Lab 13

LAB 13.1 Declaring Classes & Writing Client Code

Retrieve program `square.cpp` from the Lab 13 folder. The code is as follows:

```
// This program declares the Square class and uses member functions to find
// the perimeter and area of the square
#include <iostream>
using namespace std;

// FILL IN THE CODE TO DECLARE A CLASS CALLED Square. TO DO THIS SEE
// THE IMPLEMENTATION SECTION.

int main()
{
    Square box;    // box is defined as an object of the Square class
    float size;    // size contains the length of a side of the square

    // FILL IN THE CLIENT CODE THAT WILL ASK THE USER FOR THE LENGTH OF THE
    // SIDE OF THE SQUARE. (This is stored in size)

    // FILL IN THE CODE THAT CALLS SetSide.

    // FILL IN THE CODE THAT WILL RETURN THE AREA FROM A CALL TO A FUNCTION
    // AND PRINT OUT THE AREA TO THE SCREEN.

    // FILL IN THE CODE THAT WILL RETURN THE PERIMETER FROM A CALL TO A
    // FUNCTION AND PRINT OUT THAT VALUE TO THE SCREEN.

    return 0;
}
//
//Implementation section Member function implementation
//************************************************
// setSide
//
// task: This procedure takes the length of a side and
// places it in the appropriate member data
// data in: length of a side
//************************************************
void Square::setSide(float length)
{
    side = length;
}

//************************************************
// findArea
//
// task: This finds the area of a square
// data in: none (uses value of data member side)
// data returned: area of square
//************************************************
float Square::findArea()
{
    return side * side;
}
//************************************************
// findPerimeter
//
// task: This finds the perimeter of a square
// data in: none (uses value of data member side)
// data returned: perimeter of square
//************************************************
float Square::findPerimeter()
```

```
{
    return 4 * side;
}
```

Exercise 1

This program asks you to fill in the class declaration based on the implementation of the member functions.

Hint: To write your class declaration, observe carefully what the name of the class is, and what the potential members (data and functions) of the class are. Also, decide if the members should be granted public or private access.

Exercise 2

Fill in the client code (in main ( ) function) so that the following input and output will be generated:

Sample Run:

```
Please input the length of the side of the square
8
The area of the square is 64
The perimeter of the square is 32
```

Exercise 3

Add two constructors and a destructor to the class and create the implementation of each. One constructor is the default constructor that sets the side to 1. The other constructor will allow the user to initialize the side at the definition of the object. The destructor does not have to do anything (memory space is reclaimed when the object is destroyed). Create another object called `box1` that gives the value of 9 to the constructor at the definition. Add output statements so that the following is printed in addition to what is printed in Exercise 2.

Sample Run:

```
…
The area of box1 is 81
The perimeter of box1 is 36
```

LAB 13.2 Implementing Classes

Retrieve program `circles.cpp` from the Lab 13 folder. The code is as follows:

```
#include <iostream>
using namespace std;
//
// This program declares a class for a circle that will have
// member functions that set the center, find the area, find
// the circumference and display these attributes.
// The program as written does not allow the user to input data, but
// rather has the radii and center coordinates of the circles
// (spheres in the program) initialized at definition or set by a function.

// class declaration section (header file)
class Circles
{
    public:
        void setCenter(int x, int y);
        double findArea();
        double findCircumference();
```

```
        void printCircleStats(); // This outputs the radius and center of the circle.
Circles (float r); // Constructor
        Circles(); // Default constructor
    private:
        float radius;
        int center_x;
        int center_y;
};

const double PI = 3.14;

// Client section
int main()
{
    Circles sphere(8);
    sphere.setCenter(9,10);           // Exercise 1: this line cannot be used

    sphere.printCircleStats();
    cout << "The area of the circle is " << sphere.findArea() << endl;
    cout << "The circumference of the circle is "
         << sphere.findCircumference() << endl;

    return 0;
}

//Implementation section      Member function implementation

Circles::Circles()
{
    radius = 1;
}

// Fill in the code to implement the non-default constructor

// Fill in the code to implement the findArea member function

// Fill in the code to implement the findCircumference member function


void Circles::printCircleStats()
// This procedure prints out the radius and center coordinates of the circle
// object that calls it.
{
    cout << "The radius of the circle is " << radius << endl;
    cout << "The center of the circle is (" << center_x
         << "'" << center_y << ")" << endl;
}

void Circles::setCenter(int x, int y)
// This procedure will take the coordinates of the center of the circle from
// the user and place them in the appropriate member data.
{
    center_x = x;
    center_y = y;
}
```

Exercise 1

Modify the code so that setting the center of the circle is also done during the object definition. This means that the constructors will also take care of this initialization. Make the default center at point (0, 0) and keep the default radius as 1. Have `sphere` object defined with initial values of 8 for the radius and (9, 10) for the center. How does this affect existing functions and code in the main function? The following output should be produced.

