Jeremy Scheuerman

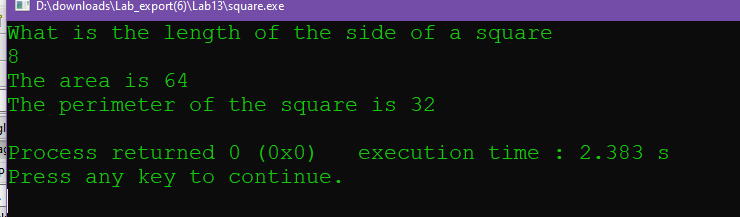
COSC 120

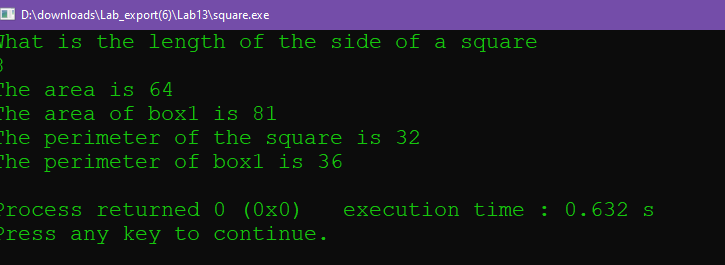
Dr. Wang

4 December 2020

Lab 10 (13) classes

13.1

1. 

2. 

Source Code:

// This program declares the Square class and uses member functions to find

// the perimeter and area of the square

// PLACE YOUR NAME HERE

#include <iostream>

using namespace std;

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

// THE IMPLEMENTATION SECTION.

class Square

{

private:

int side;

public:

Square()

{

side=1;

}

//constructor

Square(float size)

{

side=size;

}

~Square();

//destructor

void setSide(float length);

float findArea();

float findPerimeter();

};

Square::~Square()

{

}

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

float size1=9;

float area;

float area1;

float perimeter;

float perimeter1;

float side=0;

// FILL IN THE CLIENT CODE THAT WILL ASK THE USER FOR THE LENGTH OF THE

// SIDE OF THE SQUARE. (This is stored in size)

cout<<"What is the length of the side of a square"<<endl;

cin>>size;

// FILL IN THE CODE THAT CALLS SetSide.

Square Box1( size1);

box.setSide(size);

//set box side value

// FILL IN THE CODE THAT WILL RETURN THE AREA FROM A CALL TO A FUNCTION

// AND PRINT OUT THE AREA TO THE SCREEN.

area=box.findArea();

area1=Box1.findArea();

cout<<"The area is "<<area<<endl;

cout<<"The area of box1 is "<<area1<<endl;

// FILL IN THE CODE THAT WILL RETURN THE PERIMETER FROM A CALL TO A

// FUNCTION AND PRINT OUT THAT VALUE TO THE SCREEN.

perimeter=box.findPerimeter();

perimeter1=Box1.findPerimeter();

cout<<"The perimeter of the square is "<<perimeter<<endl;

cout<<"The perimeter of box1 is "<<perimeter1<<endl;

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

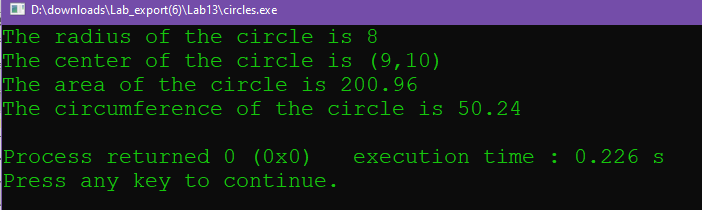
//\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

