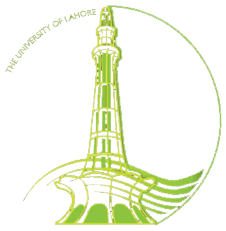

THE UNIVERSITY OF LAHORE
ISLAMABAD CAMPUS

Object Oriented Programming (CS02203)

Lab Report

Name:	Noor Baho
Registration #:	CSU-F12-116
Lab Report #:	06
Dated:	19-11-2018
Submitted To:	Mr. Usman Ahmed

The University of Lahore, Islamabad Campus
Department of Computer Science & Information Technology

Experiment # 6
Constructor and Destructor

Objective
Objective of this lab session is to reinforce programming concepts regarding to class, object constructor and destructors.

Software Tool

- **OS Windows 10**
- **Editor DEV C++**
- **Language C++**

1 Theory

1.1 Constructor

Constructors are special class functions which performs initialization of every object. The Compiler calls the Constructor whenever an object is created. Constructors iitialize values to object members after storage is allocated to the object.

```
class A
{
int x;
public:
A(); //Constructor
};
```

While defining a constructor you must remember that the name of constructor will be same as the name of the class, and constructors never have return type.
Constructors can be defined either inside the class definition or outside class definition using class name and scope resolution :: operator.

```
class A
{
int i;
public:
A(); //Constructor declared
};
A::A() // Constructor definition
{
i=1;
}
```

Types of Constructors
Constructors are of three types:

- **Default Constructor**
- **Parametrized Constructor**
- **Copy Constructor**

Default Constructor
Default constructor is the constructor which doesn't take any argument. It has no parameter.
Syntax:

```
class_name ()
{ Constructor Definition }
Example :
class Cube
{
int side;
public:
Cube()
{
side=10;
}
};
int main()
```

```
{
Cube c;
cout << c.side;
}
```

Output: 10
In this case, as soon as the object is created the constructor is called which initializes its data members.
A default constructor is so important for initialization of object members, that even if we do not define a constructor explicitly, the compiler will provide a default constructor implicitly.

```
class Cube
{

int side;
};
int main()
{
Cube c;
cout << c.side;
}
```

Output: 0
In this case, default constructor provided by the compiler will be called which will initialize the object data members to default value that will be 0 in this case.
Parameterized Consructor
These are the constructors with parameter. Using this Constructor you can provide different values to data members of different objects, by passing the appropriate values as argument.

Example :

```
class Cube
{
int side;
public:
```

```
Cube(int x)
{
side=x;
}
};
int main()
{
Cube c1(10);
Cube c2(20);
Cube c3(30);
cout << c1.side;

cout << c2.side;
cout << c3.side;
}
```

OUTPUT : 10 20 30

By using parameterized constructor in above case, we have initialized 3 objects with user defined values. We can have any number of parameters in a constructor.

Copy Constructor

These are special type of Constructors which takes an object as argument, and is used to copy values of data members of one object into other object. We will study copy constructors in detail later.

Constructor Overloading

Just like other member functions, constructors can also be overloaded. Infact when you have both default and parameterized constructors defined in your class you are having Overloaded Constructors, one with no parameter and other with parameter. You can have any number of Constructors in a class that differ in parameter list.

```
class Student
{
int rollno;
string name;
public:
Student(int x)
{
rollno=x;
```

```
name="None";
}
Student(int x, string str)
{
rollno=x ;
name=str ;
}
};
int main()

{
Student A(10);
Student B(11,"Ram");
}
```

In above case we have defined two constructors with different parameters, hence overloading the constructors. One more important thing, if you define any constructor explicitly, then the compiler will not provide default constructor and you will have to define it yourself. In the above case if we write Student S; in main(), it will lead to a compile time error, because we haven't defined default constructor, and compiler will not provide its default constructor because we have defined other parameterized constructors.

1.2 Destructor

Destructor is a special class function which destroys the object as soon as the scope of object ends. The destructor is called automatically by the compiler when the object goes out of scope. The syntax for destructor is same as that for the constructor, the class name is used for the name of destructor, with a tilde sign as prefix to it.

```
class A
{
public:
~A();
};
```

Destructors will never have any arguments.

