

Don Bosco Institute of Technology, Kurla(W)
Department of Electronics and Tele-Communication Engineering
ECL304 - Skill Lab: C++ and Java Programming
Sem III
2021-22

Lab Number:	5
Student Name:	Pratham Koturwar
Roll No :	24

Title:

To perform Operator Overloading using C++ for

- adding 2 complex numbers
- adding matrices

Learning Objective:

- Students will be able to perform user-defined overloading of built-in operators.

Learning Outcome:

- Understanding the overloading concept on built-in operators.

Course Outcome:

ECL304.2	Comprehend building blocks of OOPs language, inheritance, package and interfaces
-----------------	--

Theory:

Explain about operator overloading with respect to:

- constructor,
- methods and
- operators.

Operator overloading provides the ability to a mathematical operator to perform other manipulation operations other than its original operation . Operator overloading is one of the most valuable concepts introduced by C++ language . C++ frequently uses user-defined data types such as classes and structures that are a combination of one or more basic data types. User-defined data types created from class or structure are nothing but a combination of one or more variables of basic data types. The compiler knows how to perform various operations using operators for the built-in types; however, for the objectsthose are instance of the class, the operation routine must be defined by the programmer.

For example, in traditional programming languages the operators such as +, -, <=, >=, etc.

can be used only with basic data types such as int or float. The operator +

Faculty: Ms. Deepali Kayande

Don Bosco Institute of Technology, Kurla(W)
Department of Electronics and Tele-Communication Engineering
ECL304 - Skill Lab: C++ and Java Programming
Sem III
2021-22

(plus) can be used to perform addition of two variables, but the same is not applicable for objects. The compiler cannot perform addition of two objects. The compiler would throw an error if addition of two objects is carried out. The compiler must be made aware of the addition process of two objects. When an expression including operation with objects is encountered, a compiler searches for the definition of the operator, in which a code is written to perform an operation with two objects. Thus, to perform an operation with objects we need to redefine the definition of various operators. For example, for addition of objects A and B, we need to define operator + (plus). Redefining the operator plus does not change its natural meaning. It can be used for both variables of built-in data type and objects of user-defined data type and this is called as operator overloading .

Method overloading is the process of overloading the method that has the same name but different parameters. C++ provides this method of overloading features. Method overloading allows users to use the same name to another method, but the parameters passed to the methods should be different. The return type of methods can be the same or different. 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.

Algorithm:	1.Start 2. Define functions for get_element(), display(), and overload the '+' operator. 3. Take user input for matrices. 4. Decide on two variables of the Matrix type.
-------------------	---

Don Bosco Institute of Technology, Kurla(W)
Department of Electronics and Tele-Communication Engineering
ECL304 - Skill Lab: C++ and Java Programming
Sem III
2021-22

	<p>5. Use the <code>get_element()</code> function to receive the matrix</p> <p>6. Use the <code>display_element()</code> function to display the matrices.</p> <p>7. Add them using the overloaded '+' operator.</p> <p>8. Print the result.</p>
Program:	<pre># skill_lab_with_OOPM #include<iostream> using namespace std; class matrices { int a[2][2]; int b[2][2]; int c[2][2]; public: void get_elements(); //take numbers from user matrices operator +(matrices m2);</pre>

Don Bosco Institute of Technology, Kurla(W)
Department of Electronics and Tele-Communication Engineering
ECL304 - Skill Lab: C++ and Java Programming
Sem III
2021-22

```
//operator overloading

void display();    //print the result

};

//functions outside class, using scope resolution

void matrices::get_elements()

{

    cout<<"enter the elements";

    for(int i=0;i<2;i++)  //for row

    {

        for(int j=0;j<2;j++)  //for columns

            cin>>a[i][j];

    }
```

Don Bosco Institute of Technology, Kurla(W)
Department of Electronics and Tele-Communication Engineering
ECL304 - Skill Lab: C++ and Java Programming
Sem III
2021-22

```
}

void matrices:: display()

{

    for(int i=0;i<2;i++)

    {

        for(int j=0;j<2;j++)

            cout<<a[i][j]<<" ";

        cout<<endl;

    }

}

matrices matrices::operator+(matrices m2)

{
```

Don Bosco Institute of Technology, Kurla(W)
Department of Electronics and Tele-Communication Engineering
ECL304 - Skill Lab: C++ and Java Programming
Sem III
2021-22

```
        matrices m3;

        for(int i=0;i<2;i++)

        {

                for(int j=0;j<2;j++)

                        m3.a[i][j]=a[i][j]+m2.a[i][j];

        }

        return(m3);

}

int main()

{

        matrices ob1,ob2;

        ob1.get_elements();

        ob2.get_elements();
```

Don Bosco Institute of Technology, Kurla(W)
Department of Electronics and Tele-Communication Engineering
ECL304 - Skill Lab: C++ and Java Programming
Sem III
2021-22