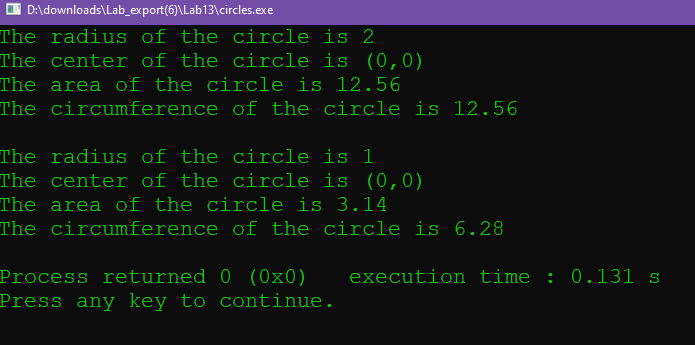
float Square::findPerimeter()

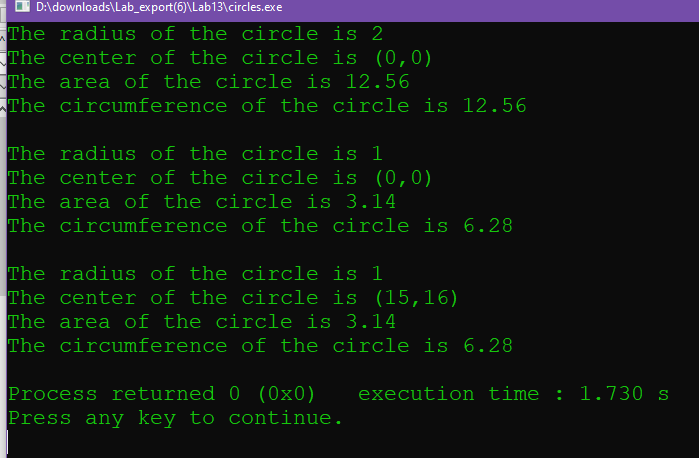
{

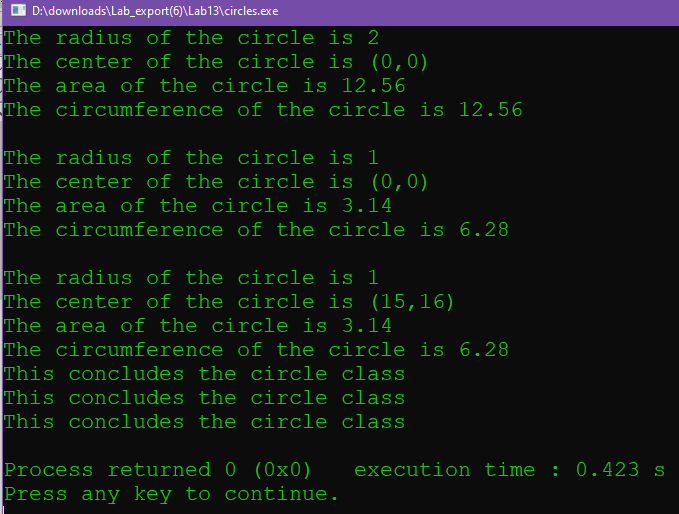
return (4 \* side);

}

13.2  
1. 

2

3 

4

It does it 3 times because there are 3 instances created in that moment

Source Code

#include <iostream>

#include <iomanip>

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)

// PLACE YOUR NAME HERE

class Circles

{

public:

void setCenter(int x, int y);

double findArea();

double findCircumference();

void printCircleStats();

// This outputs the radius and center of the circle.

float r;

Circles(float r); // Constructor

// Circles(default);

Cicles(float r=1){

radius=r;

}

// Default constructor

~Circles();

private:

float radius;

int center\_x;

int center\_y;

};

const double PI = 3.14;

// Client section

int main()

{

Circles sphere(2);

// sphere.setCenter(0, 0);

sphere.printCircleStats();

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

cout << "The circumference of the circle is "

<< sphere.findCircumference() << endl;

cout<<endl;

Circles sphere1(1);

//sphere1.setCenter(0, 0);

sphere1.printCircleStats();

cout << "The area of the circle is " << sphere1.findArea() << endl;

cout << "The circumference of the circle is "

<< sphere1.findCircumference() << endl;

cout<<endl;

Circles sphere3(1);

sphere3.setCenter(15, 16);

sphere3.printCircleStats();

cout << "The area of the circle is " << sphere3.findArea() << endl;

cout << "The circumference of the circle is "

<< sphere3.findCircumference() << endl;

return 0;

}

// \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

//

// Implementation section Member function implementation

Circles::Circles(float r)

{

radius=r;

center\_x=0;

center\_y=0;

//default center is 0,0

}

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

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

double Circles::findArea()

{

double area=PI\*(radius\*radius);

return area;

}

double Circles::findCircumference()

{

double circ=2\*PI\*radius;

return circ;

}

Circles::~Circles()

{

cout<<"This concludes the circle class"<<endl;

}

// 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.

