Unit-3
Constructor & Destructor

Constructor & Destructor

3.1 Constructor Concepts

3.2 Destructor

3.3 Parameterized Constructor,

3.4 Multiple Constructors in a Class,

3.5 Constructor with Default Arguments,

3.6 Copy Constructor,

3.7 Dynamic Constructor

3.1 Constructor

- A Constructor is a special member function whose task is to initialize the object of its class.
- It is special because it name is same as the class name.
- The constructor is invoked whenever an object of its associated class is created.

Constructor Example

Example :

```
class integer
  {
      int a,b;
    public :
          integer()  //constructor
          {
            a=0;
            b=0;
          }
  };
```


Characteristics of Constructors

- They should be declared in the public section of the class declaration.
- They are invoked automatically when the object are created.
- They cannot be virtual.
- We cannot refer their addresses.
- They cannot return values.
- They can have default arguments.

3.2 Destructor

- A destructor is used to destroy the object that have created by a constructor.
- The destructor is the member functiom whose name is same as the class name but it is procceded by a tilde(~).
- A destructor never takes argument.
- It never return any values.

Destructor Example

Example :

```
  class integer
{
 public :
 integer()
 {
   count++;
   cout<<"\n No. of object created"<<count;
 }
~integer()
 {
    cout<<"\n No. of object destroyed"<<count;
 }
};
```


3.3 Parameterized Constructor

- The constructor which can take the arguments that is called as Parameterized Constructor.
- The initial values have to be passed as arguments to the **constructor** function.
- Example :

```
class  integer
{
  int a,b;
     public :
          integer(int x,int y) //Parameterized Constructor
          {
          a=x;
            b=y;
          }
};
```

PARAMETERIZED CONSTRUCTOR

Example :

```
#include<iostream>
 #include<conio.h>
class Example
{
    int a, b; public:
    Example(int x, int y)
     {
    // Assign Values In Constructor
    a = x;
    b = y;
    cout << "Im Constructor\n";
    }
```

CONT…

```
void Display()
{
  cout << "Values :" << a << "\t" << b
   }
   };
  void main()
  {
            Example Object(10, 20); // Constructor
    invoked.
            Object.Display();
            getch(); return 0;
  }
```

Output

Im Constructor Values :10 20

Default Constructor

- It is possible to define constructors with default arguments.
- The default argument constructor can be called with either one argument or no arguments.
- When called with no arguments, it becomes a default constructor.

3.4 Multiple Constructors in a Class

```
class point
{
private:
   int x;
   int y;
public:
point( ) //default constructor
{
   x=0;
   y=0;
}
```

CONT…

```
point(int x1,int y1) //parameterized constructor
{
    x=x1;
    y=y1;
}
point(point& p) //copy constructor
{
    x=p.x;
    y=p.y;
}
void putpoint( )
{
    cout<<“(“<<x<<“,”<<y<<“ )”<<endl;
}
};
```

CONT….

```
void main( )
{
point p1;          //call default constructor
cout<<”p1= “;
p1.putpoint( );
point p2(5,7);  //call parameterized constructor
cout<<”p2= “;
p2.putpoint( );
point p3(p2);     //call copy constructor
cout<<”p3= “;
p3.putpoint( );
getch();
}
```

3.5 Constructor with Default Arguments

- A **Default constructor** is that will either have no parameters, or all the parameters have default values.
- If no constructors are available for a class, the compiler implicitly creates a default parameter less constructor without a constructor initializer and a null body.

Default Constructor Example

```
class integer
  {
       int a,b;
    public  :
         integer() //default constructor
         {
              a=10;
               b=20;
         }
          void putdata()
         {
           cout<<"\n Value  of a is:"<<a;
         }
};
```


3.6 Copy constructor

- The copy constructor is a constructor which creates an object by initializing it with an object of the same class, which has been created previously.
- It is used to initialize one object from another of the same type.
- It Copy an object to pass it as an argument to a function.
- It Copy an object to return it from a function.

EXAMPLE:

```
 class point
{
    private:
            int x;
            int y;
Public:
            point( )            //default constructor
            {
                    x=0;
                    y=0;
            }
            point(int x1,int y1)            //parameterized constructor
            {           x=x1;
                    y=y1;
            }
```

CONT…

```
        point(point&  p)              //copy  constructor
        {
                x=p.x;
                y=p.y;
        }
        void putpoint( )
        {
                cout<<" ( "<<x<<" , "<<y<<" ) "<<endl;
        }
};
void main( )
{
        point p1;
        cout<<"p1= ";
        p1.putpoint( );

        point p2(5,7);
        cout<<"p2= ";
        p2.putpoint( );
```

CONT.

```
        point p3(p2);          //Using copy constructor   // implicit call
        cout<<"p3= ";
        p3.putpoint( );

        point  p4=p3;          //Using copy constructor          //explicit call
        cout<<"p4= ";
        p4.putpoint( );

        point p5;
        p5=p4;
        cout<<"p5= ";
        p5.putpoint( );
}
```

Output:

```
p1= (0,0)
p2=(5,7)
p3=(5,7)
p4=(5,7)
p5=(5,7)
```

3.7 Dynamic Constructor

- Dynamic constructor is used to allocate the memory to the objects at the run time
- Memory is allocated at run time with the help of 'new' operator.
- By using this constructor, we can dynamically initialize the objects.

EXAMPLE

```
#include <iostream.h>
#include <conio.h>
class dyncons
{
 int * p;
 public:
 dyncons()
 {
     p=new int;
     *p=10;
 }
 dyncons(int v)
 {
     p=new int;
     *p=v;
 }
```

CONT…

```
 int dis()
 {
     return(*p);
 }
};
void main()
{
    clrscr();
    dyncons o, o1(9);
    cout<<"The value of object o's p is:";
    cout<<o.dis();
    cout<<"\nThe value of object 01's p is:"<<o1.dis();
    getch();
}
```

Output:

The value of object o's p is:10

The value of object 01's p is:9

Thank You….