Sample Run:

```
The radius of the circle is 8
The center of the circle is (9, 10)
```

```
The area of the circle is 200.96
The circumference of the circle if 50.24
```

Note: The function `setCenter` will no longer be called. You can choose to remove this function (both from class declaration and implementation).

Exercise 2

Modify the program so that the user can enter either just the radius, the radius and the center, or nothing at the time the object is defined. You can continue to assume that the default radius is 1 and the default center is (0, 0). Go on and define the following objects:

- An object `sphere1`, giving just radius 2
- An object `sphere2`, giving neither the radius nor the center
- An object `sphere3`, giving just the center (15, 16)

Additional Output:

```
The radius of the circle is 2
The center of the circle is (0, 0)
The area of the circle is 12.56
The circumference of the circle if 12.56

The radius of the circle is 1
The center of the circle is (0, 0)
The area of the circle is 3.14
The circumference of the circle if 6.28

The radius of the circle is 1
The center of the circle is (15, 16)
The area of the circle is 3.14
The circumference of the circle if 6.28
```

Exercise 3

Add a destructor to the code. It should print the message "**This concludes the Circles class"** for each object that is destroyed. How many times is this printed? Why?

LAB 13.3 Array of Objects

Retrieve program `inventory.cpp` and `Inventory.dat` from the Lab 13 folder. The code is as follows:

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

// This program declares a class called Inventory that has itemnNumber (which
// contains the id number of a product) and numOfItem (which contains the
// quantity on hand of the corresponding product)as private data members.
// The program will read these values from a file and store them in an
// array of objects (of type Inventory). It will then print these values
// to the screen.

// Example: Given the following data file:
// 986 8
// 432 24

// This program reads these values into an array of objects and prints the
// following:
```

```
// Item number 986 has 8 items in stock
// Item number 432 has 24 items in stock

const NUMOFPROD = 10; // This holds the number of products a store sells class Inventory
{
    public:
        void getId(int item);   // This puts item in the private data member
                                // itemNumber of the object that calls it.
        void getAmount(int num); // This puts num in the private data member
                                // numOfItem of the object that calls it.
        void display();     // This prints to the screen
                            // the value of itemNumber and numOfItem of the
                            // object that calls it.
    private:
        int itemNumber;    // This is an id number of the product
        int numOfItem;     // This is the number of items in stock
};

int main()
{
    ifstream infile;        // Input file to read values into array
    infile.open("Inventory.dat");

    // Fill in the code that defines an array of objects of class Inventory
    // called products. The array should be of size NUMOFPROD


    int pos; // loop counter
    int id; // variable holding the id number
    int total; // variable holding the total for each id number

    // Fill in the code that will read inventory numbers and number of items
    // from a file into the array of objects. There should be calls to both
    // getId and getAmount member functions somewhere in this code.
    // Example: products[pos].getId(id); will be somewhere in this code


    // Fill in the code to print out the values (itemNumber and numOfItem) for
    // each object in the array products.
    // This should be done by calling the member function display within a loop


    return 0;
}

// Write the implementations for all the member functions of the class.
```

Exercise 1

Complete the program (at locations explained with instructions in bold), to produce the output below.

Sample Run:

```
Item number 986 has 8 items in stock
Item number 432 has 24 items in stock
Item number 132 has 100 items in stock
Item number 123 has 89 items in stock
Item number 329 has 50 items in stock
Item number 503 has 30 items in stock
Item number 783 has 78 items in stock
Item number 822 has 32 items in stock
Item number 233 has 56 items in stock
Item number 322 has 74 items in stock
```

LAB 13.4 Student Generated Code Assignments

Option 1:

Give a C++ class declaration called `SavingsAccount` with the following information:

Operations (Member Functions)

1. Open account (with an initial deposit). This is called to put initial values in `dollars` and `cents`.
2. Make a deposit. A function that will add value to `dollars` and `cents`.
3. Make a withdrawal. A function that will subtract values from `dollars` and `cents`.
4. Show current balance. A function that will print `dollars` and `cents`.

Data (Member Data)

1. `dollars`
2. `cents`

Part 1: Write the implementation code for all the member functions.

NOTE: You must perform normalization on `cents`. This means that if `cents` is 100 or more, it must increment `dollars` by the appropriate amount. Example: if `cents` is 234, then `dollars` must be increased by 2 and `cents` reduced to 34.

Write code that will create an object called `bank1`. The code will then initially place $200.50 in the account. The code will deposit $40.50 and then withdraw $100.98. It will print out the final value of `dollars` and `cents`.

Part 2: Change the program to allow the user to input the initial values, deposit and withdrawal.