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:	**Umer Rafique Shaikh**
Roll No :	**44**

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:

As there is a concept of function overloading, similarly constructor overloading is applied. When we overload a constructor for more than a purpose it is called constructor overloading. The declaration is the same as the class name but as they are constructors, there is no return type. The criteria to overload a constructor is to differ the number of arguments or the type of arguments

· 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:

Additional 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 +.

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

1. Multiplying 2 complex numbers.

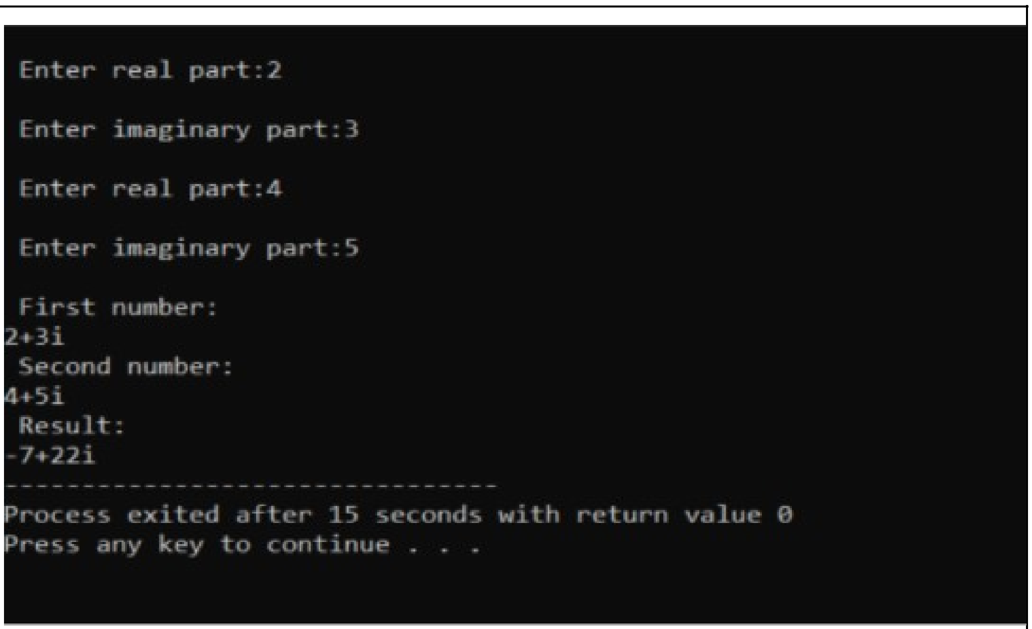
Algorithm :	STEP1 : start STEP2 : creating class complexno STEP3 : declaring attributes real(r) and imaginary (i) STEP4 : declaring methods (i)get elements () (ii) display () STEP5 : operator overloading STEP6 : creating object of class in main functions STEP7 : calling methods using objects of class STEP8 : display results STEP9 :end

Program:	//write a c++ program to overload the * operator so that it can multiply two complex numbers.
	
	#include<iostream>
	using namespace std;
	
	class complexno
	{
	 private:
	 int r,i;
	
	        public:
	                void get_elements();
	         complexno operator *(complexno c);
	         void display();
	};
	void complexno::get_elements()
	{

```
 cout<<"\n Enter real part:";
 cin>>r;
 cout<<"\n Enter imaginary part:";
 cin>>i;
}

complexno complexno::operator *(complexno s)
//(a+ib)*(c+id)=ac+i(ad)+i(bc)-bd {
        int a,b,c,d;
        a=r;
        b=i;
        c=s.r;
        d=s.i;
        int v1,v2,v3,v4;
        v1=a*c;
        v2=-1*b*d;
        v3=a*d;
        v4=b*c;
        s.r=v1+v2;
        s.i=v3+v4;
        return s;
}
void complexno::display()
{
 if (i>0)
 {
 cout<<"\n"<<r<<"+"<<i<<"i";
 }
        else if(i<0)
        {
```

	```
				cout<<"\n"<<r<<""<<i<<"i";
			}
	}
	int main()
	{
			complexno o1,o2,o3;;
			o1.get_elements();
			o2.get_elements();
			o3=o1*o2;
	 cout<<"\n First number:";
			o1.display();
			cout<<"\n Second number:";
			o2.display();
			cout<<"\n Result:";
			o3.display();
	}
	```
Input given:	Real part:2 Imaginary part:3 Real part:4 Imaginary part:5

