## Constructor:

A constructor is a special member function whose task is to initialize the object of its class. It is special because its name is the same as the class name. the constructor is invoked whenever an object of its associated class is created. It is called constructor because it construct the values of data members of the class. Constructors must be declared in the public section. They are invoked automatically when the object are created. They do not have any return types, not even void and therefore, and they cannot return values. They cannot be inherited. Constructors can be of different types:

- Default constructor
- Parameterized constructor
- Copy constructor

## Default Constructor:

A constructor that accepts no parameters is called default constructor. If parameterized or default constructor is defined, then it is necessary to define a default constructor. e.g.

```
class rectangle
{
    private:
            int length;
            int width;
            int area;

    public:
            rectangle(void);
            void output();
};

rectangle::rectangle(void)
{
     length=5;
     width=10;
}

void rectangle::output(void)
{
     area=length*width;
     cout<<area:
}

void main()
{    rectangle r1, r2;
     r1.output();
     r2.output();
}
```

## Parameterized Constructor:

In practice it may be necessary to initialize the various data members of different objects with different values when they are created. C++ permits us to achieve this objective by passing arguments to the constructor when objects are created. The constructor that can take arguments are called parameterized constructor. e.g.

```
class rectangle
{
    private:

            int length;
            int width;
            int area;

    public:

            rectangle(int,int);
            void output();
};

rectangle::rectangle(int l, int w)
{
     length=l;
     width=w;
}

void rectangle::output(void)
{
     area=length*width;
     cout<<area:
}

void main()
{    rectangle r1(10,20);
     r1.output();
}
```

## Copy Constructor:

A constructor through which an object is initialized from another object is called copy constructor. e.g.

```
class rectangle
{
    private:

            int length;
            int width;
            int area;

    public:

            rectangle(int,int);
            rectangle(rectangle &);
            void output();

};

rectangle::rectangle(int x,int y)
{
     length=x;
     width=y;
}

rectangle::rectangle(rectangle &r3)
{
     length=r3.length;
     width=r3.width;
}

void rectangle::output(void)
{
     area=length*width;
     cout<<area:
}

void main()
{
    rectangle r1(3,5), r2(r1);

    r1.output();
    r2.output();

}
```

## Destructor:

A destructor is used to destroy the object that has been created by a constructor. Like a 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 to clean up the storage that is no longer accessible. It is a good practice to define destructor in a program since it release memory space for future use.

```
class rectangle
{
     private:

               int length;
               int width;
               int area;

     public:

              rectangle(int,int);
              ~rectangle();

};

rectangle::rectangle(int x,int y)
{
       length=x;
       width=y;
}

void rectangle::output(void)
{
       area=length*width;
       cout<<area:
}

rectangle::~rectangle()
{
       cout<< "object is destroyed";
}

void main()
{
     rectangle r1(3,5));
     r1.output()
};
```