Constructors/Destructors

Constructors are special functions used to initialize the values of the objects. It has the following specialty:

Its name is same as the name of the class.

It has no return type

It can be default or parameterized constructor.

Its syntax is :

Name_of_class()

{

//initialization statements

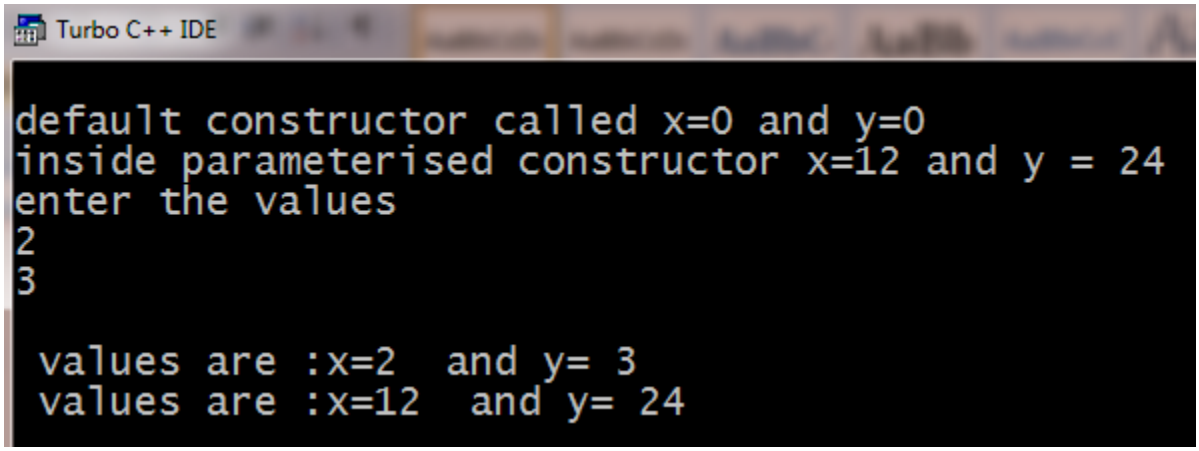
}

Like user defined functions it can be defined inside class or outside the class.

Example of default constructor

```
#include<iostream.h>
#include<conio.h>
class myClass
{
int x, y;
public:
myClass()
        {
                x=0;
                y=0;
                cout<<"\ndefault constructor called x="<<x<<" and y=" <<y<<endl;
        }
myClass(int a, int b);
void get_val()
        {
                cout<<"enter the values\n";
                cin>>x>>y;
        }
void display();
};
myClass::myClass(int a, int b)
        {
                x=a;
```

```
                y=b;
                cout<<"inside parameterised constructor x="<<x<< " and y = "<<y<<endl;
        }
void myClass :: display()
        {
                cout <<"\n values are :x="<<x <<"  and y= "<<y;
        }
void main()
{
        clrscr();
         myClass obj1, obj2(12,24);
         obj1.display();
         obj2.display();
         getch();
 }
```



Copy Constructor

A copy constructor is used to declare and initialize the value to an object from another object. That is it copies the value of on object into another object.

Its syntax is :

```
myClass (myClass &obj)

        {

                Val1=obj.val;

                Val2=obj.val;

        }
```

Example:

```
#include<iostream.h>
#include<conio.h>
class myClass
{
int x, y;
public:
myClass()
        {
                x=0;
                y=0;
                cout<<"\ndefault constructor called x="<<x<<" and y=" <<y<<endl;
        }
myClass(int a, int b);
myClass(myClass &);
void display();
void get_val()
        {
                cout<<"enter the values\n";
                cin>>x>>y;
        }
void display();
};
myClass::myClass(int a, int b)
        {
                x=a;
                y=b;
                cout<<"inside parameterised constructor x="<<x<< " and y = "<<y<<endl;
        }

myClass::myClass(myClass &obj)
        {
                x=obj.x;
                y=obj.y;
        }

void myClass :: display()
        {
                 cout <<"\n values are :x="<<x <<"  and y= "<<y;
        }
void main()
{
        clrscr();
        myClass obj1, obj2(12,24);
        myClass obj3(obj2);
```

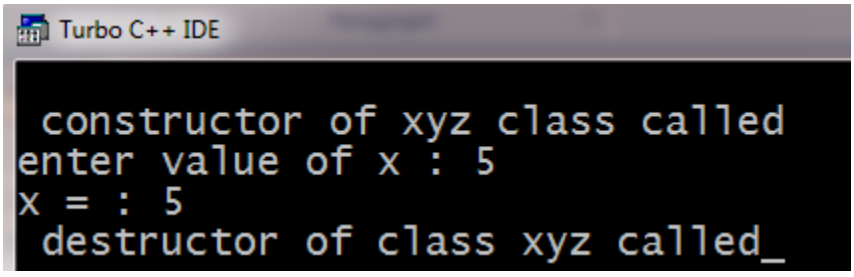
```
        obj1.get_val();
        obj1.display();
        obj2.display();
        obj3.display();
        getch();
}
```

Destructors

Destructors in C++ operate as complement of the constructor. Whereas the constructor is used to construct/ initialize the objects, destructors are used to destroy or release the memory held by the member data of the object.

Here is an example of the use of destructor

```
#include<iostream.h>
#include <conio.h>
class XYZ
{
    int x;
public:
    XYZ ()
    {
    cout<<"\n constructor of xyz class called";
    }
    //void set_data(const int=0);
    void get_data()
    {
    cout <<"\nenter value of x : ";
    cin>>x;
    }
    void disp()
    {
    cout <<"x = : "<<x;
    }
    ~XYZ()
    {
    cout<<"\n destructor of class xyz called";
    }
};
void main()
{
clrscr();
 XYZ obj;
 obj.get_data();
 obj.disp();
getch();
}
```



In the next example, we take another example where we input the name and registration number. For storing the name in an array/ or pointer variable we need to allocate a memory heap using the new operator, and the destructor uses the delete operator to delete the memory held by the data member of an object. We should remember that though the memory held by an object is released by the destructor, but that held by the data member of the object is not released. We need to use the delete operator inside the destructor.

```
#include<iostream.h>
#include<string.h>
#include <conio.h>
class XYZ
{
      int reg_no, len;
      char *std_name;

public:
      XYZ (int a, const char * const name=NULL);
      void get_data()
      {      cout<<" \nFunction get_data called";
             cout <<"\nenter value of x : ";
             cin>>reg_no;
             cout<<"\n enter name : ";
             cin>>std_name;
      }
      void disp()
      {
                    cout <<"\nReg_no = : "<<reg_no<<" and nmae = "<<std_name ;
      }
      ~XYZ();
};

XYZ::XYZ (int a, const char * const name)
  {
     cout<<"\n constructor invoked reg_no and name  is =";
     int len;
     reg_no = a;
     if(name == NULL)
     {
              std_name=NULL;
              len=0;
     }
     else
     {
             len=strlen(name);
             std_name=new char[len+1];  //dynamically allocate the separate memory block
             strcpy(std_name, name);
     }
```

```
  }
XYZ::~XYZ()
     {
               cout<<"\n destructor invoked memory held by memder deleted";
               if(std_name!=NULL)
               delete[] std_name;
     }

void main()
{
          clrscr();
          XYZ obj(101, "Ravinder");
          obj.disp();
          obj.get_data();
          obj.disp();
         getch();
}
```