

Class – 12

OOP c++

Class – 05

## Copy Constructor – 02 & Destructor

---

```
#include<iostream>
using namespace std;

class A
{
    int n;

public:

    A(int num)
    {
        cout<<"Parameterized constructor called"<<endl;
        n = num;
    }

    void display()
    {
        cout<<"n = "<<n<<endl;
    }

    A(const A& o) // pass by reference
    {
        cout<<"Copy constructor called"<<endl;
```

```

        n = o.n; // o2.n = o1.n
    }
};

```

Copy constructor makes a new object that is a copy of another existing object.

The syntax is

```

classname(const classname &obj)

```

It copies the values from one object to another.

Here `n = o.n` copies the value of `n` from object `o` into the new one.

```

void f1(A p) // pass by value
{
    p.display();
    cout<<"f1 function ends here"<<endl;
}

```

When an object is passed by value, a new copy of the object is made.

This new copy is destroyed automatically when the function ends.

```

A f2() // returns an obj of class A
{
    A t(12);
    cout<<"f2 function returned"<<endl;
    return t; // the data of obj t gets copied to a temporary
obj
}

```

When an object is returned by value, a copy constructor is called.

Here `t` is copied into a temporary object and then used to create another one.

```

int main()
{
    A o1(99);
    A o2 = o1; //invokes copy constructor

    cout<<endl;

    f1(o2);
    A o4 = f2();
    o4.display();
}

```

A o1(99) calls the parameterized constructor and sets n = 99.

A o2 = o1 calls the copy constructor and copies o1 into o2.

f1(o2) calls the copy constructor again because o2 is passed by value.

A o4 = f2() calls the copy constructor again because an object is returned by value.

Copy constructor is called when an object is copied, passed by value, or returned by value.

```

#include<iostream>
using namespace std;

```

```

class A
{
    int n;

```

```

public:

```

```

    A(int num)
    {
        cout<<"Parameterized constructor called"<<endl;
        n = num;
    }

```

```

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

void display()
{
    cout<<"n = "<<n<<endl;
}

};

```

Destructor has the same name as the class but starts with a tilde (~).

It is called automatically when an object is destroyed or goes out of scope.

```

void f1(A p) // pass by value , copy created
{
    p.display();
    cout<<"f1 function ends here"<<endl;
}

```

Passing by value makes a new copy of the object.

The destructor is called for that copy when the function ends.

```

void ff(A &q) // pass by reference , no copy created
{
    q.display();
    cout<<"ff function ends here"<<endl;
}

```

Passing by reference does not make a new copy.

The same object is used inside the function, so destructor is not called here.

```
int main()
{
    A o1(99) , o2(1);
    f1(o1);

    cout<<endl;
    ff(o1);

    cout<<"main function ends here"<<endl;
}
```

A o1(99) and A o2(1) call the parameterized constructor.

f1(o1) makes a copy of o1, and its destructor is called after f1 ends.

ff(o1) uses o1 directly without making a copy.

When main ends, destructors are called automatically in reverse order of creation first o2, then o1.