

**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**

<b>Lab Number:</b>	<b>5</b>
<b>Student Name:</b>	<b>Rakshita Khantwal</b>
<b>Roll No :</b>	<b>26</b>

**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:**

<b>ECL304.2</b>	Comprehend building blocks of OOPs language, inheritance, package and interfaces
-----------------	--

**Theory:**

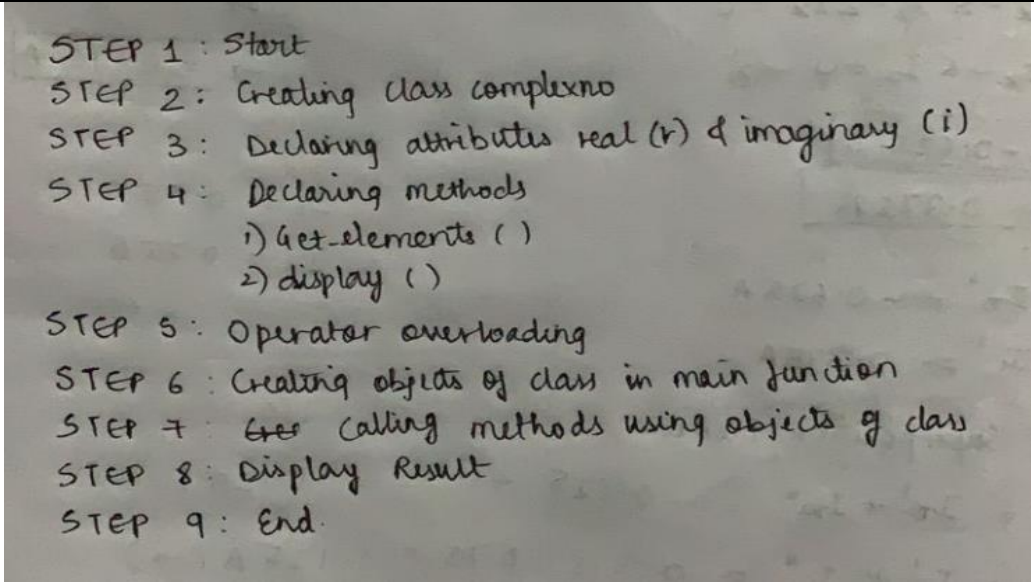
Explain about operator overloading with respect to:

- **Constructor:**  
As there is a concept of function overloading, similarly constructor overloading is applied. When we overload a constructor 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:**  
Addition special features to the functionality and behaviour of already existing operators like arithmetic 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 +.

**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**

**1. Multiplying 2 complex numbers.**

<b>Algorithm :</b>	 <p>STEP 1 : Start STEP 2 : Creating class complexno STEP 3 : Declaring attributes real (r) &amp; imaginary (i) STEP 4 : Declaring methods 1) get_elements () 2) display () STEP 5 : Operator overloading STEP 6 : Creating objects of class in main function STEP 7 : <del>Get</del> Calling methods using objects of class STEP 8 : Display Result STEP 9 : End.</p>
<b>Program:</b>	<pre>//write a c++ program to overload the * operator so that it can multiply two complex numbers.  #include&lt;iostream&gt; using namespace std;  class complexno { private:     int r,i;  public:     void get_elements();     complexno operator *(complexno c);     void display(); };  void complexno::get_elements() {</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**

```
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";
    }

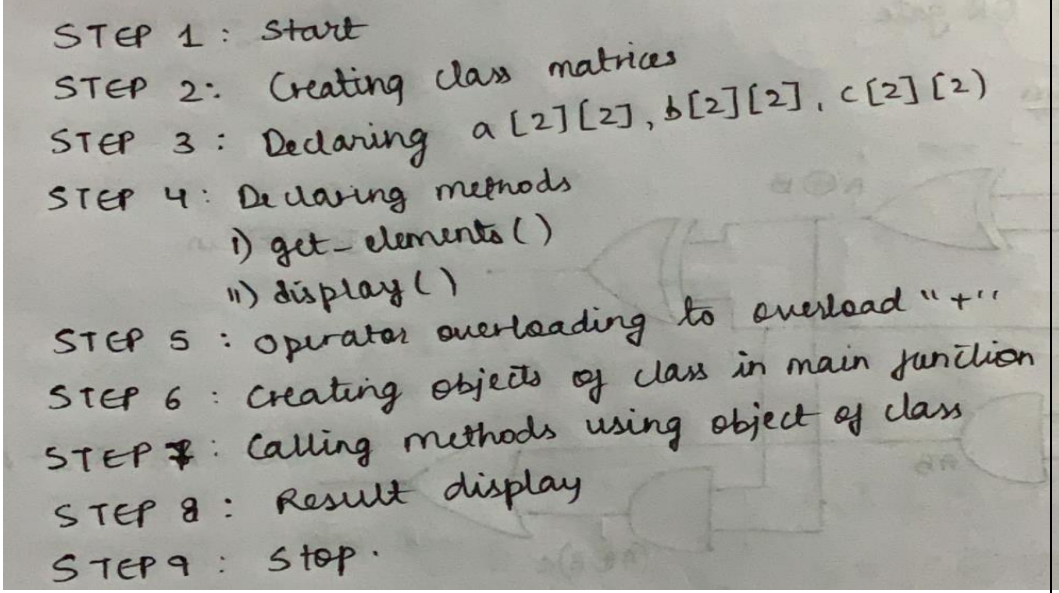
    else if(i<0)
    {
```

