Assa

NATIONAL UNIVERSITY OF COMPUTER AND EMERGING SCIENCES

CL-1004 – Object Oriented Programming Lab

Lab 04

Outline

- Default Constructor
- Parameterized Constructor
- Copy Constructor
- Destructor
- Exercise

- **Constructor**
  - A constructor is a member function of a class.
  - C++ requires a constructor call for each object that's created, which helps ensure that each object is initialized properly before it's used in a program. The constructor call occurs implicitly when the object is created.
  - This ensures that objects will always have valid data to work on
  - It has the same name of the class.
  - It must be a public member.
  - No Return Values.
  - Default constructors are called when constructors are not defined for the classes.

  Syntax

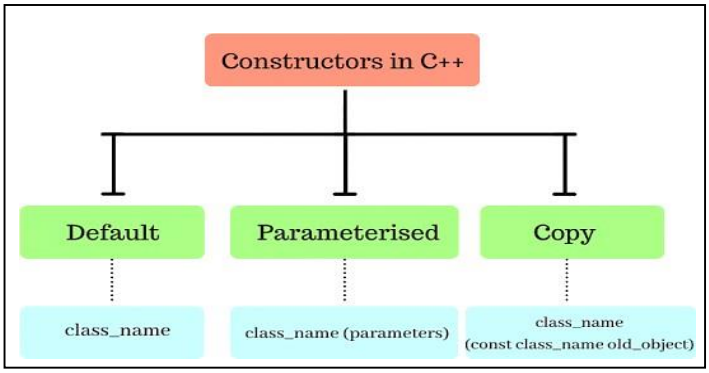
```
class Car
{
public:
      //Constructor
      Car(){

      }
};
```

How Constructors Are Different From A Normal Member Function?

Constructor	Method
Constructor name must be same as class name	Functions cannot have same name as class name.
Constructor does not have return type	Whereas functions must have.
Constructors are invoked at the time of object creation automatically and cannot be called explicitly	Functions are called explicitly using class objects.
Constructors are not inherited by subclasses as they are not members of class	Methods are inherited by subclass to provide the reusability of code.

Types of Constructors

- Default Constructor: 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.
- Copy Constructor: A copy constructor is a member function which initializes an object using another object of the same class.




```
Myclass (int x, int y) {
Int a=x;//12
Int b=y; //10
}
Int main (){
Myclass obj(12,10)
}
```

```
Myclass ( ) {
Int a=12
Int b=10
}
Int main (){
Myclass obj ()
```

1. Default Constructor

- Default constructor is the constructor which doesn't take any argument. It has no parameters.
- In this case, as soon as the object is created the constructor is called which initializes its data members.
- A default constructor is so important for initialization of object members, that even if we do not define a constructor explicitly, the compiler will provide a default constructor implicitly.

Program as example

```
#include<iostream>
#include<conio.h>
using namespace std;
class Example {
    // Variable Declaration
    int a, b;
public:
    //Constructor
    Example() {
        // Assign Values In Constructor
        a = 10;
        b = 20;
        cout << "Im Constructor\n";
    }
    void Display() {
        cout << "Values :" << a << "\t" << b;
    }
};
int main() {
    Example Object;
    // Constructor invoked.
    Object.Display();

    // Wait For Output Screen
    getch();
    return 0;
}
```

2. Parameterized Constructors

- Arguments can be passed to parameterized constructor.
- 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.
- we can also have more than one constructor in a class and that concept is called constructor overloading.

Program as example

```
#include <iostream>
using namespace std;
class PrepInsta {
private:
    int a, b;
public:
    PrepInsta(int a1, int b1)
    {
        a = a1;
        b = b1;
    }
    int getA()
    {
        return a;
    }
    int getB()
    {
        return b;
    }
};
int main()
{
    PrepInsta p1(10, 15);
    cout << "a = " << p1.getA() << ", b = " << p1.getB();
    return 0;
}
```

Program as example

lab-04-task-1.cpp

```
#include<iostream>
using namespace std;
class CoOrds
{
public:
int x1, y1;
// constructor with two arguments:
public:
CoOrds(int x, int y)
 {
    x1 = x;
    y1 = y;
 }
void display()
 {
cout<<"x = {0}, y = {1} : "<<x1<<" "<<y1<<endl;
 }
};
int main()
{
 CoOrds p1(5, 3);
p1.display();
}
```

```
C:\Users\Administrator\Documents\lab-04-task-1.exe
x = {0}, y = {1} : 5 3

--------------------------------
Process exited after 0.007043 seconds with return value 0
Press any key to continue . . .
```