{

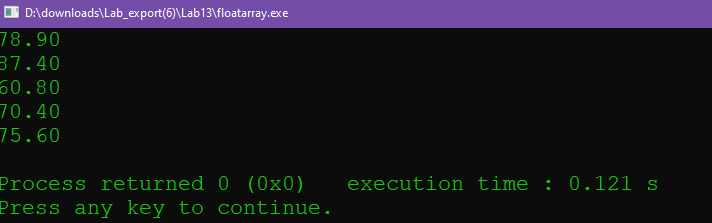
center\_x = x;

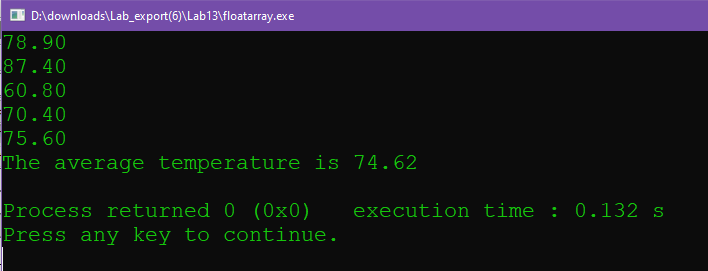
center\_y = y;

}

13.3

1. It is declared constant because that does not allow any of its members to be changed without resulting in a compiler error

2. 

3. 

Soruce Code:

// This program reads floating point data from a data file and places those

// values into the private data member called values (a floating point array)

// of the FloatList class. Those values are then printed to the screen.

// The input is done by a member function called GetList. The output

// is done by a member function called PrintList. The amount of data read in

// is stored in the private data member called length. The member function

// GetList is called first so that length can be initialized to zero.

// PLACE YOUR NAME HERE

#include <iostream>

#include <fstream>

#include <iomanip>

using namespace std;

const int MAX\_LENGTH = 50; // MAX\_LENGTH contains the maximum length of our list

class FloatList // Declares a class that contains an array of

{

public:

void getList(ifstream& tempData); // Member function that gets data from a file

void printList() const; // Member function that prints data from that

void printAvg();

// file to the screen.

FloatList(); // constructor that sets length to 0.

~FloatList(); // destructor

private:

int length; // Holds the number of elements in the array

float values[MAX\_LENGTH]; // The array of values

};

int main()

{

ifstream tempData; // Defines a data file

// Fill in the code to define an object called list of the class FloatList

FloatList list;

cout << fixed << showpoint;

cout << setprecision(2);

tempData.open("temperatures.txt");

// Fill in the code that calls the getList function.

list.getList(tempData);

// Fill in the code that calls the printList function.

list.printList();

list.printAvg();

return 0;

}

void FloatList::getList(ifstream& tempData)

{

length=5;

float input;

for (int i=0; i<length; i++)

{

tempData>>input;

values[i]=input;

}

}

// Fill in the entire code for the getList function

// The getList function reads the data values from a data file

// into the values array of the class FloatList

void FloatList::printList() const

{

for (int i=0; i<length; i++)

{

cout<<values[i]<<endl;

}

}

// Fill in the entire code for the printList function

// The printList function prints to the screen the data in

// the values array of the class FloatList

void FloatList::printAvg()

{

float total=0;

float avg=0;

for (int i=0; i<length; i++)

{

total+=values[i];

}

avg=total/length;

cout<<"The average temperature is "<<avg<<endl;