**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&lt;&lt;"\n"&lt;&lt;r&lt;&lt;" "&lt;&lt;i&lt;&lt;"i";      }  }  int main() {      complexno o1,o2,o3;;      o1.get_elements();     o2.get_elements();      o3=o1*o2;      cout&lt;&lt;"\n First number:";      o1.display();      cout&lt;&lt;"\n Second number:";      o2.display();      cout&lt;&lt;"\n Result:";      o3.display();  } </pre>
<b>Input given:</b>	<p>Real part:2</p> <p>Imaginary part:3</p> <p>Real part:4</p> <p>Imaginary part:5</p>
<b>Output Screenshot:</b>	 <p>The screenshot shows a Windows command prompt window titled "C:\Users\khant\Downloads\complex no multip.exe". The program prompts the user to enter the real and imaginary parts of two complex numbers. The first number has a real part of 2 and an imaginary part of 3. The second number has a real part of 4 and an imaginary part of 5. The program then displays the first number as 2+3i, the second number as 4+5i, and the result of their multiplication as -7+22i. The program exits after 15 seconds with a return value of 0.</p> <pre> C:\Users\khant\Downloads\complex no multip.exe  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 . . . </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**

**2. Adding matrices**

<b>Algorithm</b> :	 <p>STEP 1 : Start STEP 2 : Creating class matrices STEP 3 : Declaring a[2][2], b[2][2], c[2][2] STEP 4 : Declaring methods 1) get_elements() 2) display() STEP 5 : Operator overloading to overload "+" STEP 6 : Creating objects of class in main junction STEP 7 : Calling methods using object of class STEP 8 : Result display STEP 9 : Stop.</p>
<b>Program:</b>	<pre>#include&lt;iostream&gt;  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&lt;&lt;"Enter the elements\n";             for(int i=0;i&lt;2;i++)             {                 for(int j=0;j&lt;2;j++)                 {</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**

```
cin>>a[i][j];

    }

}

}

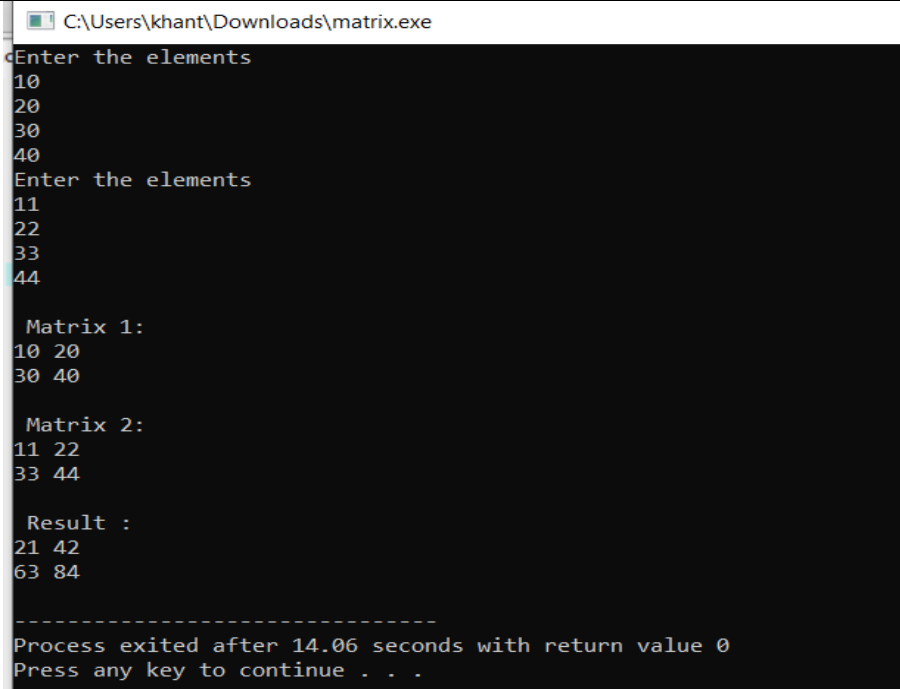
matrices operator +(matrices m2)    //To overload '*'
{
    matrices m3;
    for(int i=0;i<2;i++)
    {
        for(int j=0;j<2;j++)
            m3.a[i][j]=a[i][j]+m2.a[i][j];
    }
    return(m3);
}

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
```

**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> ob1.get_elements();           //Calling method  ob2.get_elements();           //Calling method  cout&lt;&lt;"\n Matrix 1:\n";  ob1.display();  cout&lt;&lt;"\n Matrix 2:\n";  ob2.display();  ob1=ob1+ob2;  cout&lt;&lt;"\n Result : \n";  ob1.display();  } </pre>
<b>Input given:</b>	<p>ELEMENTS OF 1 MATRIX:</p> <p>10 20</p> <p>30 40</p> <p>ELEMENTS OF 2 MATRIX:</p> <p>11 22</p> <p>33 44</p>
<b>Output Screenshot :</b>	 <p>The screenshot shows a Windows command prompt window titled 'C:\Users\khant\Downloads\matrix.exe'. The program prompts the user to 'Enter the elements' for two matrices. For Matrix 1, the inputs are 10, 20, 30, and 40. For Matrix 2, the inputs are 11, 22, 33, and 44. The program then displays 'Matrix 1:' followed by '10 20' and '30 40'. It displays 'Matrix 2:' followed by '11 22' and '33 44'. The result is shown as 'Result :' followed by '21 42' and '63 84'. At the bottom, it says 'Process exited after 14.06 seconds with return value 0' and 'Press any key to continue . . .'.</p>