

# Constructors and Destructors

Module 3

OOP (IT 2005)

4<sup>th</sup> Semester ECSc

# Contents

---

- Constructor
- Default constructor
- Parameterized constructor
- Copy constructor
- Constructor overloading
- Destructors



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

class rectangle {
    private:
        int h;
        int w;
    public:
        int area();
        void display();
};
```

```
int rectangle::area()
{
    return (h * w);
}
```

```
void rectangle::display()
{
    cout << "Sides are: " << h << " " << w << endl;
}
```

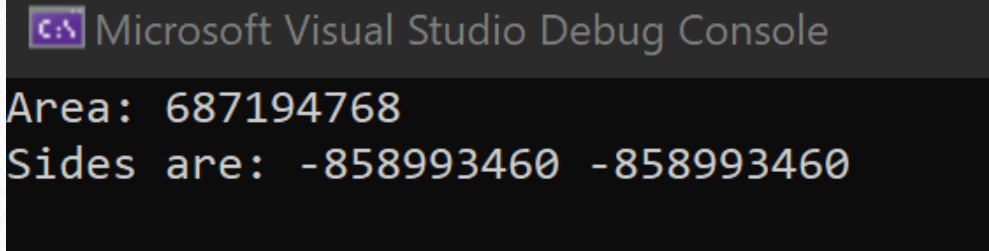
```
int main()
{
    rectangle r;
    cout << "Area: " << r.area() << endl;
    r.display();
    return 0;
}
```

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

class rectangle {
private:
    int h;
    int w;
public:
    int area();
    void display();
};
```

```
int rectangle::area()
{
    return (h * w);
}
void rectangle::display()
{
    cout << "Sides are: " << h << " " << w << endl;
}
```

```
int main()
{
    rectangle r;
    cout << "Area: " << r.area() << endl;
    r.display();
    return 0;
}
```



Microsoft Visual Studio Debug Console

```
Area: 687194768
Sides are: -858993460 -858993460
```

**A garbage value**

# Introduction

---

- A **constructor** is a special type of member function that is called automatically when an object is created.
- A constructor initializes the objects of the class.
- A constructor has the same name as that of the class.
- Constructor does not have a return type.
- A constructor must be declared in the public section.



# Types of Constructors

---

- Constructor is called by the compiler whenever the object of the class is created, it allocates the memory to the object and initializes class data members by default values or values passed by the user while creating an object.
- Three types
  - Default constructor
  - Parameterized constructor
  - Copy constructor

# Basic Syntax

---

```
class class_name
{
    private:
        data members;
    public:
        class_name(); // constructor declaration
        function declarations;
};
```

# Basic Syntax

---

```
class class_name
{
    private:
        data members;
    public:
        class_name(); // constructor declaration
        function declarations;
};
```

```
class_name :: class_name()
{
    // constructor definition
}
```



Constructor definition outside the class

**Constructors can also be defined inside the class definition.**



# Default Constructor

---

- A constructor with no parameters is known as a **default constructor**.
- See Program Example 3.1.
- A C++ program to calculate the area of a rectangle using class and default constructor.

```

#include <iostream>
using namespace std;

class rectangle {
private:
    int h;
    int w;
public:
    rectangle(); // constructor
    int area();
    void display();
};

rectangle::rectangle()
{ h = w = 0; }

int rectangle::area()
{
    return (h * w);
}

void rectangle::display()
{
    cout << "Sides are: " << h << " " << w << endl;
}

int main()
{
    rectangle r;
    cout << "Area: " << r.area() << endl;
    r.display();
    return 0;
}

```

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

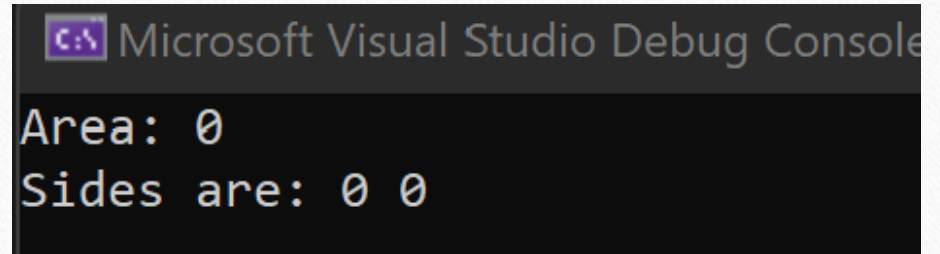
```
class rectangle {
private:
    int h;
    int w;
public:
    rectangle(); // constructor
    int area();
    void display();
};
```

```
rectangle::rectangle()
{ h = w = 0; }
```

```
int rectangle::area()
{
    return (h * w);
}
```

```
void rectangle::display()
{
    cout << "Sides are: " << h << " " << w << endl;
}
```

```
int main()
{
    rectangle r;
    cout << "Area: " << r.area() << endl;
    r.display();
    return 0;
}
```



