Constructor and Destructor

Constructor

Constructors are special class functions which performs initialization of every object. The Compiler calls the Constructor whenever an object is created. Constructors initialize values to object members after storage is allocated to the object.

```
class A
{
 int x;
 public:
 A();  //Constructor
};
```

While defining a constructor you must remember that the name of constructor will be same as the name of the class, and constructors never have return type.

Constructor

Constructors can be defined either inside the class definition or outside class definition using class name and **scope resolution** :: operator.

```
class A
{
 int i;
 public:
 A(); //Constructor declared
};

A::A()   // Constructor definition
{
 i=1;
}
```

Types of Constructors

Constructors are of three types :

1. Default Constructor
2. Parameterized Constructor
3. Copy Constructor

Default Constructor

Default constructor is the constructor which doesn't take any argument. It has no parameter.

```
class_name ()
{ Constructor Definition };
```

Default Constructor

```
class Cube
{
int side;
public:
Cube()
  {
   side=10;
  }
};

int main()
{
Cube c;
cout << c.side;
}
```

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.

Md. Sharif Hossen
Lecturer, Dept. of ICT, Comilla University, Bangladesh

Default Constructor

```
class Cube
{
 int side;
};

int main()
{
 Cube c;
 cout << c.side;
}
```

Parameterized Constructor

These are the constructors with parameter. Using this Constructor you can provide different values to data members of different objects, by passing the appropriate values as argument.

```
class Cube
{
 int side;
 public:
 Cube(int x)
  {
   side=x;
  }
};

int main()
{
 Cube c1(10);
 Cube c2(20);
 Cube c3(30);
 cout << c1.side;
 cout << c2.side;
 cout << c3.side;
}
```

Md. Sharif Hossen
Lecturer, Dept. of ICT, Comilla University, Bangladesh

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.

```
class Student
{
 int rollno;
 string name;
 public:
 Student(int x)
 {
  rollno=x;
  name="None";
 }
 Student(int x, string str)
 {
  rollno=x ;
  name=str ;
 }
};

int main()
{
 Student A(10);
 Student B(11,"Ram");
}
```

Md. Sharif Hossen
Lecturer, Dept. of ICT, Comilla University, Bangladesh

Destructors

Destructor is a special class function which destroys the object as soon as the scope of object ends. The destructor is called automatically by the compiler when the object goes out of scope.

The syntax for destructor is same as that for the constructor, the class name is used for the name of destructor, with a tilde **~ sign** as prefix to it.

```
class A
{
 public:
 ~A();
};
```

Destructors will never have any arguments.

Example to see how Constructor and Destructor is called

```
class A {
A() {
  cout << "Constructor called";
 }

~A() {
  cout << "Destructor called";
 }
};

int main() {
 A obj1;   // Constructor Called
 int x=1
 if(x) {
   A obj2;  // Constructor Called
  }   // Destructor Called for obj2
} //  Destructor called for obj1
```