Lab 2 Instructor: Saroj Shakya

Mechanisms for Creation and Initialization in C++

Constructors

*constructors*are special member function used to i**nitialize the data members of a class in** C++

A *constructor* is a method with the **same name** as that of the object class

A *constructor* method is **automatically** and **implicitly invoked** any time an object of the associated class is created

* During variable declaration (*static creation*)
* During creation of objects *dynamically* with the new operator etc

Syntax Rules for creating Constructor:

* + The name of the constructor must be the same as that of the class
  + Constructors **don’t have return types** (*not even void*) in contrast with normal functions
  + They are declared in the **public** section of a class

Types of constructors:

i. Default Constructor

A constructor **without arguments** is known as **default constructor**

It initializes each data member of an object

class integer{

int n;

public:

**//declaration of default constructor**

integer(void);

void showInteger(){

cout << “ n:” << n;

}

-----------

};

**//definition of default constructor**

integer :: integer(void){

n = 0;

};

void main(){

integer i; **//default constructor called**

i.showInteger();

}

ii. Parameterized Constructor

Arguments can be passed to the constructor function to initialize various data elements of different objects with different values when they are created.

These constructors that **can take arguments** are called **parameterized constructors**

class integer{

int n;

public:

***//parameterized constructor declared***

integer(int a);

void showInteger(){

cout << “ n:” << n;

}

-----------

};

***//definition of parameterized constructor***

integer :: integer(int a){

n = b;

};

void main(){

*// integer i; will not work, why????*

integer i(1); *//implicit call to constructor*

i.showInteger();

}

iii. Copy Constructor

This constructor is used when newly created object is to be initialized using pre existing ones. All the member variables of the existing object will be copied to the member variables of the newly created object

class integer{

int n;

public:

**//default constructor**

integer(){

n = 0;

}

**//parameterized constructor**

integer(int nn){

n = nn;

}

**//copy constructor**

integer(integer &i){

n = i.n;

}

void showInteger(){

cout << " n:" << n <<endl;

}

};

void main(){

integer i1(1); **//using parameterized constructor to initialize value of n of object i1**

integer i2(**i1**); **//using copy constructor to initialize attributes of i2 from i1 object**

i1.showInteger();

i2.showInteger();

}

Destructors

A ***destructor*** is a **function** that is **called automatically whenever an object is destroyed** (*control goes outside main()*)

A *destructor* **has the same name as the constructor (*or class*)** but is **preceded by a tilde ~**

A *destructor* **does not have a *return* value**

A *destructor* **does not take any arguments** (*the assumption being that there are many ways to construct an object but there’s only one way to destroy an object*)

The prime use of destructors is to ***de-allocate* memory** that was allocated for the object by the *constructor*

#include <iostream.h>

#include <conio.h>

int count=0;

class trace{

public:

trace(){

count++;

cout << "\nCreated object no[" << count << "]";

}

~trace(){

cout << "\nDestroyed object no[" << count << "]";

count--;

getch();

}

};

void main(){

cout << "\n In Main:";

trace t1,t2,t3,t4;

{

cout << "\n\n Entering Block One ";

trace t5;

}

{

cout << "\n\n Entering Block Two ";

trace t6;

}

cout << "\n\n Re-entering Main:";

}

Exercises

**1. Employee Class**

Write a class named Employee that has the following member variables:

**• name**. A string that holds the employee’s name.

**• idNumber**. An int variable that holds the employee’s ID number.

**• department**. A string that holds the name of the department where the employee works.

**• position**. A string that holds the employee’s job title.

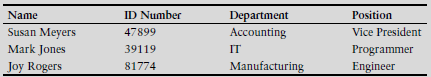
The class should have the following constructors:

**•** A constructor that accepts the following values as arguments and assigns them to the appropriate member variables: employee’s name, employee’s ID number, department, and position.

**•** A constructor that accepts the following values as arguments and assigns them to the appropriate member variables: employee’s name and ID number. The department and position fields should be assigned an empty string ("").