Microsoft Visual Studio Debug Console

```
Area: 0
Sides are: 0 0
```

No garbage



# Parameterized Constructor

---

- A constructor with parameters is known as a parameterized constructor.
- This is the preferred method to initialize member data.
- See Program Example 3.2.
- A C++ program to calculate the area of a rectangle using class and parameterized constructor.

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

class rectangle {
    private:
        int h;
        int w;
    public:
        rectangle(int height, int width); // constructor
        int area();
        void display();
};

rectangle::rectangle(int height, int width)
{
    h = height;
    w = width;
}
```

```
int rectangle::area()
{
    return (h * w);
}

void rectangle::display()
{
    cout << "Sides are: " << h << " " << w << endl;
}

int main()
{
    int h1, w1;
    cout << "Enter the values: ";
    cin >> h1 >> w1;
    rectangle r(h1, w1);
    cout << "Area: " << r.area() << endl;
    r.display();
    return 0;
}
```



# Copy Constructor

---

- The copy constructor is used to copy data of one object to another.
- See Program Example 3.3.
- A C++ program to calculate the area of a rectangle using class and copy constructor.

```
#include <iostream>
using namespace std;
class rectangle {
    private:
        int h;
        int w;
    public:
        rectangle(int height, int width); // constructor
        rectangle(rectangle& rec); // copy constructor
        int area();
        void display();
};

rectangle::rectangle(int height, int width)
{
    h = height; w = width;
}

rectangle::rectangle(rectangle& rec)
{
    h = rec.h; w = rec.w;
}
```

```
int rectangle::area()
{
    return (h * w);
}
void rectangle::display()
{
    cout << "Sides are: " << h << " " << w << endl;
}
int main()
{
    int h1, w1;
    cout << "Enter the values: ";
    cin >> h1 >> w1;
    rectangle r1(h1, w1);
    rectangle r2 = r1;
    r1.display(); r2.display();
    cout << "Area of rectangle 1: " << r1.area() << endl;
    cout << "Area of rectangle 2: " << (2 * r2.area()) << endl;
    return 0;
}
```



```

int rectangle::area()
{
    return (h * w);
}
void rectangle::display()
{
    cout << "Sides are: " << h << " " << w << endl;
}
int main()
{
    int h1, w1;
    cout << "Enter the values: ";
    cin >> h1 >> w1;
    rectangle r1(h1, w1);
    rectangle r2 = r1;
    r1.display(); r2.display();
    cout << "Area of rectangle 1: " << r1.area() << endl;
    cout << "Area of rectangle 2: " << (2 * r2.area()) << endl;
    return 0;
}

```

Microsoft Visual Studio Debug Console

```

Enter the values: 4 3
Sides are: 4 3
Sides are: 4 3
Area of rectangle 1: 12
Area of rectangle 2: 24

```

# Constructor Overloading

---

- Like function overloading, constructors can be overloaded.
- When a class has multiple constructors, they are called overloaded constructors.
- See Program Example 3.4
- Write a C++ program that uses an overloaded constructor.

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

class Person {
private:
    string first_name;
    string middle_name;
    string last_name;
    int age;
public:
    Person(); // default constructor
    Person(int a); // constructor with 1 argument
    Person(string s1, int a); // constructor with 2 arguments
    Person(string s2); // constructor with 1 argument
    Person(string s1, string s2, string s3, int a); // constructor with 4 arguments
    void display();
};
```



```
Person::Person()  
{  
    age = 0;  
    first_name = " ";  
    middle_name = " ";  
    last_name = " ";  
}
```

```
Person::Person(int a)  
{  
    age = a;  
    first_name = " ";  
    middle_name = " ";  
    last_name = " ";  
}
```

```
Person::Person(string s1, int a)  
{  
    age = a;  
    first_name = s1;  
    middle_name = " ";  
    last_name = " ";  
}
```

```
Person::Person(string s2)  
{  
    age = 0;  
    first_name = s2;  
    middle_name = " ";  
    last_name = " ";  
}
```

```
Person::Person(string s1, string s2, string s3, int a)
{
    age = a;
    first_name = s1;
    middle_name = s2;
    last_name = s3;
}

void Person::display()
{
    cout << "Name: " << first_name << " " <<
    middle_name << " " << last_name << endl;
    cout << "Age: " << age << endl;
}
```

```
int main()
{
    Person P1;
    Person P2(55);
    Person P3("John");
    Person P4("John", "K.", "Smith", 55);
    P1.display();
    P2.display();
    P3.display();
    P4.display();
    return 0;
}
```