- **Constructor overloading**
  - Just like other member functions, constructors can also be overloaded. Infact when you have both default and parameterized constructors defined in your class you are having Overloaded Constructors, one with no parameter and other with parameter.
  - You can have any number of Constructors in a class that differ in parameter list.

task5.cpp | task6.cpp | [*] Untitled1

```
// C++ program to illustrate
// Constructor overloading
#include <iostream>
using namespace std;

class construct
{
public:
    float area;

    // Constructor with no parameters
    construct()
    {
        area = 0;
    }
    // Constructor with two parameters
    construct(int a, int b)
    {
        area = a * b;
    }
        void disp()
    {
        cout<< area<< endl;
    }
};
int main()
{
    // Constructor Overloading
    // with two different constructors
    // of class name
    construct o;
    construct o2( 10, 20);

    o.disp();
    o2.disp();
    return 1;
}
```

Program as example

Program as example

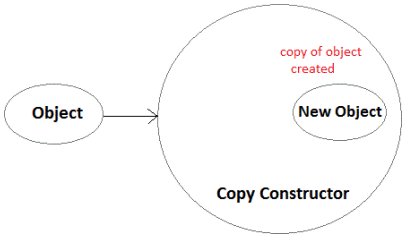
lab-04-task-1.cpp | lab-04-task-2.cpp

```
#include<iostream>
using namespace std;
class CoOrds
{
public:
int x1, y1;
// constructor with two arguments:
public:
CoOrds(int x, int y)
 {
    x1 = x;
    y1 = y;
 }
  // constructor with one arguments:
CoOrds(int cor)
 {
  x1 = cor;
  y1 = cor;
 }
void display()
 {
cout<<"x = {0}, y = {1} : "<<x1<<" "<<y1<<endl;
 }
};
int main()
{ CoOrds p(15);
p.display();
 CoOrds p1(5, 3);
p1.display();
}
```

```
C:\Users\Administrator\Documents\lab-04-task-2.exe
x = {0}, y = {1} : 15 15
x = {0}, y = {1} : 5 3

--------------------------------
Process exited after 0.01306 seconds with return value 0
Press any key to continue . . .
```

3. Copy Constructor

- A copy constructor is a member function which initializes an object using another object of the same class.
- A copy constructor has the following general function prototype:
- Syntax:
- classname (const classname &obj)
- {
- // body of constructor
- }
- The copy constructor in C++ is used to copy data of one object to another.



Types of Copy Constructors

- It is really great way of creating new object initialisations and is still widely used by programmers.
- There are two types of copy constructors which are –
  - Default Copy Constructors (Does Shallow Copying)
  - User Defined Copy Constructors (Does Deep Copying)

Default Copy Constructors

- These are in cases when a default copy constructor is created by the compiler itself. In such cases, as explained in the image, there maybe two different objects, once of which is created via default constructor called by compiler.
- The objects do not have their individual storage locations, however, are referring to common storage location

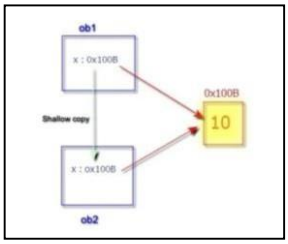
User Defined Copy Constructors

- At times when there is involvement of pointers and in some cases run time allocation of resources, like file handle, a network connection and others. We may encounter problem with direct assignment operator.

