

LAB No 12

Constructor overloading

In C++, there are multiple constructors in a class under the same name but a different list of arguments. This concept of **constructor overloading in C++** is quite similar to function overloading. Usually, you should create more than one constructor in a class to initialize member variables differently for objects. The criteria to behave constructors differently is to have a different number of parameters or different positioning or different data types for parameters. Constructors that firmly create objects using a single class and return a new instance of the class are abstracted by the industry, which creates objects but can do different ways using different classes or different allocation schemes, such as object pools

LAB Objectives

- Construct programs using multiple constructors

Apparatus

- Computer System

Required Software

You can choose any one of the software from the list given below

- Visual Studio
- Dev C++
- Codeblocks

Lab Outcomes

By the end of this experiment, student will have basic understanding of constructor overloading and its working.

Example 12.1

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

class calculate
{
public:
    float perimeter;

    calculate (); // constructor without parameter
    {
        Perimeter=0;
    }
}
```

```

    }
    calculate (float a, float b, float c) // constructor with parameters
    {
        perimeter=a+b+c;
    }
void output()
{
    cout<<perimeter<<endl;
}
};

int main ()
{
    calculate out ;
    cout<< "constructor with no parameter"<<endl;
    out.output();
    calculate out2(15,20.25) ;

    cout<< "constructor with three parameter"<<endl;
    out2.output();
    return 0;
}

```

There are 2 constructors of class “calculate”:

1. A default Constructor (without any parameters)
2. The Constructor with three-paramete

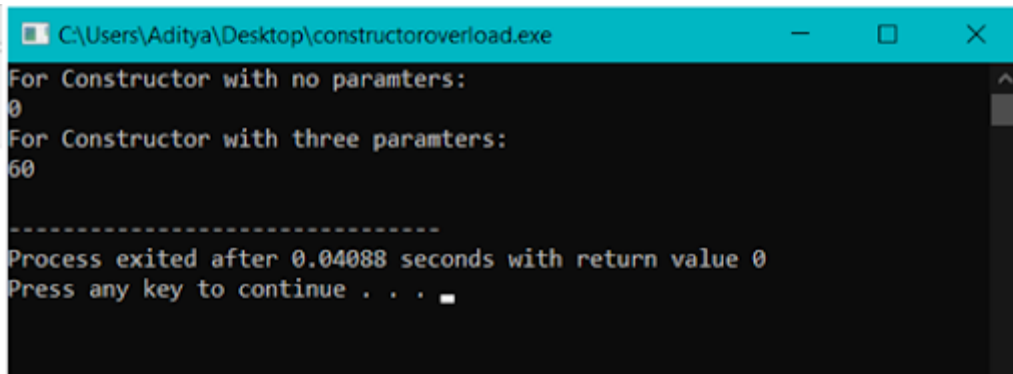
And in the main() there are two objects created.

1. out:

When it is created, it will automatically invoke the default constructor with no parameters. This is because while creating an object, there is no parameter passed. So, it matches the definition of first(Default Constructor). This will assign 0 values to all three variables for that object of the main class.

2. out2:

When it is created, it will automatically invoke the constructor with 3 parameters. This is because while creating an object, only 1 parameter is passed. So, it matches the definition of the Second Constructor. This will assign 3 values (passed as parameters) to the variable for that object of the main class.



Overloaded constructors have the same name (name of the class) but the different number of arguments. Depending upon the number and type of arguments passed, the corresponding constructor is called

Example 12.2

```
/ C++ program to illustrate
// Constructor overloading
#include <iostream>
using namespace std;

class construct
{
public:
    float area;

    // Constructor with no parameters
    construct()
    {
        area = 0;
    }

    // Constructor with two parameters
    construct(int a, int b)
    {
        area = a * b;
    }

    void disp()
    {
        cout<< area<< endl;
    }
};

int main()
{
    // Constructor Overloading
    // with two different constructors
    // of class name
    construct o;
    construct o2( 10, 20);

    o.disp();
    o2.disp();
    return 1;
}
```

```
}
```

Output

0

200

Example 12.3

A boy has his money deposited \$1000, \$1500 and \$2000 in banks-Bank A, Bank B and Bank C respectively. We have to print the money deposited by him in a particular bank.

Create a class 'Bank' with a function 'getBalance' which returns 0. Make its three subclasses named 'BankA', 'BankB' and 'BankC' with a function with the same name 'getBalance' which returns the amount deposited in that particular bank. Call the function 'getBalance' by the object of each of the three banks

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

class Bank
{
public:
    int getBalance()
    {
        return 0;
    }
};

class BankA : public Bank
{
    int amount;
public:
    BankA(int a)
    {
        amount = a;
    }

    int getBalance()
    {
        return amount;
    }
};

class BankB : public Bank
{
    int amount;
public:
    BankB(int a)
    {
        amount = a;
    }
}
```

```

    int getBalance()
    {
        return amount;
    }
};

class BankC : public Bank
{
    int amount;
public:
    BankC(int a)
    {
        amount = a;
    }

    int getBalance()
    {
        return amount;
    }
};

int main()
{
    BankA a(1000);
    BankB b(1500);
    BankC c(2000);
    cout << a.getBalance() << endl;
    cout << b.getBalance() << endl;
    cout << c.getBalance() << endl;
    return 0;
}

