Constructor -

- In C++, constructor is a special method which is invoked automatically at the time of object creation. It is used to initialize the data members of new object generally.

- The constructor in C++ has the same name as class or structure.

- There can be two types of constructors in C++.
  1. Default constructor
  2. Parameterized constructor

- **Default Constructor -** A constructor which has no argument is known as default constructor. It is invoked at the time of creating object.

- **Parameterized Constructor -** A constructor which has parameters is called parameterized constructor. It is used to provide different values to distinct objects.

Example of Default Constructor -

```
#include<iostream.h>
#include<conio.h>

class Defal
{
        public:
                        int x;
                        int y;
        Defal()
        {
                x=0;
                y=0;
        }
};

void main()
{
        Defal obj;
        cout << “Default constructor”;
        cout << obj.x ;
        cout << obj.y ;
}
```

Example of Parameterized Constructor -

```
#include<iostream.h>
#include<conio.h>

class PrCons
{
   public:
      int x,y,z;

      PrCons(int a, int b)
      {
         x = a;
         y = b;
      }
};

void main()
{
   PrCons obj(10,20);

   cout<<"\nValue of a: "<<obj.x;
   cout<<"\nValue of b: "<<obj.y;
}
```

Example of Parameterized Constructor – (Type – II)

```
#include<iostream.h>
#include<conio.h>

class Rectangle
{
                int width, height;
  public:
        Rectangle (int,int);

        int area () {
                return (width*height);
        }
};
```

```
Rectangle::Rectangle (int a, int b) {
                width = a;
                height = b;
}

void main()
{
        Rectangle rect (3,4);
        Rectangle rectb (5,6);

        cout<<“rect area: “<<rect.area();
        cout<<“rectb area: “<<rectb.area() ;
}
```

Destructor -

- A destructor is used to destroy the objects that have been created by a constructor.
- Like constructor, the destructor is a member function whose name is the same as the class name but is preceded by a tilde.
- A destructor never takes any argument nor does it return any value.
- It will be invoked implicitly by the compiler upon exit from the program – or block or function as the case may be – to clean up storage that is no longer accessible.
- It is a good practice to declare destructors in a program since it releases memory space for further use.

Note - C++ destructor cannot have parameters. Moreover, modifiers can't be applied on destructors.

Example of Destructor -

```
#include<iostream.h>
#include<conio.h>

class Destruct
{
  public:
      Destruct()
      {
         cout<<"Constructor Invoked"<<endl;
      }
      ~Destruct()
      {
         cout<<"Destructor Invoked"<<endl;
      }
};
```

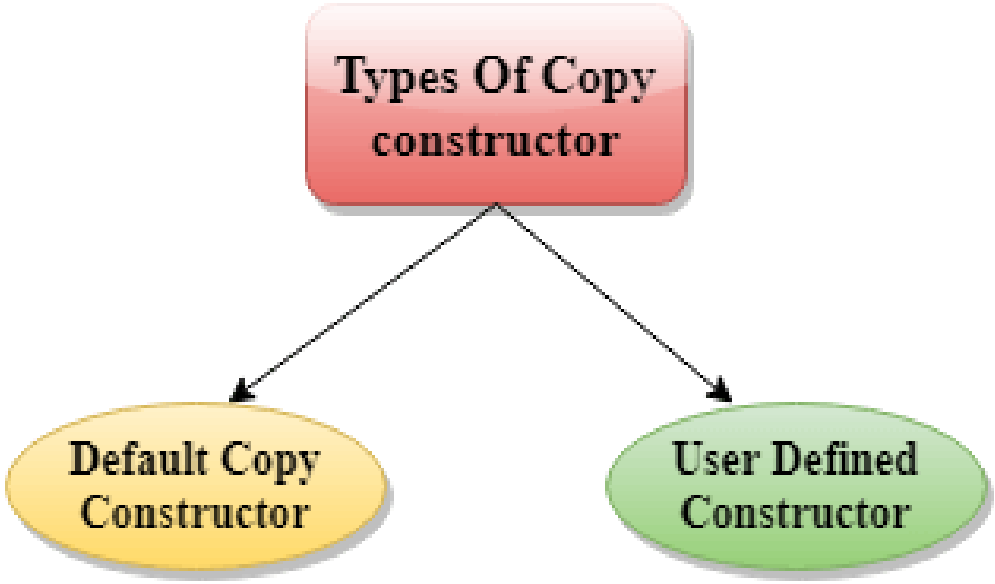
```
void main()
{
        Destruct e1;
}
```

C++ Copy Constructor –

A Copy constructor is an overloaded constructor used to declare and initialize an object from another object.

Copy Constructor is of two types -

- **Default Copy constructor -** The compiler defines the default copy constructor. If the user defines no copy constructor, compiler supplies its constructor.
- **User Defined constructor -** The programmer defines the user-defined constructor.



Explanation of Copy Constructor -

- A copy constructor is used to declare and initialize an object from another object.
- The process of initializing through a copy constructor is known as ***copy initialization***.
- A reference variable has been used as an argument to the copy constructor.
- We cannot pass the argument by value to a copy constructor.

The copy constructors are used in the following situations –

>> The initialization of an object by another object of the same class.

>> Return of objects as a function value.

>> Starting the object as by value parameters of a function.

Example of Copy Constructor -

```
#include<iostream.h>
#include<conio.h>

class Example{
                int a, b;
        public:
        Example(int x, int y){
                a = x;b = y;
        }

        Example(const Example& obj){
                a = obj.a;b = obj.b;
        }

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

```
void main()
{
  //Normal Constructor Invoked
  Example Object(10, 20);

  //Copy Constructor Invoked - Method 1
  Example Object2(Object);

  //Copy Constructor Invoked - Method 2
  Example Object3 = Object;

  Object.Display();
  Object2.Display();
  Object3.Display();

}
```

Multiple Constructors in a class -

- C++ permits to use more than one constructors in a single class -

  - **MultConstructors();** //No arguments
  - **MultConstructors(int x, int y);** // Two arguments

```
#include<iostream>
#include<conio.h>

class MultConstructors
{
public:
   int a,b,c;

   MultConstructors()
   {
      c = 0;
   }

   MultConstructors(int x, int y)
   {
      a = x;
      b = y;
   }
};

void main()
{
   MultConstructors mDObj;
   cout<<"Value is: "<<mDObj.a<<"\t"<<mDObj.b;

   MultConstructors mPObj(1,2);
   cout<<"\nValue is: "<<mPObj.a<<"\t"<<mPObj.b;

    c = (a + b);
    cout<<“Sum is: ”<<c;
}
```

Constructor with Default Arguments -

It is possible to define constructors with default arguments.

```
#include<iostream.h>
#include<conio.h>

class Sample
{
      private:
            int a, b, c;
      public:
            Sample(int x = 10, int y = 20){
                  a = x;
                  b = y;
            }
            void show(){
                  cout<<“value is: ”<<a;
                  cout<<“value is: ”<<b;
            }
};
```