

CSE-284: Object Oriented Programming

Experiment 2: Constructor and Destruction in OOP

Turja Roy
ID: 2108052
Group: G-2

Objectives:

- Introduction with Constructor class in C++.
- To define different types of constructors.
- To learn Constructor and Destructor in C++.

Example 1

Write a C++ program to demonstrate the use of the default constructor.

Code

```
1 #include <bits/stdc++.h>
2 using namespace std;
3
4 class Wall {
5 private:
6     double length;
7
8 public:
9     Wall () {
10         length = 5.5;
11         cout << "Creating a wall." << endl;
12         cout << "Length: " << length << endl;
13     }
14 };
15
16 int main ()
17 {
18     Wall wall1;
19     return 0;
20 }
```

Output



```
Exp-1.cpp 1:1
3 Creating a wall.
2 Length: 5.5
1
4 [Process exited 0]
```

Figure 1: Output of Exp-1.cpp

Example 2

Write a C++ program to demonstrate the use of Parameterized Constructor.

Code

```
1 #include <bits/stdc++.h>
2 using namespace std;
3
4 class Rectangle {
5 private:
6     double length, height;
7
8 public:
9     Rectangle (double length, double height) {
10         this->length = length;
11         this->height = height;
12     }
13
14     double calculateArea () {
15         return length * height;
16     }
17 };
18
19 int main ()
20 {
21     Rectangle r1(10.5, 8.6);
22     Rectangle r2(8.5, 6.3);
23
24     cout << "Area of Rectangle 1: " << r1.calculateArea() << "\n";
25     cout << "Area of Rectangle 2: " << r2.calculateArea() << "\n";
26
27     return 0;
28 }
```

Output



```
Exp-2.cpp 1:1
3 Area of Rectangle 1: 90.3
2 Area of Rectangle 2: 53.55
1
4 [Process exited 0]
```

Figure 2: Output of Exp-2.cpp

Example 3

Write a C++ program to demonstrate the use of Copy Constructor.

Code

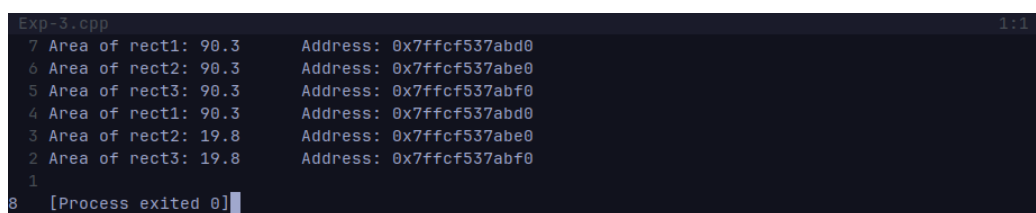
```
1 #include <bits/stdc++.h>
2 using namespace std;
3
4 class Rectangle {
5 private:
6     double length, height;
```

```

7
8 public:
9     Rectangle (double length, double height) {
10         this->length = length;
11         this->height = height;
12     }
13     Rectangle (Rectangle &rect) {
14         this->length = rect.length;
15         this->height = rect.height;
16     }
17     void setDimensions (double length, double height) {
18         this->length = length;
19         this->height = height;
20     }
21     double getArea () {
22         return length * height;
23     }
24 };
25
26 int main ()
27 {
28     Rectangle rect1(10.5, 8.6);
29     Rectangle rect2(rect1);
30     Rectangle rect3 = rect2;
31
32     cout << "Area of rect1: " << rect1.getArea() << "\t Address: " << &rect1
33     << endl;
34     cout << "Area of rect2: " << rect2.getArea() << "\t Address: " << &rect2
35     << endl;
36     cout << "Area of rect3: " << rect3.getArea() << "\t Address: " << &rect3
37     << endl;
38
39     rect2.setDimensions(5.5, 3.6);
40     rect3 = rect2;
41
42     cout << "Area of rect1: " << rect1.getArea() << "\t Address: " << &rect1
43     << endl;
44     cout << "Area of rect2: " << rect2.getArea() << "\t Address: " << &rect2
45     << endl;
46     cout << "Area of rect3: " << rect3.getArea() << "\t Address: " << &rect3
47     << endl;
48
49     return 0;
50 }

