



National University of Computer & Emerging Sciences, Karachi
Computer Science Department
Spring 2021, Lab Manual - 04



Course Code: CL-217	Course : Object Oriented Programming Lab
Instructor(s) :	Nida Munawar, Abeer Gouhar, Romasha Khurshid, M. Fahim, Sohail Afzal, Qaiser Abbas, Ali Fatmi

Contents:

1. Introduction to Classes & Objects

INTRODUCTION TO Constructor

A. Constructor and its Types

- **Constructor** is the special type of member function in C++ classes, which are automatically invoked when an object is being created. It is special because its name is same as the class name.
- **To initialize data member of class:** In the constructor member function (which will be declared by the programmer) we can initialize the default values to the data members and they can be used further for processing.
- **To allocate memory for data member:** Constructor can also be used to declare run time memory (dynamic memory for the data members).
- Constructor has the same name as the class name. It is case sensitive.
- Constructor does not have return type.
- We can overload constructor, it means we can create more than one constructor of class.
- It must be public type.

Types of Constructors

- **Default Constructors:** Default constructor is the constructor which doesn't take any argument. It has no parameters.
- **Parameterized Constructors:** It is possible to pass arguments to constructors. Typically, these arguments help initialize an object when it is created. To create a parameterized constructor, simply add parameters to it the way you would to any other function. When you define the constructor's body, use the parameters to initialize the object.
- **Copy Constructor:** A copy constructor is a member function which initializes an object using another object of the same class. The copy constructor in C++ is used to copy data of one object to another.

B. Destructors

- A destructor is a special member function that works just opposite to constructor, unlike constructors that are used for initializing an object, destructors destroy (or delete) the object.
- Destructor function is automatically invoked when the objects are destroyed.
- It cannot be declared static or const.
- The destructor does not have arguments.
- It has no return type not even void.
- An object of a class with a Destructor cannot become a member of the union.
- A destructor should be declared in the public section of the class.
- The programmer cannot access the address of destructor.

C. Data Members & its Types

- **Data Members:** The variables which are declared in any class by using any fundamental data types (like int, char, float etc) or derived data type (like class, structure, pointer etc.) are known as Data Members.

Types of Data Members

- **Private members:** The members which are declared in private section of the class (using private access modifier) are known as private members. Private members can also be accessible within the same class in which they are declared.
- **Public members:** The members which are declared in public section of the class (using public access modifier) are known as public members. Public members can access within the class and outside of the class by using the object name of the class in which they are declared.

Sample C++ Code:**Code#1 (Default Constructors)**

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

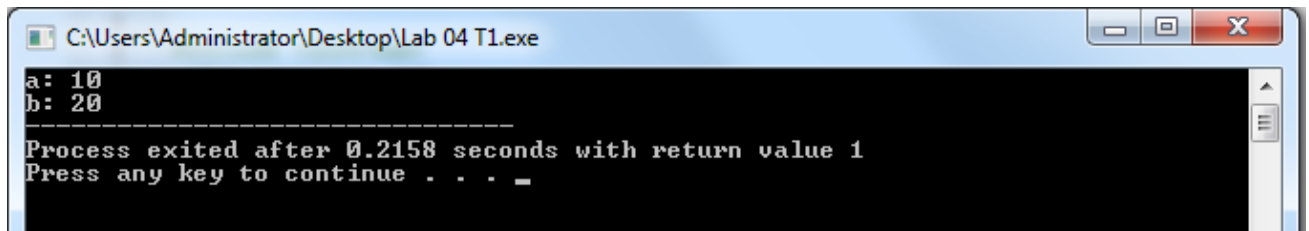
class construct
{
public:
    int a, b;

    construct()
    {
```

```
        a = 10;
        b = 20;
    }
};

int main()
{

    construct c;
    cout << "a: " << c.a << endl
         << "b: " << c.b;
    return 1;
}
```



Code#2 (Parameterized Constructors)

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

