**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:** | **Raveena Pitale** |
| **Roll No :** | **27** |

**Title:**

To perform Operator Overloading using C++ for

- Multiplying 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 : Constructor overloading is a concept in which one class can have multiple constructors with different parameters. The main thing to note here is that the constructors will run according to the arguments for example if a program consists of 3 constructors with 0, 1, and 2 arguments, so if we pass 1 argument to the constructor the compiler will automatically run the constructor which is taking 1 argument.

- methods : 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.

- Operators : In C++, it can add special features to the functionality and behaviour of already existing operators like athematic and other operations. The mechanism of giving special meaning to an operator is known as operator overloading. For example, we can overload an operator '+' in a class like string to concatenate two strings by just using +.

**Algorithm :**

1-Start

2-Creating class of name complex

3-Declaring attributes- real , img

4-Declaring methods- 1)get_elements()-to take input from user

2)display()- to print the result

5-Operator overloading function to overload "*""+"----for performing operation

6-Defining methods outside the class

7-Creating an objects of class in main function

8-Calling the methods using object of class

9-Displaying the result

10-End

**Program :**

```
# include<iostream>
using namespace std;
class complex
{
   float real;
   float img;
   public:
        void get_elements();                    //take numbers from user
        complex operator *(complex c1);      //operator overloading
         void display();                          //print the result
};

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;
```

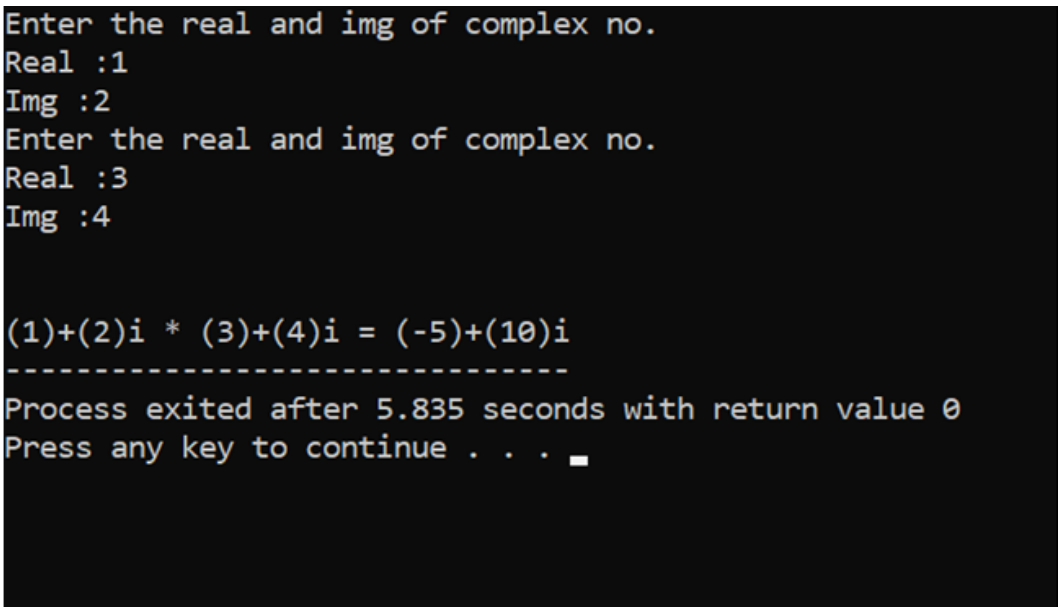
```
    cout<<"\n\n";
    obj1.display();
    cout<<" * ";
    obj2.display();
    cout<<" = ";
    obj3.display();
}
```

**Input given :**

1 number = 1+2i

2 number = 3+4i

**Output :**



**Algorithm :**

1-Start

2-Creating class of name matrices

3-Declaring a[2][2],b[2][2],c[2][2]

4-Declaring methods- 1)get_elements()-to take input from user

2)display()- to print the result

5-Operator overloading function to overload "+"----for performing operation

7-Creating an objects of class in main function

8-Calling the methods using object of class

9-Displaying the result

10-End

**Program :**

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

class matrices
{
    public:
    //Declaring attributes
    int a[2][2];
    int b[2][2];
    int c[2][2];
    //Declaring Methods
    void get_elements()          //To take input from user
    {
        cout<<"Enter the elements";
        for(int i=0;i<2;i++)
```

```
        {
                for(int j=0;j<2;j++)
                {
                        cin>>a[i][j];
                }

        }
}
matrices operator +(matrices m2)      //To overload '*'
{
        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);
}
void display()                        //To print the result
{
        for(int i=0;i<2;i++)
        {
                for(int j=0;j<2;j++)
                {
                        cout<<a[i][j]<<" ";

                }
                cout<<endl;
```

```
            }

        }

};

   int main()
   {
        matrices ob1,ob2;                //Creating object
        ob1.get_elements();              //Calling method
        ob2.get_elements();              //Calling method
        cout<<"\n Matrix 1:\n";
        ob1.display();
        cout<<"\n Matrix 2:\n";
        ob2.display();
        ob1=ob1+ob2;
        cout<<"\n Result : \n";
        ob1.display();

        }
```

**Input given :**

1 Matrix : 1 2
3 4

2 Matrix : 3 4
5 6

**Output :**


Enter the elements1
2
3
4
Enter the elements3
4
5
6
Matrix 1:
1 2
3 4
Matrix 2:
3 4
5 6
Result :
4 6
8 10
Process exited after 7.498 seconds with return value 0
Press any key to continue . . .