Output Screenshot:	

```
 Enter real part:2

 Enter imaginary part:3

 Enter real part:4

 Enter imaginary part:5

 First number:
2+3i
 Second number:
4+5i
 Result:
-7+22i
--------------------------------
Process exited after 15 seconds with return value 0
Press any key to continue . . .
```

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

2.Adding matrices

Algorithm :	Step 1 : start Step 2 :creating class matrices Step 3 :declaring a[2][2] , b[2][2] , c[2][2] Step 4 :declaring methods (i) get elements () (ii) display () Step 5 :operator overloading to overload "+" Step 6 :creating objects of class in main function Step 7 : calling methods using object of class Step 8 : result display Step 9 :stop

Program:

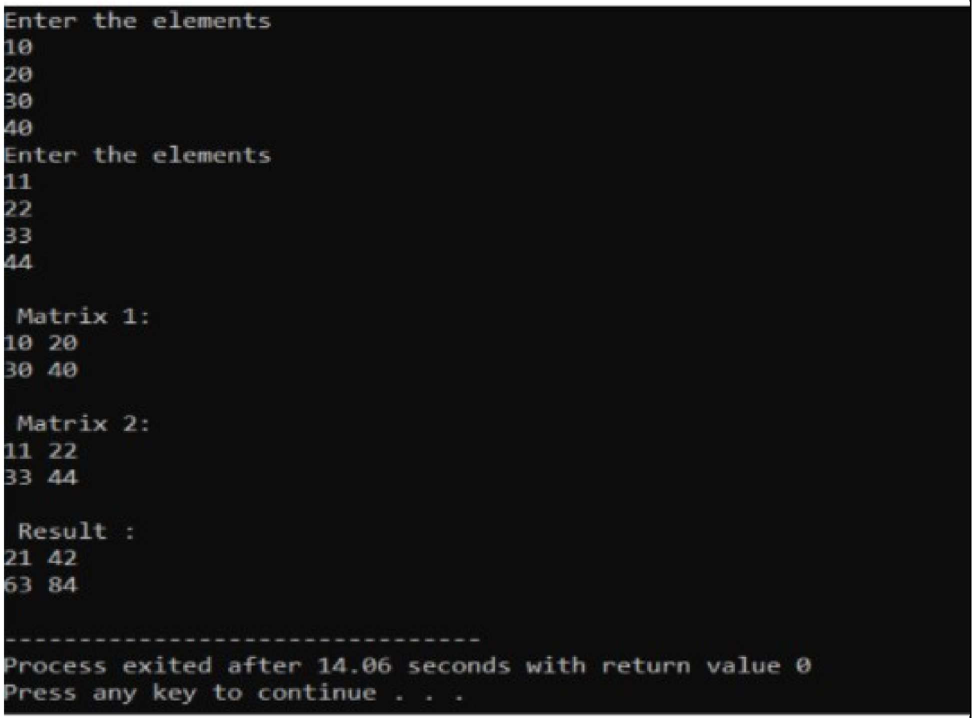
```
#include<iostream>
using namespace std;


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\n";
                        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:	ELEMENTS OF 1 MATRIX: 10 20 30 40 ELEMENTS OF 2 MATRIX: 11 22 33 44
Output Screenshot :	

```
Enter the elements
10
20
30
40
Enter the elements
11
22
33
44

 Matrix 1:
10 20
30 40

 Matrix 2:
11 22
33 44

 Result :
21 42
63 84

--------------------------------
Process exited after 14.06 seconds with return value 0
Press any key to continue . . .
```