}

FloatList::FloatList()

{

length=0;

// Fill in the code to complete this constructor that

// sets the private data member length to 0

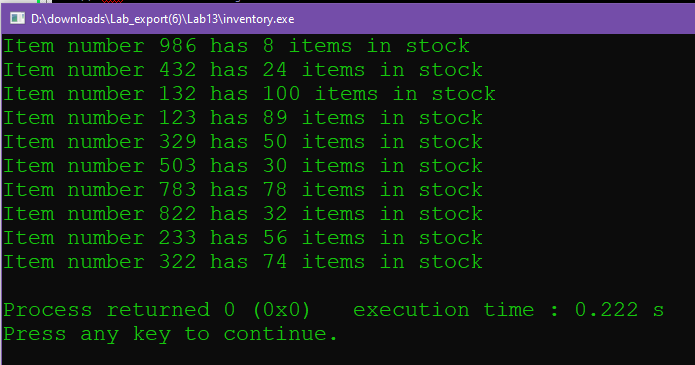
}

FloatList::~FloatList()

{

}

13.4

1

Source Code:

#include <iostream>

#include <fstream>

#include <string>

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.

// PLACE YOUR NAME HERE

// 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 int 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.

int return\_id();

int return\_amnt();

//return the values

private:

int itemNumber; // This is an id number of the product

int numOfItem; // This is the number of items in stock

};

void Inventory::display()

{

cout<<"Item number "<<return\_id()<<" has "<<return\_amnt()<<" items in stock"<<endl;

}

void Inventory::getId(int item)

//converts to private

{

itemNumber=item;

}

void Inventory::getAmount(int num)

//converts to private

{

numOfItem=num;

}

int Inventory::return\_id()

{

return itemNumber;

}

int Inventory::return\_amnt()

{

return numOfItem;

}

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

Inventory inv\_array[NUMOFPROD];

int pos; // loop counter

int id; // variable holding the id number

int total; // variable holding the total for each id number

//int getId();

// int getAmount();

for ( pos=0; pos<NUMOFPROD; pos++)

{

infile>>id;

infile>>total;

inv\_array[pos].getId(id);

inv\_array[pos].getAmount(total);

inv\_array[pos].display();

}

// 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

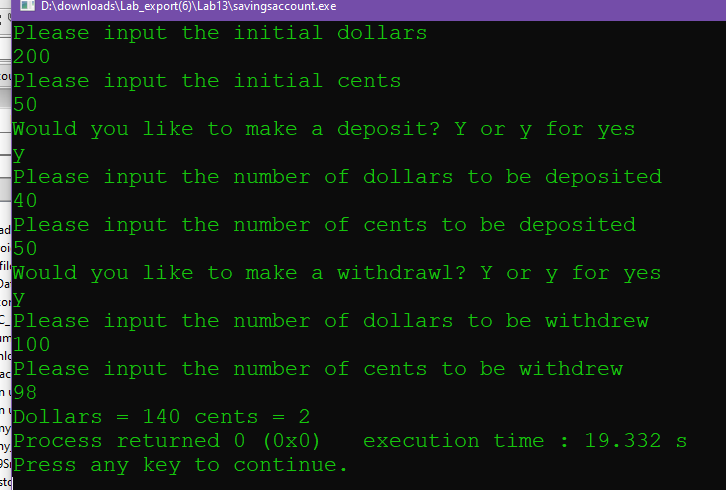
return 0;