# Destructors

---

- Like a constructor, a destructor is also a member function that is automatically invoked.
- The job of the destructor is to delete the object.
- When a destructor is invoked, the memory allocated for an object is de-allocated or released.

# Important Note on Destructors

---

- The name of the destructor is also the same as that of the class.
- The destructor's name is preceded by a symbol  $\sim$ .
- A destructor must be declared in the public section.
- A destructor does not take any argument and hence cannot be overloaded.
- A destructor does not return any value.

# Program Example 3.5

---

- A C++ program to demonstrate the use of destructors.



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

class rectangle {
    private:
        int h;
        int w;
    public:
        rectangle(int height, int width); // constructor
        int area();
        ~rectangle(); // destructor
};

rectangle::rectangle(int height, int width)
{
    h = height;
    w = width;
    cout << "Sides are: " << h << " " << w << endl;
}
```

```
rectangle::~rectangle()
```



Destructor definition

```
{  
    cout << "Delete " << h << " " << w << endl;  
}
```

```
int rectangle::area()
```

```
{  
    return (h * w);  
}
```

```
int main()  
{
```

```
    int h1, w1, h2, w2;
```

```
    cout << "Enter the values: ";
```

```
    cin >> h1 >> w1;
```

```
    cout << "Enter the values: ";
```

```
    cin >> h2 >> w2;
```

```
    rectangle r1(h1, w1);
```

```
    rectangle r2(h2, w2);
```

```
    cout << "Area: " << r1.area() << endl;
```

```
    cout << "Area: " << r2.area() << endl;
```

```
    return 0;
```

```
}
```

Observe the order of destructor call

```
Microsoft Visual Studio Debug Console  
Enter the values: 5 8  
Enter the values: 6 9  
Sides are: 5 8  
Sides are: 6 9  
Area: 40  
Area: 54  
Delete 6 9  
Delete 5 8
```



Observe the order of destructor call

```
Microsoft Visual Studio Debug Console
Enter the values: 5 8
Enter the values: 6 9
Sides are: 5 8
Sides are: 6 9
Area: 40
Area: 54
Delete 6 9
Delete 5 8
```



# Exercise 3.1

---

- Define a class to represent a bank account. Include the following data members:

Name of the account holder

Account number

Balance amount in the account

Write constructors to update the balance amount in the account after every deposit and withdraw. Also display the balance amount after each deposit or withdraw.

## Exercise 3.2

---

- Write a C++ program that calculates the commission of 3 salespersons. For every salesperson, a target is defined. If a salesperson sells products less than the target assigned, he gets no commission. If a salesperson completes the target, he/she gets a 10% commission on the extra sales made; if he/she sells beyond the target, he/she gets a 25% commission on extra sales. The program must record the sales of each month made by the salesman. The commission is calculated at the end of the year. Use class and constructor to design the program.



## Exercise 3.3

- A class has 3 students. Each student must read 3 subjects. Write a C++ program to read the marks obtained by each student in each subject. Also calculate the total marks obtained by each student and average marks obtained in each subject. Apply the concept of constructor and destructor in your code.

Student ID	Subject 1 (100)	Subject 2 (100)	Subject 3 (100)
Student 1	75	80	55
Student 2	80	75	63
Student 3	70	64	42

## Exercise 3.4

---

- A book shop maintains the inventory of books that are being sold at the shop. The list includes details such as author, title, price, publisher and stock position. Whenever a customer wants a book, the sales person enters the title and author and the system searches the list and displays whether it is available or not. If it is not, an appropriate message is displayed. If it is, then the system displays the book details and requests for the number of copies required. If the requested copies are available, the total cost of the requested copies is displayed; otherwise “Required copies not in stock” is displayed. Write a C++ program consisting of a class called book with suitable member functions and constructors.



# Constructors and Destructors

Module 3

OOP (IT 2005)

4<sup>th</sup> Semester ECSc