".
- Add a member function void advanceTime(int numMinutes) to advance the time by *numMinutes*. If the object's time after advancement exceeds the 24 hour limit, the object's hour should be set to the modulo 24. For example, if the hour of a time object after advancement is 27, then its hour should be set to 27 modulo 24, which is equal to 3.

main()

- The main() is given but is incomplete and contains syntax bugs. You have to remove the bugs so that the main() function can be correctly compiled. You must also add some statements to generate the output exactly same as that shown in the example output.
- You can only add 4 statements(lines) and modify two statements (lines) in main(). 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();
   // advance time by 360 minutes
   tPtr->advanceTime(360);
   tPtr->printUniversal();
   tPtr->printStandard();
    // rest Time object t
   tRef.resetTime();
   t.printUniversal();
   t.printStandard();
   t1.setTime( 23, 23, 23 ); // set a new time
   // output Time object t1's new values
   t1.printUniversal(); // 23:23:23
   t1.printStandard(); // 11:23:23 PM

   t1.compareTime(tPtr);
   tPtr.compareTime(t1);
   t1.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 four lines added and two lines modified.
- A line added should have a comment //########
- A line modified should have a comment //*********
- resetTime(), advanceTime,and compareTime(Time) should be added.
- No lines are deleted.