```

Output



```

Exp-3.cpp 1:1
7 Area of rect1: 90.3 Address: 0x7ffc537abd0
6 Area of rect2: 90.3 Address: 0x7ffc537abe0
5 Area of rect3: 90.3 Address: 0x7ffc537abf0
4 Area of rect1: 90.3 Address: 0x7ffc537abd0
3 Area of rect2: 19.8 Address: 0x7ffc537abe0
2 Area of rect3: 19.8 Address: 0x7ffc537abf0
1
8 [Process exited 0]

```

Figure 3: Output of Exp-3.cpp

Example 4

Write a C++ program to understand the Destructor Class in C++.

Code

```
1 #include <bits/stdc++.h>
2 using namespace std;
3
4 class Cube {
5     int side;
6 public:
7     Cube () {
8         cout << "Constructor called" << endl;
9     }
10    Cube (int side) {
11        this->side = side;
12        cout << "Constructor called" << endl;
13    }
14    ~Cube () {
15        cout << "Destructor called" << endl;
16    }
17    void setSide (int side) {
18        this->side = side;
19    }
20    double getVolume () {
21        return side * side * side;
22    }
23 };
24
25 int main ()
26 {
27     Cube c1;
28     Cube c2(10);
29
30     c1.setSide(5);
31     cout << "Volume of c1: " << c1.getVolume() << endl;
32     cout << "Volume of c2: " << c2.getVolume() << endl;
33
34     c2.setSide(15);
35     cout << "Volume of c1: " << c1.getVolume() << endl;
36     cout << "Volume of c2: " << c2.getVolume() << endl;
37
38     return 0;
39 }
```

Output



```
Exp-4.cpp 1:1
9 Constructor called
8 Constructor called
7 Volume of c1: 125
6 Volume of c2: 1000
5 Volume of c1: 125
4 Volume of c2: 3375
3 Destructor called
2 Destructor called
1
10 [Process exited 0]
```

Figure 4: Output of Exp-4.cpp

Lab Task

Write a C++ program with a class named `Student` that contains three variables `a`, `b`, `c`. If a constructor is not used, the variables will be initialized with values 1, 2, 3 respectively. If a constructor is used, the variables will be initialized with the values passed as arguments.

Code

```
1 #include <bits/stdc++.h>
2 using namespace std;
3
4 class Student {
5 private:
6     double a, b, c;
7 public:
8     Student () {
9         a = 1;
10        b = 2;
11        c = 3;
12    }
13    Student (double a, double b, double c) {
14        this->a = a;
15        this->b = b;
16        this->c = c;
17    }
18 };
19
20 int main ()
21 {
22     Student st1;
23     Student st2(4, 5, 6);
24
25     return 0;
26 }
```

Practice 1

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

```
1 #include <bits/stdc++.h>
2 using namespace std;
3
4 class AddMoney {
5 private:
6     int amount=500;
7 public:
```

```

8     AddMoney() {}
9     AddMoney(int a) {
10         amount += a;
11     }
12     void display() {
13         cout << "Final Amount: " << amount << endl;
14     }
15 };
16
17 int main ()
18 {
19     AddMoney obj1;
20     obj1.display();
21     AddMoney obj2(1000);
22     obj2.display();
23
24     return 0;
25 }

```

Output



```

Prac-1.cpp 25:1
3 Final Amount: 500
2 Final Amount: 1500
1
4 [Process exited 0]

