UNIVERSITY OF BUEA

FACULTY OF ENGINEERING AND TECHNOLOGY

DEPARTMENT OF COMPUTER ENGINEERING

CEF 691 - ADVANCED OOP

COMPLEX NUMBER ADT

LENYA HOPE NEMBI
FE19P030

Introduction

Complex numbers are numbers that consist of two parts — a real number and an imaginary number. Complex numbers are the building blocks of more intricate math, such as algebra. They can be applied to many aspects of real life, especially in electronics and electromagnetism.

In this report, we are going to define complex numbers and the operations carried out on them to create a complex number abstract data type. The abstract data type will be implemented in C++ using templates as well as in C using structures.

Complex Number Abstract Data Type Specification

A complex number is made up of two parts; The Read(a) and the Imaginary part (b) and it is represent in the form below:

a + bi where i = √-1

The operations that will be carried out include;

- Creating a complex number.
- Adding, Dividing, Multiplying and Subtracting two complex numbers.
- Print out complex number.

IMPLEMENTATION

a) In C programming language

A main.c file was created which reads the real and imaginary parts of two complex numbers and then prints out the solution of the sum, difference, division and multiplication of both numbers. The code is shown below:

```
#include <stdio.h>
#include <string.h>
#include <cmath>

struct complex
```

```
{
   int real, img;
};

void printComplex(complex c);
void sum(complex a, complex b);
void subtract(complex a, complex b);
void divide(complex a, complex b);
void multiply(complex a, complex b);

int main()
{
   struct complex a, b, c;

   printf("Enter a and b where a + ib is the first complex number: C1.\n");
   scanf("%d%d", &a.real, &a.img);
   printf("Enter c and d where c + id is the second complex number: C2.\n");
   scanf("%d%d", &b.real, &b.img);
    printf("\n");
    printf("C1: "); printComplex(a);
     printf("C2: "); printComplex(b);
     printf("\n");
   sum(a, b);
   subtract(a, b);
   divide(a, b);
   multiply(a, b);
   return 0;
}

void multiply(complex a, complex b) {
          struct complex c;
```

```
  c.real = a.real * b.real - a.img * b.img;
  c.img = a.real * b.img + a.img * b.real;

  printf("C1 * C2: ");
  printComplex(c);
}

void divide(complex a, complex b) {
        struct complex c;
  c.real = (a.real * b.real + a.img * b.img) / (pow(b.real, 2) + pow(b.img, 2));
  c.img = (a.img * b.real - a.real * b.img) / (pow(b.real, 2) + pow(b.img, 2));
  printf("C1 / C2: ");
  printComplex(c);
}

void subtract(complex a, complex b) {
        struct complex c;
        c.real = a.real - b.real;
  c.img = a.img - b.img;
  printf("C1 - C2: ");
  printComplex(c);
}

void sum(complex a, complex b) {
        struct complex c;
        c.real = a.real + b.real;
  c.img = a.img + b.img;
  printf("C1 + C2: ");
  printComplex(c);
}
```

```
void printComplex(complex c) {
                if ((c.img > 0 && c.real > 0) || (c.img > 0 && c.real < 0)) {
    printf("%d + %di\n", c.real, c.img);
  } else if (c.img == 0) {
    printf("%d\n", c.real);
  } else if (c.real == 0) {
    printf("%di\n", c.img);
  } else {
    printf("%d %di\n", c.real, c.img);
  }
}
```

Testing

The following result was obtained when running the program from the terminal.