A) Example-Shallow

```
#include <iostream>
using namespace std;
class ShalloC
{
//Sample 01: Private Data Member
private:
    int * x;
public:
    //Sample 02: Constructor with single parameter
    ShalloC(int m)
    {
        x = new int;
        *x = m;
    }
    //Sample 08: Introduce Copy Constructor and perform Deep Copy
/*  ShalloC(const ShalloC& obj)
    {
        x = new int;
        *x = obj.GetX();
    }*/
    //Sample 03: Get and Set Functions
    int GetX() const
    {
        return *x;
    }
    void SetX(int m)
    {
        *x = m;
    }
    //Sample 04: Print Function
    void PrintX()
    {
        cout << "Int X=" << *x << endl;
    }
    //Sample 05: DeAllocate the heap
    ~ShalloC()
    {
        delete x;
    }
};
int main()
{
    //Sample 06: Create Object 1 and copy that to Object 2.
    //           Print the data member for both Object 1 & 2.
    ShalloC ob1(10);
    ShalloC ob2 = ob1 ;
    ob1.PrintX();
    ob2.PrintX();
    //Sample 07: Change the Data member value of Object 1
    //           And print both Object 1 and Object 2
    ob1.SetX(12);
    ob1.PrintX();
    ob2.PrintX();
}
```

```
C:\Users\Administrator\Documents\task6.exe
Int X=10
Int X=10
Int X=12
Int X=12
```



B) Example-Deep

```
[*] task6.cpp
private:
    int * x;
public:
    //Sample 02: Constructor with single parameter
    ShalloC(int m)
    {
        x = new int;
        *x = m;
    }
    //Sample 08: Introduce Copy Constructor and perform Deep Copy
    ShalloC(const ShalloC& obj)
    {
        x = new int;
        *x = obj.GetX();
    }
    //Sample 03: Get and Set Functions
    int GetX() const
    {
        return *x;
    }
    void SetX(int m)
    {
        *x = m;
    }
    //Sample 04: Print Function
    void PrintX()
    {
        cout << "Int X=" << *x << endl;
    }
    //Sample 05: DeAllocate the heap
    ~ShalloC()
    {
        delete x;
    }
};
```

```
    //Sample 05: DeAllocate the heap
    ~ShalloC()
    {
        delete x;
    }
};
int main()
{
    //Sample 06: Create Object 1 and copy that to Object 2.
    //           Print the data member for both Object 1 & 2.
    ShalloC ob1(10);
    ShalloC ob2 = ob1 ;
    ob1.PrintX();
    ob2.PrintX();
    //Sample 07: Change the Data member value of Object 1
    //           And print both Object 1 and Object 2
    ob1.SetX(12);
    ob1.PrintX();
    ob2.PrintX();
}
```

```
C:\Users\Administrator\Documents\task6.exe
Int X=10
Int X=10
Int X=12
Int X=10

--------------------------------
Process exited after 0.04887 seconds with return value 0
Press any key to continue . . .
```

Destructors

- Destructor is a member function which destructs or deletes an object.
- When is destructor called?
  A destructor function is called automatically when the object goes out of scope:
  (1) the function ends
  (2) the program ends
  (3) a block containing local variables ends
  (4) a delete operator is called

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

class Student {
public:
    int rollnumber, age;
    // Default Constructor
    Student()
    {
    cout<<"Constructor is called";
    }

    ~Student(){
    cout<<"Destructor is called";
    }
};
int main()
{
    Student S1;
    return 1;
}
```

Activity

1. A phone number, such as (021) 38768214, can be thought of as having three parts: the area code (021) the exchange (3876) and the number (8214). Write a program that uses a class Phone to store these three parts of a phone number in specific attributes. Add a constructor that accept a number and separate these elements from that number. Write a display function that display the details of the number.

   Sample Program output:

   Please enter Your No: 02134567893

   Your Area code is: 021

   Your Exchange Code is: 3456

   Your Consumer No is: 7893

2. Create a class distance that stores distance in feet and inches. Add a constructor that initializes the object with default values. There must be a function that ask user to enter distance in meters and stores accordingly. Add two functions to display the distance in meters and in feet. Add a destructor that will notify the user when an object is killed.
3. Create a class Sales with 3 private variables SaleID of type integer, ItemName of type string , and Quantity of type integer.
   a. Use a default constructor to initialize all variables with any values.
   b. Use a constructor to take user input in all variables to display data.
   c. Use a parameterized constructor to initialize the variables with values of your choice.
   d. Use copy constructor to copy the quantity of previously created object to current one.
4. Write a program in which a class named student has member variables name, roll_no, semester and section. Use a parameterized constructor to initialize the variables with your name, roll no, semester and section. Print all data calling some public method and display it.
5. Write a program using class to process shopping list for a departmental store. The list includes details such as code no, price, qty and total and perform operations like adding and deleting items from the list. The program will also print the total value of an order. You are supposed to add a constructor, destructor and other necessary functions