- A class **constructor** is a special member function of a class that is executed whenever we create new objects of that class.
- A constructor will have exact same name as the class and it does not have any return type at all, not even void. Constructors can be very useful for setting initial values for certain member variables.

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.

- **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.

- **Copy Constructor :** These are special type of Constructors which takes an object as argument, and is used to copy values of data members of one object into other object. We will study copy constructors in detail later.

Structure of Constructor

```
class constname
{
   Access Specifier:
      Member-Variables
      Member-Functions
   public:
      constname()
      {
         // Constructor code
      }

      ... other Variables & Functions
};
```

Example of Constructor

```
#include<iostream>
#include<conio.h>
class Example
  {
      int a,b;    // Variable Declaration
    public:
        Example() //Constructor
         {
         // Assign Values In Constructor
              a=10;
              b=20;
             cout<<"I m in Constructor\n“
         }

       void Display()
         {
         cout<<"Values : "<< a << “ \t “ <<b;
         }
  };
```

```
void main()
 {
    Example Object;
    // Constructor invoked.
    Object.Display();

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

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.

Example

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

                        Student(int x, string str)
                        {
                           rollno=x ;
                           name=str ;
                        }
   };
```

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

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();
    };
```