

# Chittagong University of Engineering and Technology



## *Department of Electronics and Telecommunication Engineering*

**Name of the Experiment :**

Constructor and Destructor in OOP

**Course No. :** CSE 284

**Course Title :** Object Oriented Programming

**Experiment No. :** 02

**Date of Experiment :** 08-10-2024

**Date of Submission :** 23-10-2024

Remarks	Submitted by,
	<b>Name :</b> Belayet Hossain Saimun <b>ID :</b> 2108032 <b>Level :</b> 2 <b>Term :</b> 2 <b>Group :</b> 2

## **Name of the Experiment:** Constructor and Destructor in OOP

### **Objectives:**

1. Introduce the Constructor Class in C++.
2. Define different types of constructors.
3. Learn Constructor and Destructor in C++ with the help of Examples.

**Problem Statement-01:** Suppose you have a Savings Account with an initial amount of 500 and you have to add some more amount to it. Create a class 'AddMoney' with a data member named 'amount' with an initial value of 500. Now make two constructors of this class as follows:

- without any parameter - no amount will be added to the Savings Account.
- having a parameter which is the amount that will be added to the Savings Account.

Create an object of the 'AddMoney' class and display the final amount in the Savings Account.

### **Code:**

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

class AddMoney
{
public:
    int initial_amount = 500;
    int add = 0;

    AddMoney()
    {
        int savingsAccount = add + initial_amount;
        cout << "No money added" << endl;
        cout << "So savings account amount is = " <<
            savingsAccount << endl;
    }
}
```

```

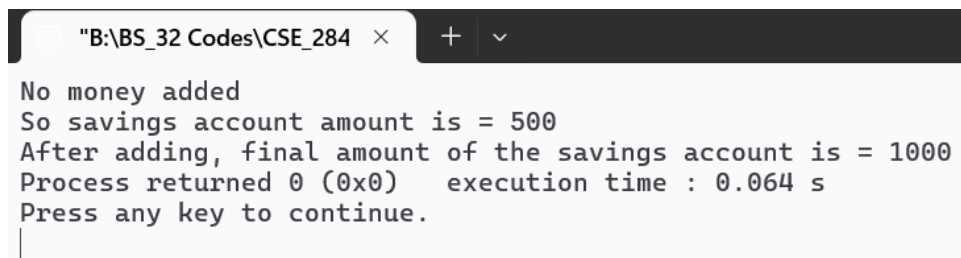
    }

    AddMoney(int addedMoney)
    {
        int savingsAccount = addedMoney + initial_amount;
        cout << "After adding, final amount of the savings
        account is = " << savingsAccount;
    }
};

int main()
{
    AddMoney();
    AddMoney add2(500);
}

```

## Output:



```

"B:\BS_32 Codes\CSE_284  ×  +  ▾
No money added
So savings account amount is = 500
After adding, final amount of the savings account is = 1000
Process returned 0 (0x0)  execution time : 0.064 s
Press any key to continue.

```

**Problem Statement-02:** Write a C++ Program to define a class Car with the following specifications:

### Private members:

**car name, model name, fuel type:** string type

**mileage:** float type

**price:** double type

### Public members:

**displaydata():** Function to display the data members on the screen.

Use Constructor (both Default and parameterized) and destructor. When no parameter is passed the default constructor will be called with the message "Default constructor has been called".

## Code:

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

class Car
{
private:
    string car_name;
    string model_name;
    string fuel_type;
    float mileage;
    double price;

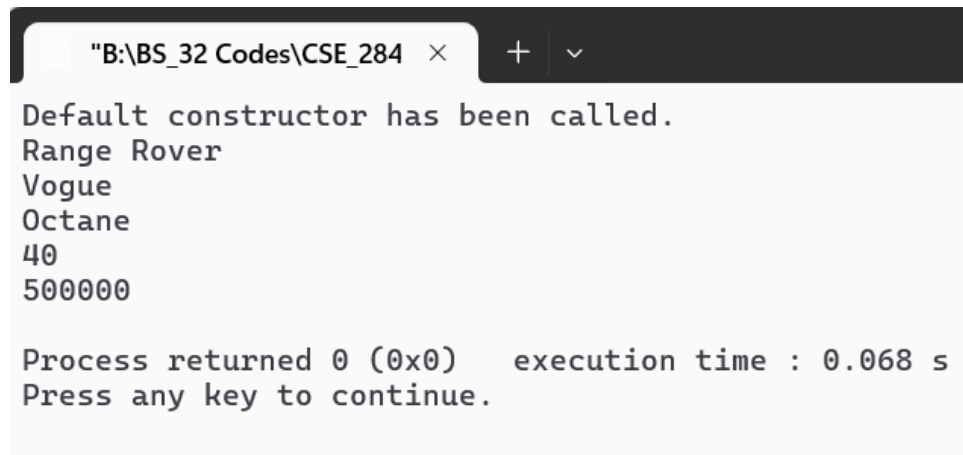
public:
    void displayData()
    {
        cout << car_name << endl;
        cout << model_name << endl;
        cout << fuel_type << endl;
        cout << mileage << endl;
        cout << price << endl;
    }

    Car()
    {
        cout << "Default constructor has been called." << endl;
    }

    Car(string car, string model, string fuel, float mile, double
prc)
    {
        car_name = car;
        model_name = model;
        fuel_type = fuel;
        mileage = mile;
        price = prc;
    }
    ~Car()
    {
    }
};

int main()
{
    Car();
    Car car1("Range Rover", "Vogue", "Octane", 40, 500000);
    car1.displayData();
}
```

## Output:

A screenshot of a terminal window with a dark theme. The title bar shows the file path "B:\BS\_32 Codes\CSE\_284" and window controls. The output text is as follows:

```
Default constructor has been called.  
Range Rover  
Vogue  
Octane  
40  
500000  
  
Process returned 0 (0x0)   execution time : 0.068 s  
Press any key to continue.
```

## Discussion:

In this experiment, I learnt the fundamental concepts of constructors and destructors in OOP using C++ through various examples. I also learnt the the uses of default constructors, parameterized constructors, and destructors. This experiment provided a foundational understanding of how constructors and destructors are essential in managing object lifecycle in C++. Also emphasized the importance of proper initialization and cleanup, which is crucial in memory management and preventing resource leaks in programming.