How constructor and destructor is called

```
class A
{
A()
{
cout << "Constructor called";
}
~A()
{
cout << "Destructor called";

}
};
int main()
{
A obj1; // Constructor Called
int x=1
if(x)
{
A obj2; // Constructor Called
} // Destructor Called for obj2
} // Destructor called for obj1
```

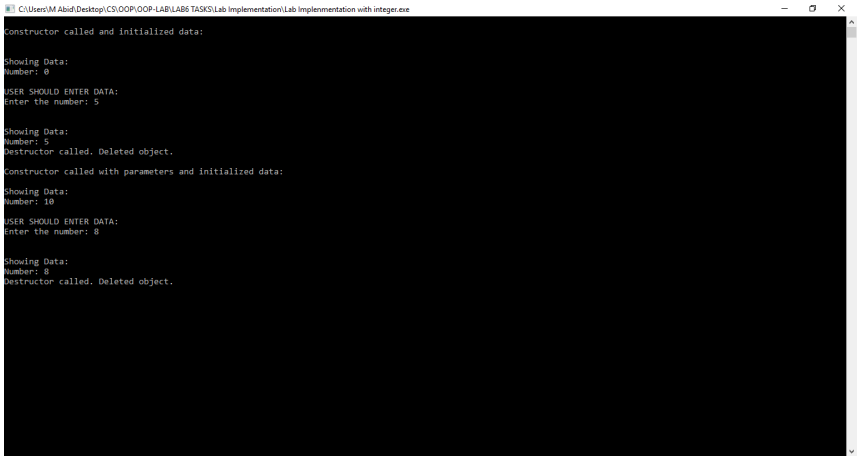

Figure 1: Use of Constructor and Destructor

2 Task

2.1 Procedure: Task 1

Implement Following:
Classes
Constructor with parameter
Constructor without parameter
Destructor
Solution:

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

class Data{
        private:
                int x;

        public:
                void set();
```

```
                void get();
                Data();
                ~Data();
                Data(int);
};
void Data::set()
{
        cout<<endl<<endl<<"USER SHOULD ENTER DATA:";
        cout<<endl<<"Enter the number: ";
        cin>>x;

}
void Data::get()
{
        cout<<endl<<endl<<"Showing Data: "<<endl;
        cout<<"Number: "<<x;
}
Data::Data()
{
        x=0;
        cout<<endl<<"Constructor called and initialized data: "<<endl;;
        get();
}
Data::Data(int y)
{
        x=y;
        cout<<endl<<"Constructor called with parameters";
        cout<<" and initialized data: ";
        get();
}
Data::~Data()
{

        cout<<endl<<"Destructor called. Deleted object."<<endl;
}

int main()
{
        Data obj1;      //obj1 created with default constructor
        obj1.set();     //obj1 setting its variables
```

```
        obj1.get();      //obj1 showing its data
        obj1.~Data();//destructor deleting obj1 data from ram

        Data obj2(10);  //obj2 created with perameters
        obj2.set();              //obj2 setting its variables
        obj2.get();              //obj2 showing its data
        obj2.~Data();   //destructor deleting obj2 data from ram

        getch();
        return 0;
}
```

Explaination: In this program created a class and its two constructor which have same name with the difference of parameters, one have and other dont have. When the object will be created for this class, according to the given parameter in the object, the constructor will be called. Destructor is releasing the memroy occupied by the previous object.

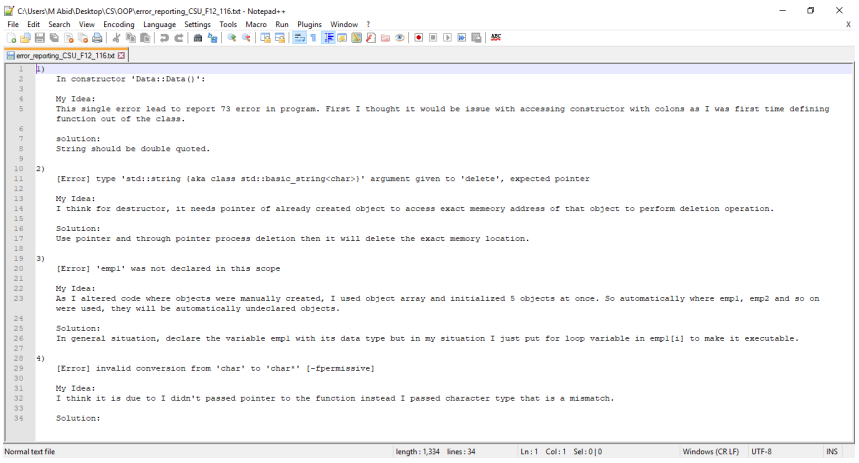

Figure 2: Error Reporting

2.2 Procedure: Task 2

Maintain a text file named "error_reporting_ROLLNUMBER.txt" in a way together the errors you are facing on daily routine. Also in one or two lines, explain your idea why this error occured and after resolving this error, also write down the solution.

Solution

The file is uploaded and can be checked.
https://github.com/miannoorbaho/OOP-LAB

3 Conclusion

We discussed and implemented the classes, constructor and destructor deeply with all of its variations. Also learned about access specifiers which allow the access of the members of class to outside in short they define their scope. The object concept of a class, how through an object of a class we can access each member and how to define multiple objects in just one line of code.