AST21105 Object-Oriented Programming & Design

Lab 7 – Classes and Objects

A. Submission Details

In this lab, you are required to submit **ONE** C++ program with **TWO** files to solve the given problem shown in the section "Date Class". To make the program implementation easier, you are suggested to write your program using Visual Studio .NET instead of doing it directly using paper & pencil. After you have completed the implementation, submit your program files (i.e. Date.h and Date.cpp for this lab) using the electronic "drop-box" in Canvas, **One week** after your lab section is conducted. For details, please refer to the following:

Tuesday sections	*by*	*19:00 on 8 March 2016 (Tuesday)*
Wednesday sections	*by*	*19:00 on 9 March 2016 (Wednesday)*

You are reminded to double-check your solution to verify everything is correct before submission. You are also required to put your name and student ID at the beginning of your source file.

Important: You are only able to submit your work once.

B. Objective

User-defined type makes our program data well-organized and can be manipulated more appropriately. Also, it enhances the readability and re-usability of our program code. The objective of this lab is to help you get familiar with the procedure on creating user-defined type that you have learned in the last lecture. The first part of this lab is meant to be a simple review of classes and objects. While the second part of this lab is a practical task that allows you to use all the techniques introduced.

C. Review

1. What is a class?

   C++ Class is a mechanism used to define a **new type (user-defined type)**. It combines two things together:

   (1) Data (i.e. variables) – **Data members**
   (2) Functions – **Member functions**

2. What is the syntax used to define a new type?

   The keyword “class” is used to define a **new type** and the following is the syntax.

```
class <class name>
{
        // private data members and member functions
        <access specifier>:
                // data members and member functions
        <access specifier>:
                // data members and member functions
} ;
```

where <class name> is the name of the **new type**, <access specifier> is one of three C++ keywords shown below, which used to specify the access right of members (i.e. data members and member functions).

a) public
b) protected
c) private

3. Example

In the following, we show how to create a user-defined type called **Point3D**, which allows us to store (via data members) and manipulate (via member functions) a 3-dimensional point (i.e. x, y and z coordinates).

```
#ifndef POINT3D_H
#define POINT3D_H

class Point3D
{
    private:
        double _x;
        double _y;
        double _z;
    public:
        double getX();
        double getY();
        double getZ();
        void setX(double x);
        void setY(double y);
        void setZ(double z);
};
#endif
```

Point3D.h

```
#include "Point3D.h"
double Point3D::getX() {
    return _x;
}
double Point3D::getY() {
    return _y;
}
double Point3D::getZ() {
    return _z;
}
void Point3D::setX(double x) {
    _x = x;
}
void Point3D::setY(double y) {
    _y = y;
}
void Point3D::setZ(double z) {
    _z = z;
}
```

Point3D.cpp

4. How to use user-defined type?

The usage of user-defined type is very similar to those regular types (i.e. short, int, long, float, double, char, bool). We can use it to declare a variable. The following is an example.

```
#include <iostream>

#include "Point3D.h"

using namespace std;


int main()

{

    Point3D p;

    p.setX(10.1);

    p.setY(20.4);
    p.setZ(30.5);

    cout << "Point3D (x,y,z): "

    << p.getX() << " " << p.getY() << " " << p.getZ() << endl;

    system("pause");

    return 0;

 }
```

main.cpp

From the example above, we can see that the variable p (**also called object**) is in type Point3D. Using p, we can call its member functions using **"." operator**.

5. More details about access specifiers

In C++, access specifiers (i.e. public, private & protected) can be used to set the access right for each member (i.e. data member or member function).

a) public: accessible to anybody, e.g.
   i. Member functions in class
   ii. Member functions in derived class (will be covered later in the course)
   iii. Global functions (i.e. the functions that are not in class)

   (Note: Avoid declaring data members as public!)

b) private: accessible only to
   i. Member functions in class
   ii. Friends of the class (will be covered later in the course)

   (Note: Highly recommend to declare data members as private!)

c) protected: accessible only to
   i. Member functions in class
   ii. Friends of the class (will be covered later in the course)
   iii. Member functions of its derived class / subclasses (will be covered later in the course)

Example:

```
#include <iostream>
#include "Point3D.h"
using namespace std;

int main()
{
    Point3D p;
    p._x = 10.1;            // ERROR: x is private
    p._y = 20.4;            // ERROR: y is private
    p._z = 30.5;            // ERROR: z is private
    p.setX(10.1);           // OK: setX is public
    p.setY(20.4);           // OK: setY is public
    p.setZ(30.5);           // OK: setZ is public
    system("pause");
    return 0;
}
```

main_demo.cpp

D. Date Class

In this part, you are required to:

i) Create a New Project and give your project the name Lab7.

ii) In this lab, you are required to create a class **Date** that has the following features.