```
class Point
{
private:
```

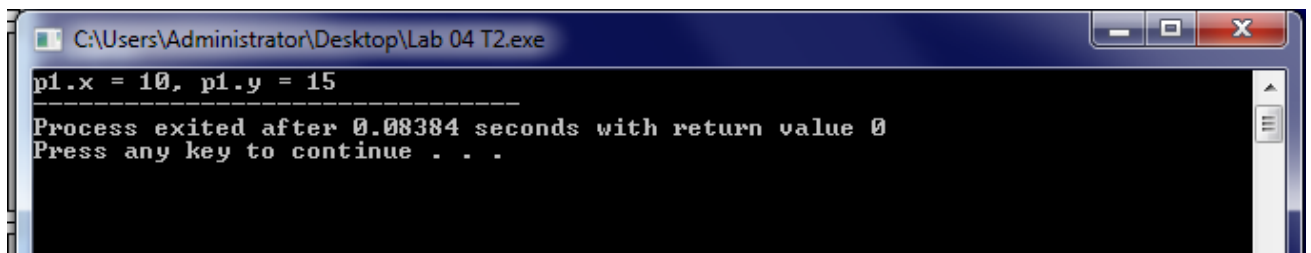
```
    int x, y;
```

```
public:
```

```
    Point(int x1, int y1)
    {
        x = x1;
        y = y1;
    }
```

```
    int getX()
    {
        return x;
```

```
    }  
    int getY()  
    {  
        return y;  
    }  
};  
  
int main()  
{  
  
    Point p1(10, 15);  
  
    cout << "p1.x = " << p1.getX() << ", p1.y = " << p1.getY();  
  
    return 0;  
}
```

**Code#3 (Copy Constructor)**

```
#include<iostream>  
#include<conio.h>  
  
using namespace std;  
  
class Example {  
  
    int a, b;  
public:  
  
    Example(int x, int y) {  
  
        a = x;  
        b = y;  
        cout << "\nIm Constructor";  
    }  
};
```

```
}
```

```
Example(const Example& obj) {
```

```
    a = obj.a;
```

```
    b = obj.b;
```

```
    cout << "\nIm Copy Constructor";
```

```
}
```

```
void Display() {
```

```
    cout << "\nValues : " << a << "\t" << b;
```

```
}
```

```
};
```

```
int main() {
```

```
    Example Object(10, 20);
```

```
    Example Object2(Object);
```

```
    Example Object3 = Object;
```

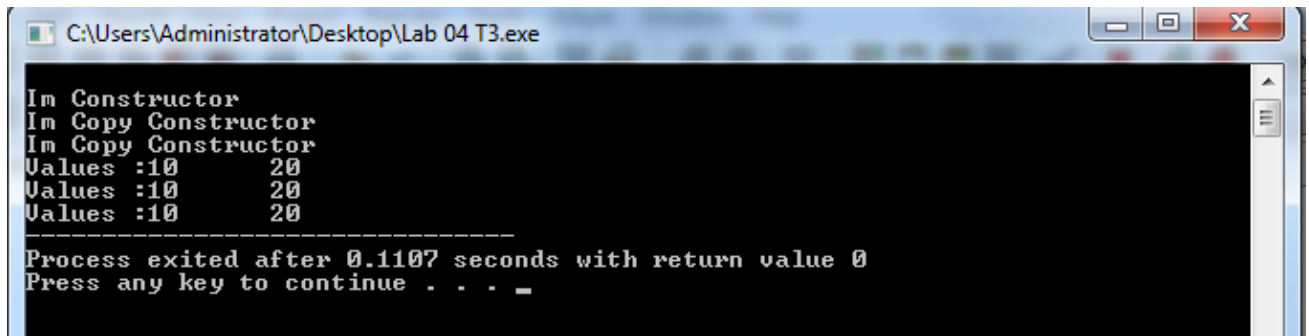
```
    Object.Display();
```

```
    Object2.Display();
```

```
    Object3.Display();
```

```
    return 0;
```

```
}
```



```

C:\Users\Administrator\Desktop\Lab 04 T3.exe
Im Constructor
Im Copy Constructor
Im Copy Constructor
Values :10      20
Values :10      20
Values :10      20
-----
Process exited after 0.1107 seconds with return value 0
Press any key to continue . . . _

```

D. Code#4 (Destructors)

```

#include <iostream>
using namespace std;
class HelloWorld{
public:

