|  |  |  |  |
| --- | --- | --- | --- |
| **Branch: Instrumentation & Control Engineering** | | **Year:** Second Year | |
| **Division: C** | **Roll No: 04** | **GR Number: 11911180** | **Subject:** OOPS |
| **Assignment No: 3** | **Date of Submission: 29-04-2021** | **Student Full Name: Shaunak Sudhir Deshpande** | |

Aim: Write C++ program for addition,substraction,multiplication and division of two complex numbers.

1. use all types of constructors. create muliplication function as friend function.

2. overload operators +, -,\*, /,<<,>>

Software Used: MinGW, VSCode

Code:

#include<iostream>

using namespace std;

class Complex {

private:

    double real, imag;

public:

    Complex(double r = 0, double i =0)

    {

        real = r;

        imag = i;

    }

    Complex operator + (Complex const &obj) {

         Complex res;

         res.real = real + obj.real;

         res.imag = imag + obj.imag;

         return res;

    }

    Complex operator - (Complex const &obj) {

         Complex res;

         res.real = real - obj.real;

         res.imag = imag - obj.imag;

         return res;

    }

    Complex operator \* (Complex const &obj) {

         Complex res;

         res.real = (real \* obj.real) - (imag \* obj.imag);

         res.imag = (imag \*  obj.real) + (real \* obj.imag);

         return res;

    }

    Complex operator / (Complex const &obj){

        Complex res;

        res.real= ((real \* obj.real) + (imag \* obj.imag))/ ((obj.imag\*obj.imag) + (obj.real \* obj.real));

        res.imag= ((imag \* obj.real) - (real \* obj.imag)) / ((obj.imag\*obj.imag) + (obj.real \* obj.real));

        return res;

    }

    void print()

    {

        if( imag>=0)

        cout << real << " + " << imag <<"i"<< endl;

        else

        cout << real << " - " << abs(imag) <<"i"<< endl;

    }

};

int main()

{

    Complex c1(10, 5), c2(2, 4);

    Complex c3 = c1 + c2;

    Complex c4 = c1 - c2;

    Complex c5 = c1 \* c2;

    Complex c6 = c1 / c2;

    cout<<endl<<endl;

    c3.print();

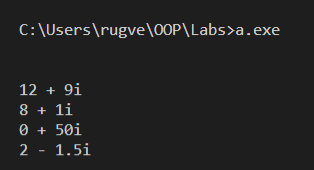
    c4.print();

    c5.print();

    c6.print();

}

Output:



Analysis of Program:

This code is an example of using operator overloading in C++. Since we know that +, - , / & \* operator don’t work the same way in complex numbers how they do for regular numbers, we use operator overloading to make sure that we can use these symbols for the addition, subtraction, division and multiplication of complex numbers just like real numbers.

Conclusion:

Operator overloading is a compile-time polymorphism in which the operator is overloaded to provide the special meaning to the user-defined data type, and can be used to redefine most of the operators in C++. It makes code easier to read and understand for humans, and has a wide range of applications.