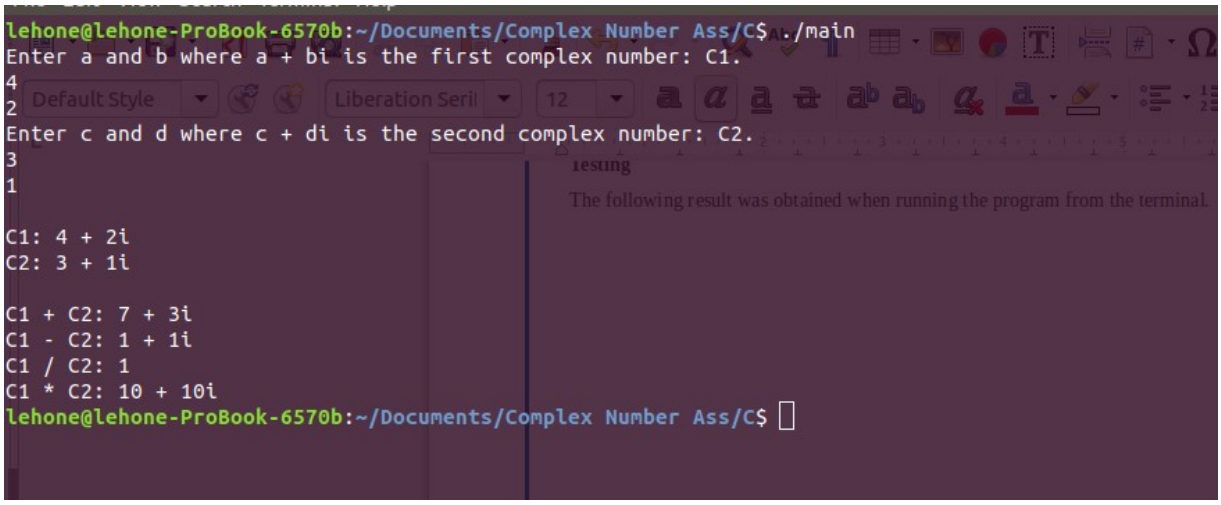

Figure 1: Output sample of C implementation

b) In C++ programming language

The C++ program is made up of three files; the interface (Complex.h), implementation of the interface operations (Complex.cpp) and a test program file (test.cpp). The content of the various files are shown below;

Complex.h

```
#ifndef COMPLEX__H__
#define COMPLEX__H__

#include <string>
#include <cmath>
#include <iostream>

class Complex {
        public:
                Complex();
                Complex(double re, double im);
                Complex add(Complex a);
                Complex subtract(Complex a);
                Complex multiply(Complex a);
                Complex divide(Complex a);
                friend Complex operator+(Complex a, const Complex &b);
                friend Complex operator-(Complex a, const Complex &b);
                friend Complex operator*(Complex a, const Complex &b);
                friend Complex operator/(Complex a, const Complex &b);
                friend std::ostream &operator<<(std::ostream &out, const Complex &c);
        private:
                double real;
                double imag;
};
```

Complex.cpp

```
#include "Complex.h"
using namespace std;

Complex::Complex(double re, double im)
{
   real = re;
   imag = im;
}

Complex::Complex()
{
   real = 0;
   imag = 0;
}

// Print complex number
ostream& operator<<(ostream& out, const Complex& c)
{
   if ((c.imag > 0 && c.real > 0) || (c.imag > 0 && c.real < 0)) {
      out << "" << c.real << "+" << c.imag << "i";
   } else if (c.imag == 0) {
      out << c.real;
   } else if (c.real == 0) {
      out << c.imag << "i";
   } else {
      out << c.real << "" << c.imag << "i";
   }
   return out;
}
```

```
// Add complex numbers
Complex Complex::add(Complex a) {
                return *this + a;
}

Complex operator+(Complex a, const Complex& b) 77{
   a.real += b.real;
   a.imag += b.imag;
   return a;
}

// Subtract complex numbers
Complex Complex::subtract(Complex a) {
                return *this - a;
}

Complex operator-(Complex a, const Complex& b) {
   a.real -= b.real;
   a.imag -= b.imag;
   return a;
}

// multiply complex numbers
Complex Complex::multiply(Complex a) {
                return *this * a;
}
Complex operator*(Complex a, const Complex& b) {
   double x, y;
   x = a.real * b.real - a.imag * b.imag;
   y = a.real * b.imag + a.imag * b.real;
```

```
    a.real = x;
    a.imag = y;
    return a;
}

// divide complex numbers
Complex Complex::divide(Complex a)  {
                    return *this / a;
}
Complex operator/(Complex a, const Complex& b) {
    double x, y;
    x = (a.real * b.real + a.imag * b.imag) / (pow(b.real, 2) + pow(b.imag, 2));
    y = (a.imag * b.real - a.real * b.imag) / (pow(b.real, 2) + pow(b.imag, 2));
    a.real = x;
    a.imag = y;
    return a;
}
```

test.cpp

```
#include "Complex.h"
#include <array>
using namespace std;

int main()
{
    std::array<Complex, 2> cx;
    double x, y;

    for (int i = 0; i < 2; i++) {
        cout << "Enter complex number " << i + 1 << endl;
        cout << "Real part" << endl;
```

```
        cin >> x;
        cout << "Imaginary part" << endl;
        cin >> y;
        cx[i] = Complex(x, y);
        cout << endl;
    }
    cout << "c1: " << cx[0] << endl;
    cout << "c2: " << cx[1] << endl << endl;
    cout << "c1 + c2: " << cx[0].add(cx[1]) << endl;
    cout << "c1 - c2: " << cx[0].subtract(cx[1]) << endl << endl;
    cout << "c1 / c2 : " << cx[0].divide(cx[1]) << endl;
    cout << "c1 * c2 : " << cx[0].multiply(cx[1]) << endl << endl;
}
```

Testing

Upon running the program, the following output was obtained.