    HelloWorld(){
        cout<<"Constructor is called"<<endl;
    }

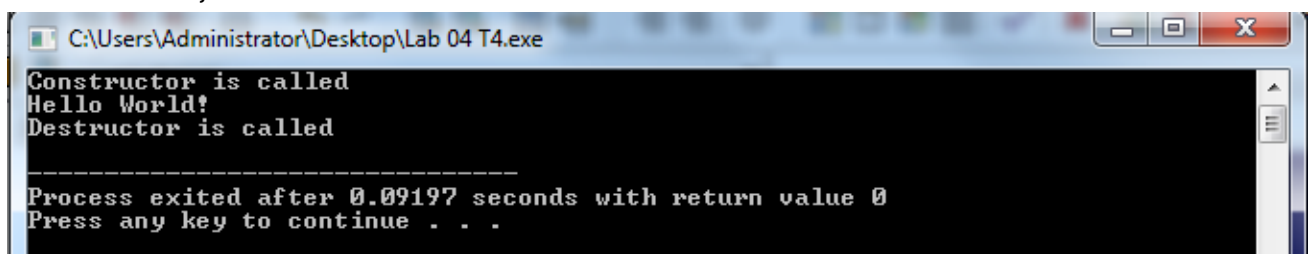
    ~HelloWorld(){
        cout<<"Destructor is called"<<endl;
    }

    void display(){
        cout<<"Hello World!"<<endl;
    }
};
int main(){

    HelloWorld obj;

    obj.display();
    return 0;
}

```



```

C:\Users\Administrator\Desktop\Lab 04 T4.exe
Constructor is called
Hello World!
Destructor is called
-----
Process exited after 0.09197 seconds with return value 0
Press any key to continue . . .

```

LAB TASKS:

Task - 01:

Your task is to create a Circle constructor that creates a circle with a radius provided by an argument. The circles constructed must have two getters `getArea()` ($\text{PI}r^2$) and `getPerimeter()` ($2\text{PI}r$) which give both respective areas and perimeter (circumference).

Task - 02:

Create an Account class that a bank might use to represent customers bank accounts. Include a data member to represent the account balance. Provide three member functions. Member function `credit` should add an amount to the current balance. Member function `debit` should withdraw money from the Account. Member function `get Balance` should return the current balance.

Task - 03:

Create A class called Invoice that a hardware store might use to represent an invoice for an item sold at the store. An Invoice should include four pieces of information as instance variables - a part number (type String), a part description (type String), a quantity of the item being purchased (type int) and a price per item (double). Your class should have a constructor that initialize the four instance variables. In addition, provide a method named `getInvoiceAmount` that calculates the invoice amount (i.e., multiplies the quantity by the price per item), then returns the amount as a double value. If the quantity is not positive, it should be set to 0. If the price per item is not positive, it should be set to 0.0.

Task - 04:

A book shop maintains the inventory of books that are being sold at the shop. The list includes details such as author, title, price, publisher and stock position. Whenever a customer wants a book, the sales person inputs the title and author and the system searches the list and displays whether it is available or not. If it is not, an appropriate message is displayed. If it is, then the system displays the book details and requests for the number of copies required. If the requested copies are available, the total cost of the requested copies is displayed; otherwise "Required copies not in stock" is displayed. Design a system using a class called books with suitable member functions and constructors.

Task - 05:

Write a class called CoffeeShop, which has three instance variables:

Name: a string (basically, of the shop)

Menu: an array of items (of type MenuItem), with each item containing the item (name of the item), type (whether a food or a drink) and price.

Orders: an empty array and seven methods:

addOrder: adds the name of the item to the end of the orders array if it exists on the menu.

Otherwise, return "This item is currently unavailable!"

fulfillOrder: if the orders array is not empty, return "The {item} is ready!". If the orders array is empty, return "All orders have been fulfilled!"

listOrders: returns the list of orders taken, otherwise, an empty array.

dueAmount: returns the total amount due for the orders taken.

cheapestItem: returns the name of the cheapest item on the menu.

drinksOnly: returns only the item names of type drink from the menu.

foodOnly: returns only the item names of type food from the menu.