Lab 8

## LAB 8.1 Working with Basic Structures

Retrieve program rect\_struct.cpp from the Lab 8 folder. The code is as follows:

#include <iostream>

#include <iomanip>

using namespace std;

// This program uses a structure to hold data about a rectangle

**// PLACE YOUR NAME HERE**

**// Fill in code to declare a structure named rectangle which has**

**// members length, width, area, and perimeter all of which are floats**

int main()

{

**// Fill in code to define a rectangle variable named box**

cout << "Enter the length of a rectangle: ";

**// Fill in code to read in the length member of box**

cout << "Enter the width of a rectangle: ";

**// Fill in code to read in the width member of box**

cout << endl << endl;

**// Fill in code to compute the area member of box**

**// Fill in code to compute the perimeter member of box**

cout << fixed << showpoint << setprecision(2);

**// Fill in code to output the area with an appropriate message**

**// Fill in code to output the perimeter with an appropriate message**

return 0;

}

Exercise 1

Fill in the code (places in bold) so that the program will compute and print out the area and perimeter of the rectangle based on the user’s input length and width.

Exercise 2

Add code to the program so that the modified program will determine whether or not the rectangle entered by the user is a square.

Sample Run:

Enter the length of a rectangle: 7

Enter the width of a rectangle: 7

The area of the rectangle is 49.00

The perimeter of the rectangle is 28.00

The rectangle is a square.

**LAB 8.2 Initializing Structures**

Retrieve program init\_struct.cpp from the Lab 8 folder. The code is as follows:

#include <iostream>

#include <string>

#include <iomanip>

using namespace std;

// This program demonstrates partially initialized structure variables

**// PLACE YOUR NAME HERE**

struct taxPayer

{

string name;

long socialSecNum;

float taxRate;

float income;

float taxes;

};

int main()

{

**// Fill in code to initialize a structure variable named citizen1 so that**

**// the first three members are initialized. Assume the name is Tim**

**// McGuiness, the social security number is 255871234, and the tax rate is .35**

**// Fill in code to initialize a structure variable named citizen2 so that**

**// the first three members are initialized. Assume the name is John Kane,**

**// the social security number is 278990582, and the tax rate is .29**

cout << fixed << showpoint << setprecision(2);

// calculate taxes due for citizen1

**// Fill in code to prompt the user to enter this year's income for the citizen1**

**// Fill in code to read in this income to the appropriate structure member**

**// Fill in code to determine this year's taxes for citizen1**

cout << "Name: " << citizen1.name << endl;

cout << "Social Security Number: " << citizen1.socialSecNum << endl;

cout << "Taxes due for this year: $" << citizen1.taxes << endl << endl;

// calculate taxes due for citizen2

**// Fill in code to prompt the user to enter this year's income for citizen2**

**// Fill in code to read in this income to the appropriate structure member**

**// Fill in code to determine this year's taxes for citizen2**

cout << "Name: " << citizen2.name << endl;

cout << "Social Security Number: " << citizen2.socialSecNum << endl;

cout << "Taxes due for this year: $" << citizen2.taxes << endl << endl;

return 0;

}

Exercise 1

Fill in the code (places in bold) to make the program work as follows:

Sample Run:

Please input the yearly income for Tim McGuiness: 30000

Name: Tim McGuiness

Social Security Number: 255871234

Taxes due for this year: $10500.00

Please input the yearly income for John Kane: 60000

Name: John Kane

Social Security Number: 278990582

Taxes due for this year: $17400.00

**LAB 8.3 Arrays of Structures**

Retrieve program array\_struct.cpp from the Lab 8 folder. The code is as follows:

#include <iostream>

#include <iomanip>

using namespace std;

// This program demonstrates how to use an array of structures

**// PLACE YOUR NAME HERE**

**// Fill in code to declare a structure called taxPayer that has three**

**// members: taxRate, income, and taxes — each of type float**

int main()

{

**// Fill in code to define an array named citizen which holds**

**// 5 taxPayers structures**

cout << fixed << showpoint << setprecision(2);

cout << "Please enter the annual income and tax rate for 5 tax payers: ";

cout << endl << endl << endl;

for(int count = 0;count < 5;count++)

{

cout << "Enter this year's income for tax payer " << (count + 1);

cout << ": ";

**// Fill in code to read in the income to the appropriate place**

cout << "Enter the tax rate for tax payer # " << (count + 1);

cout << ": ";

**// Fill in code to read in the tax rate to the appropriate place**

**// Fill in code to compute the taxes for the citizen and store it**

**// in the appropriate place**

cout << endl;

}

cout << "Taxes due for this year: " << endl << endl;

**// Fill in code for the first line of a loop that will output the**

**// tax information**

{

cout << "Tax Payer # " << (index + 1) << ": " << "$ "

<< citizen[index].taxes << endl;

}

return 0;

}

Exercise 1

Fill in the code (places in bold) so that the program is able to read in 5 taxpayer’s income and tax rate. The program should also output the tax information of all 5 taxpayers.

Sample Run:

Enter this year’s income for tax payer 1: 45000

Enter the tax rate for tax payer # 1: .19

Enter this year’s income for tax payer 1: 60000

Enter the tax rate for tax payer # 1: .23

Enter this year’s income for tax payer 1: 12000

Enter the tax rate for tax payer # 1: .01

Enter this year’s income for tax payer 1: 104000

Enter the tax rate for tax payer # 1: .30

Enter this year’s income for tax payer 1: 50000

Enter the tax rate for tax payer # 1: .22

Tax Payer # 1: $ 8550.00

Tax Payer # 2: $ 13800.00

Tax Payer # 3: $ 120.00

Tax Payer # 4: $ 31200.00

Tax Payer # 5: $ 11000.00

Exercise 2

In the previous code we have the following:

cout << "Tax Payer # " << (index+1) << ": " << "$ " << citizen[index].taxes << endl;

Why do you think we need (index+1) in the first line but index in the second line?

**LAB 8.4 Declaring Classes & Writing Client Code**

Retrieve program square.cpp from the Lab 8 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 8.5 Implementing Classes**

Retrieve program circles.cpp from the Lab 8 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?

It printed 3 times , this is because it prints for each sphere

**LAB 8.6 Array of Objects**

Retrieve program inventory.cpp and Inventory.dat from the Lab 8 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 8.7 Student Generated Code Assignments (Structures & Classes)**

Option 1: Re-write your final program in Lab 8.2 so that it uses an array of structures. Write the program so that the array will store many taxPayer structures. You may choose to maintain the same program output as before, or you may opt to modify the program for better usability (e.g. allow user to continue entering new taxpayer entries, with a sentinel value to stop).

Option 2:Write a program that uses a structure to store the following information for a particular month at the local airport:

* Total number of planes that landed
* Total number of planes that departed
* Greatest number of planes that landed in a given day that month
* Least number of planes that landed in a given day that month

The program should have an array of twelve structures to hold travel information for the entire year. The program should prompt the user to enter data for each month. Once all data is entered, the program should calculate and output the average monthly number of landing planes, the average monthly number of departing planes, the total number of landing and departing planes for the year, and the greatest and least number of planes that landed on any one day (and which month it occurred in).

Option 3:

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.