**•** A default constructor that assigns empty strings ("") to the name, department, and position member variables, and 0 to the idNumber member variable.

Write appropriate mutator functions that store values in these member variables and accessor functions that return the values in these member variables. Once you have written the class, write a separate program that creates three Employee objects to hold the following data.



The program should store this data in the three objects and then display the data for each employee on the screen.

**2. Car Class**

Write a class named Car that has the following member variables:

**• yearModel.** An int that holds the car’s year model.

**• make.** A string that holds the make of the car.

**• speed.** An int that holds the car’s current speed.

In addition, the class should have the following constructor and other member functions.

**• Constructor.** The constructor should accept the car’s year model and make as arguments. These values should be assigned to the object’s yearModel and make member variables. The constructor should also assign 0 to the speed member variables.

**• Accessor.** Appropriate accessor functions to get the values stored in an object’s yearModel, make, and speed member variables.

**• accelerate.** The accelerate function should add 5 to the speed member variable each time it is called.

**• brake.** The brake function should subtract 5 from the speed member variable each time it is called.

Demonstrate the class in a program that creates a Car object, and then calls the accelerate function five times. After each call to the accelerate function, get the current speed of the car and display it. Then, call the brake function five times. After each call to the brake function, get the current speed of the car and display it.

**3. Personal Information Class**

Design a class that holds the following personal data: name, address, age, and phone number. Write appropriate accessor and mutator functions. Demonstrate the class by writing a program that creates three instances of it. One instance should hold your information, and the other two should hold your friends’ or family members’ information.

**4. RetailItem Class**

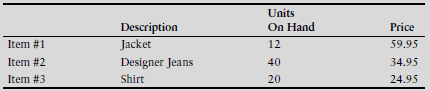
Write a class named RetailItem that holds data about an item in a retail store. The class should have the following member variables:

**• description.** A string that holds a brief description of the item.

**• unitsOnHand.** An int that holds the number of units currently in inventory.

**• price.** A double that holds the item’s retail price.

Write a constructor that accepts arguments for each member variable, appropriate mutator functions that store values in these member variables, and accessor functions that return the values in these member variables. Once you have written the class, Write a separate program that creates three RetailItem objects and stores the following data in them.

****

**5. Widget Factory**

Design a class for a widget manufacturing plant. Assuming that 10 widgets may be produced each hour, the class object will calculate how many days it will take to produce any number of widgets. (The plant operates two shifts of eight hours each per day.) Write a program that asks the user for the number of widgets that have been ordered and then displays the number of days it will take to produce them.

*Input Validation: Do not accept negative values for the number of widgets ordered.*

**6. TestScores Class**

Design a TestScores class that has member variables to hold three test scores. The class should have a constructor, accessor, and mutator functions for the test score fields, and a member function that returns the average of the test scores. Demonstrate the class by writing a separate program that creates an instance of the class. The program should ask the user to enter three test scores, which are stored in the TestScores object. Then the program should display the average of the scores, as reported by the TestScores object.

**7. Circle Class**

Write a Circle class that has the following member variables:

radius: a double pi: a double initialized with the value 3.14159

The class should have the following member functions:

**• Default Constructor.** A default constructor that sets radius to 0.0.

**• Constructor.** Accepts the radius of the circle as an argument.

**• setRadius**. A mutator function for the radius variable.

**• getRadius**. An accessor function for the radius variable.

**• getArea**. Returns the area of the circle, which is calculated as area = pi \* radius \* radius

**• getDiameter**. Returns the diameter of the circle, which is calculated as diameter = radius \* 2

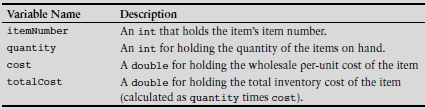
**• getCircumference**. Returns the circumference of the circle, which is calculated as

circumference = 2 \* pi \* radius

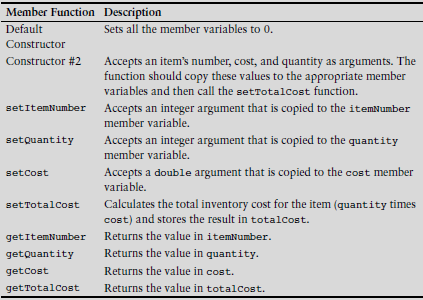
Write a program that demonstrates the Circle class by asking the user for the circle’s radius, creating a Circle object, and then reporting the circle’s area, diameter, and circumference.