}

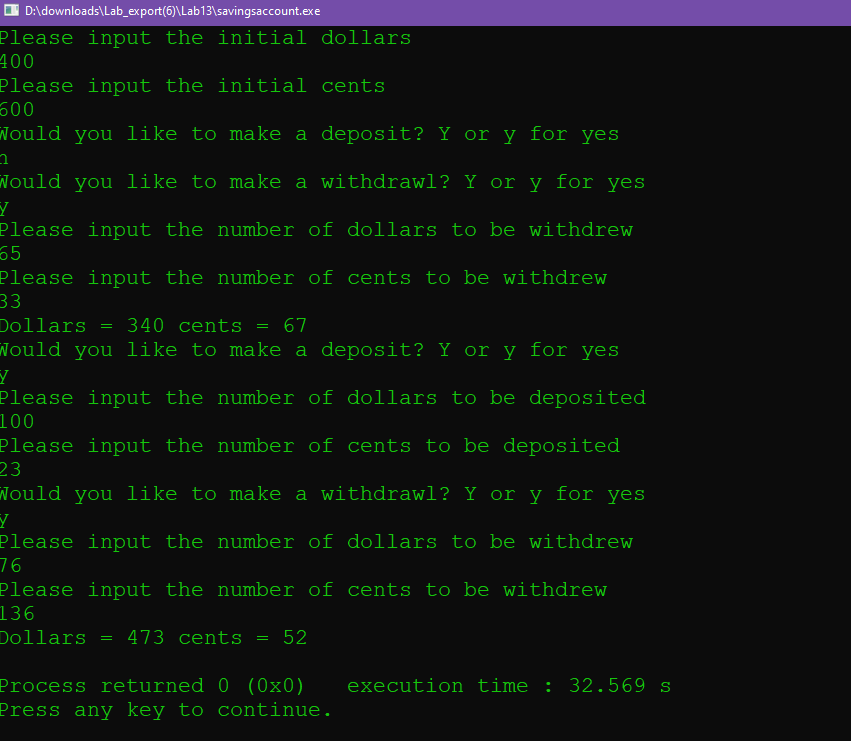
// Write the implementations for all the member functions of the class.

13.5

Exercise 1

1. 2.

Exercise 2



Source Code:

#include <iostream>

using namespace std;

class SavingsAccount

{

public:

void deposit();

void withdrawl();

void show\_balance();

void InitialValues();

SavingsAccount(int doll=0,int cent=0)

//default constructor

{

dollars=doll;

cents=cent;

}

private:

int dollars=0;

int cents=0;

};

void SavingsAccount::deposit()

{

char choice;

int input\_dollars;

int input\_cents;

cout<<"Would you like to make a deposit? Y or y for yes"<<endl;

cin>>choice;

if((choice=='Y') ||(choice=='y'))

{

cout<<"Please input the number of dollars to be deposited"<<endl;

cin>>input\_dollars;

cout<<"Please input the number of cents to be deposited"<<endl;

cin>>input\_cents;

dollars+=input\_dollars;

cents+=input\_cents;

}

else

{

}

}

void SavingsAccount::withdrawl()

{

//withdrawl

char choice;

int withd\_dollars;

int withd\_cents;

cout<<"Would you like to make a withdrawl? Y or y for yes"<<endl;

cin>>choice;

if((choice=='Y') ||(choice=='y'))

{

cout<<"Please input the number of dollars to be withdrew"<<endl;

cin>>withd\_dollars;

cout<<"Please input the number of cents to be withdrew"<<endl;

cin>>withd\_cents;

if (withd\_cents>cents)

{

//transfer a dollar to cents

dollars-=1;

cents+=100;

}

dollars-=withd\_dollars;

cents-=withd\_cents;

//update values

}

else

{

}

}

void SavingsAccount::InitialValues()

{

cout<<"Please input the initial dollars"<<endl;

cin>>dollars;

cout<<"Please input the initial cents"<<endl;

cin>>cents;

//set initial values

}

void SavingsAccount::show\_balance()

{

while (cents>100)

{

//udpate dollars and cents

cents-=100;

dollars+=1;

}

cout<<"Dollars = "<<dollars<<" cents = "<<cents<<endl;

//show dollars and cents

}

int main()

{

SavingsAccount Bank1;

SavingsAccount Bank2(450,65);

//create bank 1

Bank1.InitialValues();

Bank1.deposit();

Bank1.withdrawl();

Bank1.show\_balance();

// Bank2.InitialValues();

Bank2.deposit();

Bank2.withdrawl();

Bank2.show\_balance();

return 0;

}