```

Example 12.4

```

// C++ program to demonstrate constructor overloading
#include <iostream>
using namespace std;

class Person {
private:
    int age;

public:
    // 1. Constructor with no arguments
    Person() {
        age = 20;
    }

    // 2. Constructor with an argument
    Person(int a) {

```

```

    age = a;
}

int getAge() {
    return age;
}
};

int main() {
    Person person1, person2(45);

    cout << "Person1 Age = " << person1.getAge() << endl;
    cout << "Person2 Age = " << person2.getAge() << endl;

    return 0;
}

```

Output

```

Person1 Age = 20
Person2 Age = 45

```

In this program, we have created a class **Person** that has a single variable **age**.

We have also defined two constructors **Person()** and **Person(int a)**

When the object **person1** is created, the first constructor is called because we have not passed any argument. This constructor initializes **age** to 20.

When **person2** is created, the second constructor is called since we have passed 45 as an argument. This constructor initializes **age** to 45.

Example 12.5

```

// C++ program to demonstrate constructor overloading
#include <iostream>
using namespace std;

class Room {
private:
    double length;
    double breadth;

public:
    // 1. Constructor with no arguments
    Room() {
        length = 6.9;
    }
}

```

```

        breadth = 4.2;
    }

    // 2. Constructor with two arguments
    Room(double l, double b) {
        length = l;
        breadth = b;
    }
    // 3. Constructor with one argument
    Room(double len) {
        length = len;
        breadth = 7.2;
    }

    double calculateArea() {
        return length * breadth;
    }
};

int main() {
    Room room1, room2(8.2, 6.6), room3(8.2);

    cout << "When no argument is passed: " << endl;
    cout << "Area of room = " << room1.calculateArea() << endl;

    cout << "\nWhen (8.2, 6.6) is passed." << endl;
    cout << "Area of room = " << room2.calculateArea() << endl;

    cout << "\nWhen breadth is fixed to 7.2 and (8.2) is passed:" << endl;
    cout << "Area of room = " << room3.calculateArea() << endl;

    return 0;
}

```

Output

```

When no argument is passed:
Area of room = 28.98

```

```

When (8.2, 6.6) is passed.
Area of room = 54.12

```

```

When breadth is fixed to 7.2 and (8.2) is passed:
Area of room = 59.04

```

When room1 is created, the first constructor is called. length is initialized to 6.9 and breadth is initialized to 4.2. When room2 is created, the second constructor is called. We have passed the arguments 8.2 and 6.6. length is initialized to the first argument 8.2 and breadth is initialized to 6.6. When room3 is created, the third constructor is called. We have passed one argument 8.2. length is initialized to the argument 8.2. breadth is initialized to the 7.2 by default.

LAB Review Questions

Question 1

Solve all lab exercise questions and verify the output by your lab instructor

Question 2

Show the output of a program to print the names of students by creating a Student class. If no name is passed while creating an object of the Student class, then the name should be "Unknown", otherwise the name should be equal to the String value passed while creating the object of the Student class.

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

class Student
{
    string name;
public:
    Student(string s)
    {
        name = s;
    }
    Student()
    {
        name = "Unknown";
    }
    void print_name()
    {
        cout << name << endl;
    }
};

int main()
{
    Student s1("Jhon");
    Student s2;
    s1.print_name();
    s2.print_name();
    return 0;
}
```

Question 3

Create a class named 'Rectangle' with two data members- length and breadth and a function to calculate the area which is 'length*breadth'. The class has three constructors which are :

- 1 - having no parameter - values of both length and breadth are assigned zero.
 - 2 - having two numbers as parameters - the two numbers are assigned as length and breadth respectively.
 - 3 - having one number as parameter - both length and breadth are assigned that number.
- Now, create objects of the 'Rectangle' class having none, one and two parameters and print their areas.

Question 4

Create a class 'Student' with three data members which are name, age and address. The constructor of the class assigns default values to name as "unknown", age as '0' and address as "not available". It has two functions with the same name 'setInfo'. First function has two parameters for name and age and assigns the same whereas the second function takes has three parameters which are assigned to name, age and address respectively. Print the name, age and address of 10 students.

Hint - Use array of objects