МИНИСТЕРСТВО ОБРАЗОВАНИЯ И НАУКИ РОССИЙСОЙ ФЕДЕРАЦИИ МОСКОВСКИЙ АВЦИАЦИОННЫЙ ИНСТИТУТ

(НАЦИОНАЛЬНЫЙ ИССЛЕДОВАТЬЕЛЬСКИЙ УНИВЕРСТИТЕТ)

ЛАБОРАТОРНАЯ РАБОТА №2 **по курсу**

объектно-ориентированное программирование I семестр, 2019/20 уч. год

Студент *Артемьев Дмитрий Иванович, группа М8О-206Б-18*

Преподаватель *Журавлёв Андрей Андреевич*

Условие

Задание №1: написать класс, который реализует комплексные числа в алгебраической форме с операциями:

1. сложения add, (a, b) + (c, d) = (a + c, b + d);
2. вычитания sub, (a, b) – (c, d) = (a – c, b – d);
3. умножения mul, (a, b) * (c, d) = (ac – bd, ad + bc);
4. деления div, (a, b) / (c, d) = (ac + bd, bc – ad) / (c2 + d2);
5. сравнение equ, (a, b) = (c, d), если (a = c) и (b = d);
6. сопряженное число conj, conj(a, b) = (a, –b);
7. сравнения модулей.

Описание программы

Исходный код лежит в 3 файлах:

1. src/main.cpp: основная программа, которая считывает 2 комплексных числа и обрабатывает их
2. include/Complex.hpp: описание класса, объявление и реализация методов операций
3. src/Complex.cpp: реализация нереализованных методов класса комплексных чисел, чтобы было

Дневник отладки

Сравнении типа double нужно проводить с определённой точностью

Недочёты

Не идеально прописаны файлы meson.build.

Выводы

Я изучил перегрузку операторов в языке C++, систему сборки проектов meson, библиотеку для модульного тестирования Google C++ Testing Framework.

Исходный код

Complex.hpp

```
#pragma once

#include <iostream>
#include <cmath>

class Complex{
public:
    Complex() = default;

    Complex(double re, double im) noexcept
        : re(re)
        , im(im)
    {}

    Complex(Complex const& other) noexcept
        : re(other.re)
        , im(other.im)
    {}

    ~Complex() {};

    friend Complex operator+ (Complex const& left, Complex const& right);
    friend Complex operator- (Complex const& left, Complex const& right);
    friend Complex operator* (Complex const& left, Complex const& right);
    friend Complex operator/ (Complex const& left, Complex const& right);
    friend bool operator== (Complex const& left, Complex const& right);
    friend bool operator!= (Complex const& left, Complex const& right);

    friend Complex conj(Complex const& other);
    friend double abs(Complex const& other);
    friend int cmp(Complex const& left, Complex const& right);

    friend std::ostream& operator<< (std::ostream& out, Complex const& other);
    friend std::istream& operator>> (std::istream& in, Complex& other);

    Complex& operator- () {
        (*this) = Complex(-re, -im);
        return *this;
    }
```

```
    Complex& operator+ () {
        (*this) = Complex(re, im);
        return *this;
    }
private:
    double re;
    double im;
};

Complex operator"" _im(long double right);
Complex operator"" _re(long double left);
Complex operator"" _im(unsigned long long right);
Complex operator"" _re(unsigned long long left);
```

Complex.cpp

```cpp
#include "Complex.hpp"
#include <iostream>
#include <iomanip>

Complex operator+ (Complex const& left, Complex const& right) {
    return Complex(left.re + right.re, left.im + right.im);
}

Complex operator- (Complex const& left, Complex const& right) {
    return Complex(left.re - right.re, left.im - right.im);
}

Complex operator* (Complex const& left, Complex const& right) {
    return Complex(left.re * right.re - left.im * right.im,
                   left.re * right.im + left.im * right.re);
}

Complex operator/ (Complex const& left, Complex const& right) {
    double den = (right.re * right.re + right.im * right.im);
    return Complex(
                   (left.re * right.re + left.im * right.im) / den,
                   (left.im * right.re - left.re * right.im) / den
                   );
}

bool operator== (Complex const& left, Complex const& right) {
    return (left.re - right.re) < 0.001 && (left.im - right.im) < 0.001;
}

bool operator!= (Complex const& left, Complex const& right) {
    return !(left == right);
}

double abs(Complex const& other) {
    return sqrt(other.re * other.re + other.im * other.im);
}

Complex conj(Complex const& other) {
    return Complex(other.re, -other.im);
}
```

```
int cmp(Complex const& left, Complex const& right) {
    return (abs(left) > abs(right)) ? 1
        : (abs(left) == abs(right)) ? 0
        : -1;
}

std::ostream& operator<< (std::ostream& out, Complex const& other) {
    out << std::fixed << std::setprecision(3) << other.re << ' ' << other.im;
    return out;
}

std::istream& operator>> (std::istream& in, Complex& other) {
    in >> other.re >> other.im;
    return in;
}

Complex operator"" _im(long double right) {
    return Complex(0, double(right));
}

Complex operator"" _re(long double left) {
    return Complex(double(left), 0);
}

Complex operator"" _im(unsigned long long right) {
    return Complex(0, double(right));
}

Complex operator"" _re(unsigned long long left) {
    return Complex(double(left), 0);
}
```

main.cpp

```cpp
#include <iostream>
#include "Complex.hpp"

int main(){
    Complex a, b;
    std::cin >> a >> b;

    std::cout << "+ " << a + b << std::endl;
    std::cout << "- " << a - b << std::endl;
    std::cout << "* " << a * b << std::endl;
    std::cout << "/ " << a / b << std::endl;
    std::cout << "== " << (a == b) << std::endl;
    std::cout << "conj a = " << conj(a) << std::endl;
    std::cout << "conj b = " << conj(b) << std::endl;
    std::cout << "|a| > |b| is " << cmp(a, b) << std::endl;
    std::cout << "literal (3.14_re + 1.59_im) + (-2.65_re + 3.58_im) = " << (3.14_re + 1
    return 0;
}
```