	<pre>cout<<"\nMatrix 1:\n"; ob1.display(); cout<<"\nMatrix 2:\n"; ob2.display(); ob1=ob1+ob2; cout<<"\nResult:\n"; ob1.display(); }</pre> <p>https://github.com/prathamkoturwar/skill_lab_with_OOPM /blob/52df1d60e6ccedf3db2c500e344842ab864f3b0e/24_labs-5.1</p>
Input given:	<p>M1=345</p> <p>M2=432</p>

Don Bosco Institute of Technology, Kurla(W)
Department of Electronics and Tele-Communication Engineering
ECL304 - Skill Lab: C++ and Java Programming
Sem III
2021-22

Output
Screenshot:

```
enter the elements2
3
4
5
enter the elements5
4
3
2

Matrix 1:
2 3
4 5

Matrix 2:
5 4
3 2

Result:
7 7
7 7

-----
Process exited after 16.27 seconds with return value 0
Press any key to continue . . .
```

Algorithm:

Step 1 - Start

Step 2 - Int real , imag

Step 3 - real = r , imag = i

Step 4 - Define overload * operator

Step 5 - Use temporary variable with name 'temp'

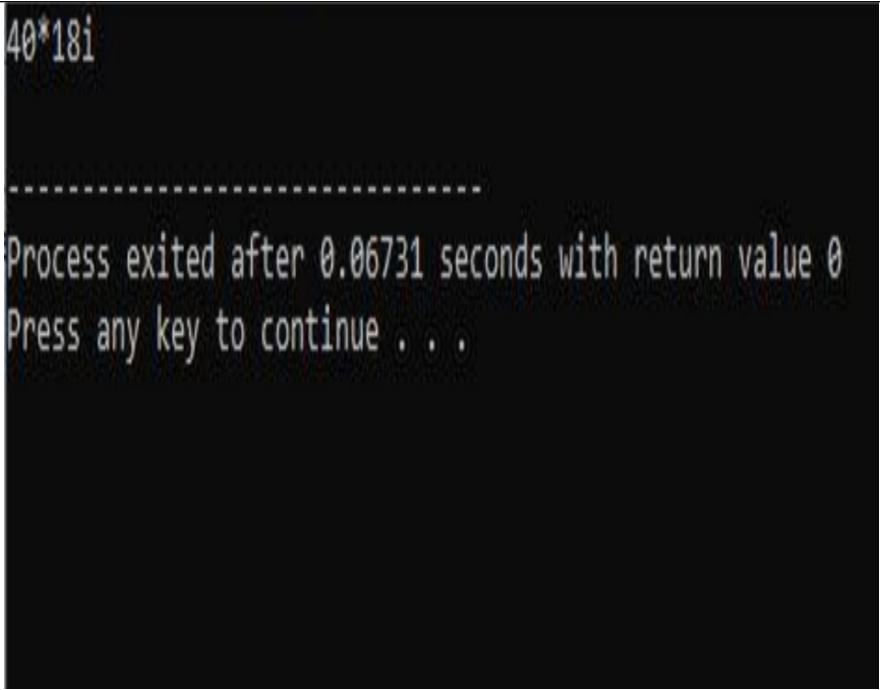
Don Bosco Institute of Technology, Kurla(W)
Department of Electronics and Tele-Communication Engineering
ECL304 - Skill Lab: C++ and Java Programming
Sem III
2021-22

	<p>Step 6 - temp.real = real*c.real temp.imag = imag*c.imag</p> <p>Step 7 = Enter the values of complexNo</p> <p>STEP 8 = Multiply them using overload * operator</p> <p>Step 9 = Display the caluculated value</p> <p>Step 10 = Stop</p>
Program:	<pre>#include<iostream> using namespace std ; class complexno { public : int real, imag; complexno() { real = 0; imag = 0; } complexno(int r, int i) { real = r; imag = i;</pre>

Don Bosco Institute of Technology, Kurla(W)
 Department of Electronics and Tele-Communication Engineering
 ECL304 - Skill Lab: C++ and Java Programming
 Sem III
 2021-22

	<pre> } void display() { cout << real << "*" << imag << "i" << endl; } complexno operator *(complexno c) { complexno temp; temp.real = real * c.real; temp.imag = imag * c.imag; return temp; } }; int main() { complexno c3; complexno c1(8,3); complexno c2(5,6); c3 = c1*c2; c3.display(); return 0; } https://github.com/prathamkoturwar/skill_lab_with_OOPM /blob/52df1d60e6ccedf3db2c500e344842ab864f3b0e/24_labs-5.2 </pre>
Input given:	8,3

Don Bosco Institute of Technology, Kurla(W)
Department of Electronics and Tele-Communication Engineering
ECL304 - Skill Lab: C++ and Java Programming
Sem III
2021-22

	5,6
Output screen shot:	 <pre>40*18i ----- Process exited after 0.06731 seconds with return value 0 Press any key to continue . . .</pre>