

**Don Bosco Institute of Technology, Kurla(W)**  
**Department of Electronics and Tele-Communication Engineering**  
**ECL304 - Skill Lab: C++ and Java Programming**  
**Sem III**  
**2021-22**

<b>Lab Number:</b>	<b>03</b>
<b>Student Name:</b>	SHIVAM SHARMA
<b>Roll No :</b>	E-19

**Title:**

Write a program to demonstrate default, overloaded and copy constructor and application of destructor using C++.

**Learning Objective:**

- Students will be able to write C++ program for to demonstrate default, overloaded and copy constructor and application of destructor.

**Learning Outcome:**

- Ability to execute a simple C++ and Java program with and without any inputs to the program.
- Understanding the default, overloaded and copy constructor and application of destructor.

**Theory:**

**Destructors** are usually used to deallocate memory and do other cleanup for a class object and its class members when the object is destroyed. A destructor is called for a class object when that object passes out of scope or is explicitly deleted.

A destructor takes no arguments and has no return type. Its address cannot be taken. Destructors cannot be declared const, volatile, const volatile or static. A destructor can be declared virtual or pure virtual.

If no user-defined destructor exists for a class and one is needed, the compiler implicitly declares a destructor. This implicitly declared destructor is an inline public member of its class.

The compiler will implicitly define an implicitly declared destructor when the compiler uses the destructor to destroy an object of the destructor's class type. Suppose a class A has an implicitly declared destructor.

Properties of Destructor:

**Don Bosco Institute of Technology, Kurla(W)**  
**Department of Electronics and Tele-Communication Engineering**  
**ECL304 - Skill Lab: C++ and Java Programming**  
**Sem III**  
**2021-22**

1. Destructor function is automatically invoked when the objects are destroyed.
2. It cannot be declared static or const.
3. The destructor does not have arguments.
4. It has no return type not even void.
5. An object of a class with a Destructor cannot become a member of the union.
6. A destructor should be declared in the public section of the class.
7. The programmer cannot access the address of destructor.

Copy constructor **is called when a new object is created from an existing object, as a copy of the existing object. Assignment operator is called when an already initialized object is assigned a new value from another existing object.**

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.

Uses of Copy Constructor:

- 1) When we initialize an object by another object of the same class.
- 2) When we return an object as a function value.
- 3) When the object is passed to a function as a non-reference parameter.

**Algorithm 1:**

Step 1: start  
Step 2: create a class student  
Step 3: declare students attributes like name, age etc  
Step 4: copy constructor  
Step 5: call student class  
Step 6: give input in program  
Step 7: return 0  
Step 8: end

default, overloaded and copy constructors.

**Don Bosco Institute of Technology, Kurla(W)**  
**Department of Electronics and Tele-Communication Engineering**  
**ECL304 - Skill Lab: C++ and Java Programming**  
**Sem III**  
**2021-22**

INPUT

```
#include <iostream>
using namespace std;
#include <string.h>

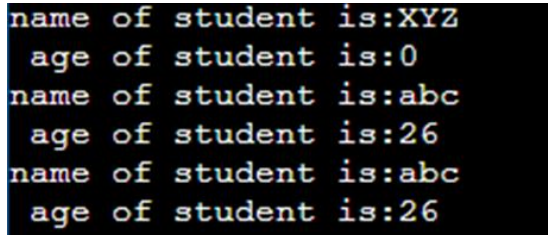
class student{
private:
    char name[20];
    int age;
public:
    student(){};
    student(char *n)
    {
        strcpy(name,n);
        age=0;
    }
    student(char *n, int a)
    {
        strcpy(name,n);
        age=a;
    }
    student(student &s)
    {
        strcpy(name,s.name);
        age=s.age;
    }
    void show();
};

void student:: show()
{
    cout<< "name of student is:"<<name<<endl;
    cout<<" age of student is:"<<age<<endl;
}

int main()
{
    student s2("XYZ");
    student s3("abc",26);
    student s4(s3);
    s2.show();
    s3.show();
    s4.show();
    return 0;
}
```

**Don Bosco Institute of Technology, Kurla(W)**  
**Department of Electronics and Tele-Communication Engineering**  
**ECL304 - Skill Lab: C++ and Java Programming**  
**Sem III**  
**2021-22**

Output:



```
name of student is:XYZ
age of student is:0
name of student is:abc
age of student is:26
name of student is:abc
age of student is:26
```

Algorithm 2:

Step 1 : start

Step 2 : create a class car

Step 3 : declare attributes of car

Step 4 : declare constructor

Step 5 : constructor definition outside the class

Step 6 : create car object and call the constructor with different values

Step 7 : return 0

Step 8 : end

## PROGRAM 2

### INPUT

```
#include <iostream>
using namespace std;
class Car {    // The class
public:       // Access specifier
    string brand; // Attribute
    string model; // Attribute
    int year;    // Attribute
    Car(string x, string y, int z); // Constructor declaration
    ~Car(){
    }
};
// Constructor definition outside the class
Car::Car(string x, string y, int z) {
    brand = x;
    model = y;
    year = z;
}
int main() {
    // Create Car objects and call the constructor with different values
    Car carObj1("BMW", "X5", 1999);
    Car carObj2("Ford", "Mustang", 1969);
    cout << carObj1.brand << " " << carObj1.model << " " << carObj1.year << "\n";
    cout << carObj2.brand << " " << carObj2.model << " " << carObj2.year << "\n";
    return 0;}
```

**Don Bosco Institute of Technology, Kurla(W)**  
**Department of Electronics and Tele-Communication Engineering**  
**ECL304 - Skill Lab: C++ and Java Programming**  
**Sem III**  
**2021-22**

**OUTPUT:**

```
BMW X5 1999
Ford Mustang 1969

...Program finished with exit code 0
Press ENTER to exit console.□
```