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:	2
Student Name:	Siddhant Kedar
Roll No :	E-05

Title: To perform operator overloading using C++ for Multiplying 2 complex numbers.

Learning Objective:

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

Learning Outcome:

- Understanding the overloading concept of built in operators.

Theory:

Operator overloading is a compile-time polymorphism in which the operator is overloaded to provide the special meaning to the user-defined data type. Operator overloading is used to overload or redefines most of the operators available in C++. It is used to perform the operation on the user-defined data type. For example, C++ provides the ability to add the variables of the user-defined data type that is applied to the built-in data types.

Operators that cannot be overloaded are:

1) Scope operator (::)

2) Size of

3) member selector(.)

4) member pointer selector(*)

5) ternary operator(?:)

Algorithm:

1) Start

2) Creating class named complex

3) Declaring attributes real and img

4) Taking data from user through get_elements()

5) Operator overloading function to overload “*”

6) Defining methods outside class

7) Creating an object of class using main function

8) Calling the method using object of class

9) Display the result

10) End

Program:

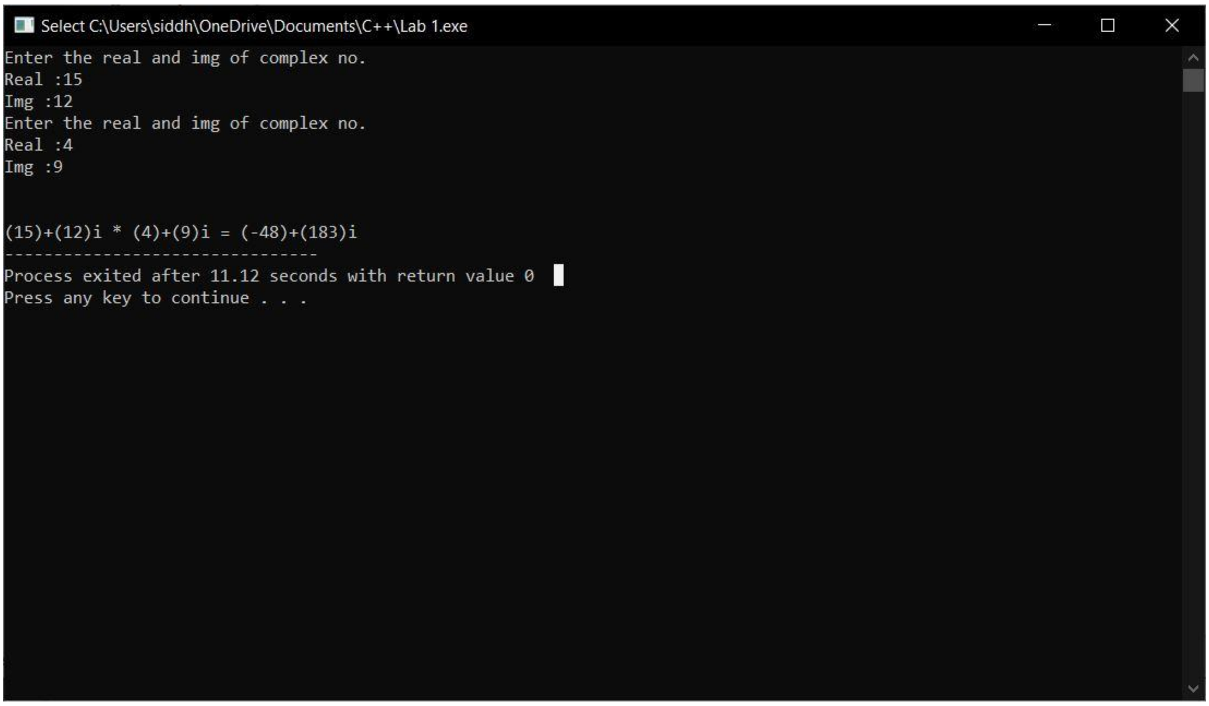
```
#include <iostream>
using namespace std;
class complex
{
    float real;
    float img;

public:
    void get_elements();
    complex operator*(complex c1);
    void display();
};

void complex::get_elements()
{
    cout << "Enter the real and img of complex no.\n";
    cout << "Real :";
    cin >> real;
    cout << "Img :";
    cin >> img;
}

void complex::display()
{
    cout << "(" << real << ")"
         << "+"
         << "(" << img << ")"
         << "i";
}

complex complex::operator*(complex c1)
{
    complex mul;
    mul.real = ((real * c1.real) - (img * c1.img));
    mul.img = ((real * c1.img) + (c1.real * img));
    return (mul);
}

int main()
{
    complex obj1, obj2, obj3;
    obj1.get_elements();
    obj2.get_elements();
    obj3 = obj1 * obj2;

    cout << "\n\n";
    obj1.display();
    cout << " * ";
    obj2.display();
    cout << " = ";
    obj3.display();
}
```

Input Given:

Real and imaginary part of complex number: 15 and 12

Real and imaginary part of complex number: 4 and 9

Output:



```
Enter the real and img of complex no.
Real :15
Img :12
Enter the real and img of complex no.
Real :4
Img :9


(15)+(12)i * (4)+(9)i = (-48)+(183)i
--------------------------------
Process exited after 11.12 seconds with return value 0
Press any key to continue . . .
```