**8. Inventory Class**

Design an Inventory class that can hold information and calculate data for items in a retail store’s inventory. The class should have the following *private* member variables:



The class should have the following *public* member functions:



Demonstrate the class in a driver program. *Input Validation: Do not accept negative values for item number*, *quantity*, *or cost.*

**9. Population**

In a population, the birth rate and death rate are calculated as follows:

Birth Rate = Number of Births ÷ Population

Death Rate = Number of Deaths ÷ Population

e.g. in a population of 100,000 that has 8,000 births and 6,000 deaths per year, the birth rate and death rate are:

Birth Rate = 8,000 ÷ 100,000 = 0.08

Death Rate = 6,000 ÷ 100,000 = 0.06

Design a Population class that stores a population, number of births, and number of deaths for a period of time. Member functions should return the birth rate and death rate. Implement the class in a program.

*Input Validation: Do not accept population figures less than 1*, *or birth or death numbers less than 0.*

**10. Number Array Class**

Design a class that has an array of floating-point numbers. The constructor should accept an integer argument and dynamically allocate the array to hold that many numbers. The destructor should free the memory held by the array. In addition, there should be member functions to perform the following operations:

**•** Store a number in any element of the array

**•** Retrieve a number from any element of the array

**•** Return the highest value stored in the array

**•** Return the lowest value stored in the array

**•** Return the average of all the numbers stored in the array

Demonstrate the class in a program.

**11. Payroll**

Design a PayRoll class that has data members for an employee’s hourly pay rate, number of hours worked, and total pay for the week. Write a program with an array of seven PayRoll objects. The program should ask the user for the number of hours each employee has worked and will then display the amount of gross pay each has earned. *Input Validation: Do not accept values greater than 60 for the number of hours worked.*

**12. Cash Register**

Design a CashRegister class that can be used with the InventoryItem class the CashRegister class should perform the following:

1. Ask the user for the item and quantity being purchased.

2. Get the item’s cost from the InventoryItem object.

3. Multiply the unit price times the quantity being purchased to get the purchase total.

4. Display the purchase total on the screen.

5. Subtract the quantity being purchased from the onHand variable of the InventoryItem class object.

Implement both classes in a complete program. Feel free to modify the InventoryItem class in any way necessary. *Input Validation: Do not accept a negative value for the quantity of items being purchased.*

**13. Trivia Game**

In this programming challenge you will create a simple trivia game for two players.

The program will work like this:

**•** Starting with player 1, each player gets a turn at answering five trivia questions.

(There are a total of 10 questions.) When a question is displayed, four possible answers are also displayed. Only one of the answers is correct, and if the player selects the correct answer he or she earns a point.

**•** After answers have been selected for all of the questions, the program displays the number of points earned by each player and declares the player with the highest number of points the winner.

In this program you will design a Question class to hold the data for a trivia question.

The Question class should have member variables for the following data:

**•** A trivia question **•** Possible answer #1

**•** Possible answer #2 **•** Possible answer #3

**•** Possible answer #4 **•** The number of the correct answer (1, 2, 3, or 4)

The Question class should have appropriate constructor(s), accessor, and mutator functions.

The program should create an array of 10 Question objects, one for each trivia question.

Make up your own trivia questions on the subject or subjects of your choice for the objects.

**Note:**

All students are required to submit the assignment in proper format by \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_