**Modern Education Society’s**

**College of Engineering, Pune**

|  |  |
| --- | --- |
| **NAME OF STUDENT:** Prathamesh Kalyan Sable | **CLASS:** SE Comp 1 |
| **SEMESTER/YEAR:** Sem-3 / 2022-23 | **ROLL NO:** 015 |
| **DATE OF PERFORMANCE:**  / /2022 | **DATE OF SUBMISSION:**  / /2022 |
| **EXAMINED BY:** Prof. R. H. Shende | **EXPERIMENT NO: A-1** |

###### TITLE: Perform arithmetic operations on COMPLEX NUMBERS

**PROBLEM STATEMENT:**  Implement a class Complex which represents the Complex Number data type. Implement the following operations:

1. Constructor (including a default constructor which creates the complex number 0+0i).

2. Overloaded operator + to add two complex numbers.

3. Overloaded operator \* to multiply two complex numbers.

4. Overloaded << and >> to print and read Complex Numbers.

### **OBJECTIVES:**

1. To understand Polymorphism.
2. To understand Compile time polymorphism.

### **OUTCOMES:**

1. Develop programming application using object oriented programming language C++.
2. Percept the utility and applicability of OOP.

**PRE-REQUISITES:**

* 1. Knowledge of class, object .
  2. Knowledge of operator overloading, friend function.

**APPARATUS:**

Working Computer system with g++ installed

**QUESTIONS:**

1. What is Polymorphism?
2. What are the rules for overloading operators ?
3. How many arguments are required in the definition of an overloaded unary operator?

**Source Code:**

#include <iostream>

using namespace std;

class Complex

{

public:

    float real, imag;

    // default constructor

    Complex()

    {

        real = 0;

        imag = 0;

    }

    Complex(float real, float imag)

    {

        this->real = real;

        this->imag = imag;

    }

    // to display complex number

    void disp()

    {

        cout << \*this;

    }

    // addition operator overloading

    Complex operator+(Complex num2)

    {

        Complex res;

        res.real = (real + num2.real);

        res.imag = (imag + num2.imag);

        return res;

    }

    // multipication operator overloading

    Complex operator\*(Complex num2)

    {

        Complex res;

        res.real = ((real \* num2.real) - (imag \* num2.imag));

        res.imag = ((real \* num2.imag) + (imag \* num2.real));

        return res;

    }

    // insertion and extraction are not of complex class so overloaded using friend function

    friend ostream &operator<<(ostream &object, Complex &num);

    friend istream &operator>>(istream &object, Complex &num);

};

ostream &operator<<(ostream &object, Complex &num)

{

    // displaying complex number

    if (num.imag < 0)

    {

        object << num.real << " " << num.imag << "i";

    }

    else

    {

        object << num.real << " +" << num.imag << "i";

    }

    return object;

}

istream &operator>>(istream &object, Complex &num)

{

    // real and imag part input

    object >> num.real >> num.imag;

    return object;

}

int main()

{

    // Declearing Variables

    bool loop\_control = true;

    int choice;

    Complex num1, num2, res;

    while (loop\_control)

    {

        cout << "\n---- MENU ----" << endl;

        cout << "1. Add two Complex numbers\n2. Multiply two Complex Numbers\n3. Exit the Program" << endl;

        cout << "Enter your choice :";

        cin >> choice;

        switch (choice)

        {

        case 1:

            cout << "Enter First Complex Number :";

            cin >> num1;

            cout << "Enter Second Complex Number :";

            cin >> num2;

            res = num1 + num2;

            cout << "The Addition of " << num1 << " and " << num2 << " is " << res << endl;

            break;

        case 2:

            cout << "Enter First Complex Number :";

            cin >> num1;

            cout << "Enter Second Complex Number :";

            cin >> num2;

            res = num1 \* num2;

            cout << "The Multiplication of " << num1 << " and " << num2 << " is " << res << endl;

            break;

        case 3:

            loop\_control = false;

            cout << "Exiting, Thank You :)" << endl;

            break;

        default:

            cout << "Enter a valid choice!" << endl;

            break;

        }

    }

    return 0;

}

**OUTPUT:**