```

Figure 5: Output of Prac-1.cpp

Practice 2

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: float 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

```

1 #include <bits/stdc++.h>
2 using namespace std;
3
4 class Car {
5 private:
6     string car_name, model_name, fuel_type;
7     float mileage, price;

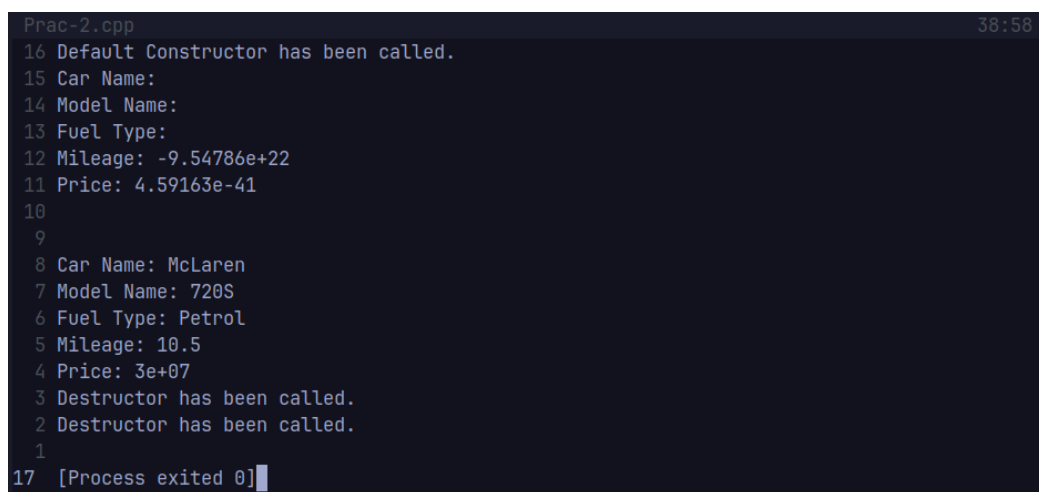
```

```

8
9 public:
10     Car() {
11         cout << "Default Constructor has been called." << endl;
12     }
13     Car (string car_name, string model_name, string fuel_type, float mileage,
14         float price) {
15         this->car_name = car_name;
16         this->model_name = model_name;
17         this->fuel_type = fuel_type;
18         this->mileage = mileage;
19         this->price = price;
20     }
21     ~Car() {
22         cout << "Destructor has been called." << endl;
23     }
24     void displaydata () {
25         cout << "Car Name: " << car_name << endl;
26         cout << "Model Name: " << model_name << endl;
27         cout << "Fuel Type: " << fuel_type << endl;
28         cout << "Mileage: " << mileage << endl;
29         cout << "Price: " << price << endl;
30     }
31 };
32
33 int main ()
34 {
35     Car car1;
36     car1.displaydata();
37
38     cout << endl << endl;
39     Car car2("McLaren", "720S", "Petrol", 10.5, 30000000);
40     car2.displaydata();
41
42     return 0;
43 }

```

Output



```

Prac-2.cpp 38:58
16 Default Constructor has been called.
15 Car Name:
14 Model Name:
13 Fuel Type:
12 Mileage: -9.54786e+22
11 Price: 4.59163e-41
10
9
8 Car Name: McLaren
7 Model Name: 720S
6 Fuel Type: Petrol
5 Mileage: 10.5
4 Price: 3e+07
3 Destructor has been called.
2 Destructor has been called.
1
17 [Process exited 0]

```

Figure 6: Output of Prac-2.cpp

Discussion

- In this lab, constructors and destructors were discussed.
- Default and parameterized constructors were implemented.
- In experiment 3, the copy constructor was used to copy the values of one object to another object.
- While using copy constructor, the addresses were checked. It can be noticed that the addresses of the two objects were different even though the values were the same.
- The destructor is always called at the end of the program. It is used to free the memory allocated to the object.
- When calling the default constructor, if no value is assigned for the variables, garbage values are assigned there.