Aim:

Write a program to add, subtract, multiply and divide two complex number using **Operator Overloading** .

Theory:

Operator overloading is an important concept in C++. It is a type of polymorphism in which an operator is overloaded to give user defined meaning to it. Overloaded operator is used to perform operation on
user-defined data type. For example, '+' operator can be overloaded to perform addition on various data types, like for Integer, String(concatenation) etc.
Almost any operator can be overloaded in C++. However, there are few operator which can not be overloaded. Operator that are not overloaded are follows

- Scope Resolution Operator - `::`
- Member Access or Dot operator - `.`
- Pointer-to-member Operator - `*`
- Ternary or Conditional Operator - `?:`

Syntax :

```
data_type classname :: operator symbol (arguments){
    //function body
}
```

Code :

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

class Complex {
    float real ;
    float imag ;

    public :
    Complex () {
        real = 0 ;
        imag = 0 ;
    }
    Complex (float x , float y) {
        real = x ;
        imag = y ;
    }
    void display(){
        cout << real << " + " << imag << "i" << endl ;
    }
    Complex operator + ( Complex x) {
```

```
        return Complex( real + x.real , imag + x.imag );
    }
    Complex operator - ( Complex x) {
        return Complex( real - x.real , imag - x.imag );
    }
    Complex operator * ( Complex x) {
        return Complex( real * x.real - imag * x.imag  , real * x.imag + imag * x.real )
    }
    Complex operator / ( Complex x) {
        return Complex( (real * x.real + imag * x.imag) / ( x.real * x.real + x.imag * x
                        (imag * x.real - real * x.imag) / ( x.real * x.real + x.imag * x
    }
} ;

int main(){

    Complex c1(2.0 , 3.0 ) , c2(3.0 , 4.0) , c3 ;

    cout << "c1 = " ;
    c1.display();
    cout << "c2 = " ;
    c2.display();

    cout << "c1 + c2 = " ;
    c3 = c1 + c2 ;
    c3.display();

    cout << "c1 - c2 = " ;
    c3 = c1 - c2 ;
    c3.display();

    cout << "c1 * c2 = " ;
    c3 = c1 * c2 ;
    c3.display();

    cout << "c1 / c2 = " ;
    c3 = c1 / c2 ;
    c3.display();

    return 0 ;
}
```

Output :

```
PS D:\College\OOPS> .\operator-overloading
c1 = 2 + 3i
c2 = 3 + 4i
c1 + c2 = 5 + 7i
c1 - c2 = -1 + -1i
c1 * c2 = -6 + 17i
c1 / c2 = 0.72 + 0.04i
```

Discussion :

In the above program we created a class complex which is used for carrying operation on complex numbers like addition, subtraction, multiplication and division with the help of operator overloading.
Operator overloading refers to operator polymorphism. For example, + operator can be used to add two number but also can be used to add two strings together. So similar above we overload +, -, *, / with each operator represents its respective operation on complex numbers operators for two complex numbers.

Learning Outcomes :

- We have learned that by using overloading operator our program will be more understandable. However, there are three methods to implement operator overloading that are: -
- Member Function
- Non-Member Function
- Friend Function
- Operator overloading function can be a member function if the Left operand is an Object of that class, but if the Left operand is different, then Operator overloading function must be a non-member function. Operator overloading function can be made friend function if it needs access to the private and protected members of class.
- However, we cannot overload some of the operators that are given below: -
- Scope Resolution Operator - `::`
- Member Access or Dot operator - `.`
- Pointer-to-member Operator - `*`
- Ternary or Conditional Operator - `?:`