Lecture 8

Constructor Overloading:

• ***A class has two or more constructor functions with the same name but different signatures are called Constructors Overloading.***

• Depending upon the type of argument, the constructors will be invoked automatically by the compiler to initialize the objects.

• **Example: Program to find simple interest using constructor overloading.**

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

class simpleinterset
{
private:
            float p, r, t, si;
public:
            simpleinterset( ) //Default constructor
            {
            }
            simpleinterset(float x, float y, float z) //Parameterized Constructor
                {
                        p = x;
                        r = y;
                        t = z;
                }
            void compute ( )
                {
                        si = (p * t * r)/100;
                        cout<<"Simple Interest is = "<< si;
                }
};

int main( )
{
    simpleinterset S1, S2(10000.0, 12.0, 2.0);
    S2.compute( );
}
```

OUTPUT:

```
Simple Interest is = 2400
```

➢ Destructors:

• ***A destructor is special member function that is executed when an object of that class is destroyed.***

• Destroying an object means, de-allocating all the resources such as memory that was allocated for the object by the constructor.
• It will have like constructor, the name same as that of the class but preceded by a tilde (~).
• The general format of destructor is as follows:

Syntax	**Example**

class Class_Name
{
public:
Class_Name( );
~ Class_Name( );
};
	class Counter
	{
	public:
	Counter( ) //Constructor
	{
	n = 0;
	}
	~Counter ( ) //Destructor
	{ }
	};

• Some of the characteristics of destructor are:

- The destructor name must have the same name as the class preceded by a tilde (~).
- The destructor cannot take arguments therefore cannot be overloaded.
- The destructor has no return type.
- There can be only one destructor in each class.
- In should have public access in the class declaration.
- The destructor cannot be inherited.

• Example: Program to illustrate the use of destructors in C++.

```
#include<iostream>
using namespace std;
class num
{
private:
   int x;
public:
      num( );
      void display( );   //Default constructor
      ~ num( );
};
```

```
num :: num( )
{
        cout<<"In Constructor: \n";
        x = 100;
}

num :: ~ num( )
{
         cout<<"In Destructor";
}

void num :: display( )
{
           cout <<"Value of X " << x <<endl;


}

int main( )
{
     num a;
     a.display( );
}
```

OUTPUT:

```
In Constructor:
Value of X = 100
In Destructor
```