МИНИСТЕРСТВО ОБРАЗОВАНИЯ И НАУКИ РОССИЙСКОЙ ФЕДЕРАЦИИ МОСКОВСКИЙ АВИАЦИОННЫЙ ИНСТИТУТ

(НАЦИОНАЛЬНЫЙ ИССЛЕДОВАТЕЛЬСКИЙ УНИВЕРСТИТЕТ)

ЛАБОРАТОРНАЯ РАБОТА №2

по курсу объектно-ориентированное программирование I семестр, 2021/22 уч. год

Студент *Абаровский Олег Александрович, группа М8О-207Б-20*

Преподаватель *Дорохов Евгений Павлович*

Условие

Разработать программу на языке C++ согласно варианту задания. Программа на C++ должна собираться с помощью системы сборки CMake. Программа должна получать данные из стандартного ввода и выводить данные в стандартный вывод.
Необходимо настроить сборку лабораторной работы с помощью CMake. Собранная программа должна называться oop_exercise_01
Задание:
Вариант 1: Комплексное число в алгебраической форме представляются парой действительных чисел (a, b), где a – действительная часть, b – мнимая часть. Реализовать класс Complex для работы с комплексными числами.
Обязательно должны быть присутствовать операции
- сложения add, (a, b) + (c, d) = (a + c, b + d);
- вычитания sub, (a, b) – (c, d) = (a – c, b – d);
- умножения mul, (a, b) ´ (c, d) = (ac – bd, ad + bc);
- деления div, (a, b) / (c, d) = (ac + bd, bc – ad) / (c 2 + d 2);
- сравнение equ, (a, b) = (c, d), если (a = c) и (b = d);
- сопряженное число conj, conj(a, b) = (a, –b).

Операции над объектами реализовать в виде перегрузки операторов.
Реализовать пользовательский литерал для работы с константами объектов созданного класса.

Описание программы

Исходный код лежит в файле main.cpp, в котором реализован класс комплексного числа и описаны методы этого класса, среди которых сложение, вычитание, умножение, деление, сравнение, нахождение сопряженного числа, реализованные в виде перегрузки операторов.

Дневник отладки

Исправлений не потребовалось.

Недочёты

Выводы

В ходе выполнения лабораторной работы я изучил такие понятия как перегрузка операторов и пользовательский литерал в c++ и создал класс с набором методов, реализованных в виде перегрузки операторов. Возможность применения стандартных понятных операторов к собственным классам показалась мне интересной и полезной для облегчения работы с программой.

Исходный код

main.cpp

```
#include <iostream>

class Complex
{
public:
  double re, im;
    Complex(){}
    Complex(double r, double i)
    {
      re = r; im = i;
    }
    Complex& operator = (Complex);
    Complex operator + (Complex);
    Complex operator - (Complex);
    Complex operator * (Complex&);
    Complex operator / (Complex&);
    bool operator == (Complex& com);
    Complex conj()
  {
      return Complex(re, -im);
  }
  ~Complex(){}
};

Complex& Complex::operator = (Complex com)
{
    this->re = com.re;
    this->im = com.im;
    return *this;
}

bool Complex::operator==(Complex& com)
{
    if(this->re == com.re && this->im == com.im)
        return 1;
    else
        return 0;
}
```

```
Complex Complex::operator*(Complex &com)
{
    double i, j;
    i = re * com.re - im * com.im;
    j = re * com.im + com.re * im;
    re = i;
    im = j;
    return *this;
}

Complex Complex::operator/(Complex &com)
{
    double i, j, k;
    k = re * re +com.im * com.im;
    i = (re * com.re + im * com.im) / k;
    j = (com.re * im - re * com.im) / k;
    re = i;
    im = j;
    return *this;
}

Complex Complex::operator+(Complex com)
{
    this->re = this->re + com.re;
    this->im = this->im + com.im;
    return *this;
}

Complex Complex::operator-(Complex com)
{
    this->re = this->re - com.re;
    this->im = this->im - com.im;
    return *this;
}

double operator "" _R(long double re)
{
    return re;
}

double operator "" _I(long double im)
{
```

```
    return im;
}

int main()
{
    double re, im;
  bool f;
  Complex com1, com2, com, comsub, Ccom1, Ccom2;
  printf("Enter complex number one\n");
  scanf("%lf %lf", &re, &im);
  com1 = Complex(re, im);
  printf("Enter complex number two\n");
  scanf("%lf %lf", &re, &im);
  com2 = Complex(re, im);
  Ccom1 = com1;
  Ccom2 = com2;
  printf("com1 = (%lf, %lf)\n", com1.re, com1.im);
  printf("com2 = (%lf, %lf)\n", com2.re, com2.im);
  com = com1 + com2;
  printf("Sum = (%lf, %lf)\n", com.re, com.im);
  com1 = Ccom1;
  com2 = Ccom2;
  comsub = com1 - com2;
  printf("Sub = (%lf, %lf)\n", comsub.re, comsub.im);
  com1 = Ccom1;
  com2 = Ccom2;
  com = com1 * com2;
  printf("Mul = (%lf, %lf)\n", com.re, com.im);
  com1 = Ccom1;
  com2 = Ccom2;
  com = com1 / com2;
  printf("Div = (%lf, %lf)\n", com.re, com.im);
  com1 = Ccom1;
  com2 = Ccom2;
  f = (com1 == com2);
  printf("Equ = %d\n", f);
  com1 = Ccom1;
  com2 = Ccom2;
  com1 = com1.conj();
  printf("Conj1 = (%lf, %lf)\n", com1.re, com1.im);
  com1 = Ccom1;
  com2 = Ccom2;
```

```
  com2 = com2.conj();
  printf("Conj2 = (%lf, %lf)\n", com2.re, com2.im);
}
```