A **Date** object should be able to hold **THREE** values, which represent day, month and year. Also, it consists of a number of functions to manipulate the data.

The UML class diagram is given below:

Date
- day: int - month: int - year: int - monthNames: string[12] - monthDays: int[12]
+ initialize() : void + leapYear() : bool + daysInMonth() : int + setDay(int dd) : void + setMonth(int mm) : void + setYear(int yyyy) : void + toMonthDayYear() : string + toMonthNameDate() : string + convertFromMonthName(string monthName) : void

where

- day: data member to store the day of month
- month: data member to store the month
- year: data member to store the year
- monthNames: data member to store the name of month. It is declared as an array of 12 string, i.e. January, February, March, April, May, June, July, August, September, October, November, December

- monthDays: data member to store the number of days in each month. It is declared as an array of 12 int, i.e. 31, 28, 31, 30, 31, 30, 31, 31, 30, 31, 30, 31
- initialize(): a member function to initialize the arrays – monthNames and monthDays
- leapYear(): a member function to verify whether the year is leap or not.

  A leap year should satisfy one of the following conditions

  i. Year is a multiple of 400

  ii. Year is a multiple of 4 AND year is NOT a multiple of 100

  This function returns true if the year is leap, false otherwise.
- daysInMonth(): a member function to return number of days in the month.

  (Note: checking of leap year is required in order to return correct information. Use leapYear() to do this.)
- setDay(int dd): a member function to assign dd to day.

  (Note: checking is required before assigning the value. Use daysInMonth() to do this.)
- setMonth(int mm): a member function to assign mm to month.

  (Note: checking is required before assigning the value.)
- setYear(int yyyy): a member function to assign yyyy to year.

  (Note: checking is required before assigning the value. If yyyy is greater than or equal to 1900 **AND** yyyy is less than or equal to 2100, then assign yyyy to year, otherwise simply assign 1900 to yyyy.)
- toMonthDayYear(): a member function, which returns a string representing the date in the format: MM/DD/YYYY, e.g. "3/9/2009"
- toMonthNameDate(): a member function, which returns a string representing the date in the format: Month-Name Day Year, e.g. "March 9, 2009"
- convertFromMonthName(string monthName): a member function, which converts assign month according to monthName.

  (Note: If the monthName is not valid, i.e. not the string from January to December, simply assign month = 1.)

iii) Amain function (in main.cpp) has been provided for you to test out your class.

At startup, the system should clear the screen and show a text-based menu:

```
Enter 1 for format: MM/DD/YYYY
Enter 2 for format: Month DD, YYYY
Enter 3 to exit

Choice:
```

After reading the choice, the program should perform let the user input the date according to the selected format.

Example output should look like this:

```
Enter 1 for format: MM/DD/YYYY
Enter 2 for format: Month DD, YYYY
Enter 3 to exit

Choice: 1

Enter Month (1-12): 3
Enter Day of Month: 9
Enter Year: 2009
3/9/2009
March 9 2009
Press any key to continue ...
```

Another example is as follows:

```
Enter 1 for format: MM/DD/YYYY
Enter 2 for format: Month DD, YYYY
Enter 3 to exit

Choice: 2

Enter Month Name: March
Enter Day of Month: 9
Enter Year: 2009
3/9/2009
March 9 2009
Press any key to continue ...
```

Quit the program:

```
Enter 1 for format: MM/DD/YYYY
Enter 2 for format: Month DD, YYYY
Enter 3 to exit

Choice: 3
```

Hints:

How to convert int to string in C++?

The following is an example demonstrating how to convert an int to string in C++.

```
#include <iostream>
#include <sstream>
using namespace std;

int main()
{
    int i = 10;
    stringstream s;
    s << i;
    cout << s.str() << endl;
    system("pause");
    return 0;
}
```

int_to_string.cpp

Marking Scheme:

Graded items	Weighting
1. Correctness of program (i.e. whether your code is implemented in a way according to the requirements as specified.)	60%
2. Indentation	30%
3. Documentation (with reasonable amount of comments embedded in the code to enhance the readability.)	10%
	100%

Program Submission Checklist

Before submitting your work, please check the following items to see you have done a decent job.

Items to be checked	☑ / ☒
1. Did I put my name and student ID at the beginning of all the source files?	☐
2. Did I put reasonable amount of comments to describe my program?	☐
3. Are they all in .cpp extension and named according to the specification?	☐
4. Have I checked that all the submitted code are compliable and run without any errors?	☐
5. Did I zip my source files using Winzip / zip provided by Microsoft Windows? Also, did I check the zip file and see if it could be opened? ***(Only applicable if the work has to be submitted in zip format.)***	☐
6. Did I submit my lab assignment to Canvas?	☐

-End-