МИНИСТЕРСТВО ОБРАЗОВАНИЯ И НАУКИ РОССИЙСКОЙ ФЕДЕРАЦИИ
МОСКОВСКИЙ АВИАЦИОННЫЙ ИНСТИТУТ
(НАЦИОНАЛЬНЫЙ ИССЛЕДОВАТЕЛЬСКИЙ УНИВЕРСИТЕТ)

ЛАБОРАТОРНАЯ РАБОТА №1

по курсу “Объектно-ориентированное программирование”

I семестр, 2021/22 учебный год

Студент: *Рожков Илья Алексеевич, группа М8О-207Б-20*

Преподаватель: *Дорохов Евгений Павлович, каф. 806*

Задание:

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.

Вариант №21:

Комплексные Числа

Дневник отладки:

Проблем не было.

Вывод:

При выполнении работы я на практике познакомился с CMake и классами, перегрузил операторы и выложил всё на гитхаб. Очень удобные технологии.

Исходный код:

CMakeLists.txt

```
cmake_minimum_required(VERSION 3.20)
project(oop_exercise_01)

set(CMAKE_CXX_STANDARD 23)

add_executable(oop_exercise_01 main.cpp Complex.cpp Complex.h)
```

Complex.cpp

```
//
// Created by Илья Рожков on 28.11.2021.
//

#include "Complex.h"
#include <cmath>

Complex::Complex() : _real(0), _imag(0){

}

Complex::Complex(float a) : _real(a), _imag(0) {

}

Complex::Complex(float a, float b) : _real(a), _imag(b){

}

std::istream &operator>>(std::istream &is, Complex &a) {
    is >> a._real >> a._imag;
    return is;
}
```

```cpp
std::ostream &operator<<(std::ostream &out, Complex &a) {
    out << a._real << ' ' << a._imag;
    return out;
}

bool Complex::operator==(Complex& a) const {
    return (_real == a._real) && (_imag == a._imag);
}

Complex &Complex::operator=(const Complex& a) {
    if (this == &a)
        return *this;
    _real = a._real;
    _imag = a._imag;
    return *this;
}

bool Complex::operator==(const Complex &a) const {
    return (_real == a._real) && (_imag == a._imag);
}

Complex Complex::operator+(Complex &a) const {
    Complex b;
    b._real = _real + a._real;
    b._imag = _imag + a._imag;
    return b;
}

Complex Complex::operator+(const Complex &a) const {
    Complex b;
    b._real = _real + a._real;
    b._imag = _imag + a._imag;
    return b;
}

Complex Complex::operator-(Complex &a) const {
    Complex b;
    b._real = _real - a._real;
    b._imag = _imag - a._imag;
    return b;
}

Complex Complex::operator-(const Complex &a) const {
    Complex b;
    b._real = _real - a._real;
    b._imag = _imag - a._imag;
    return b;
}

bool Complex::operator!=(const Complex &a) const {
    return !(*this == a);
}
```

```
bool Complex::operator!=(Complex &a) const {
    return !(*this == a);
}

Complex &Complex::operator+=(const Complex &a) {
    *this = *this + a;
    return *this;
}

Complex &Complex::operator+=(Complex &a) {
    *this = *this + a;
    return *this;
}

Complex &Complex::operator-=(const Complex &a) {
    *this = *this - a;
    return *this;
}

Complex &Complex::operator-=(Complex &a) {
    *this = *this - a;
    return *this;
}

Complex Complex::operator*(Complex &a) const {
    Complex b;
    b._real = _real * a._real - _imag * a._imag;
    b._imag = _real * a._imag + _imag * a._real;
    return b;
}

Complex Complex::operator*(const Complex &a) const {
    Complex b;
    b._real = _real * a._real - _imag * a._imag;
    b._imag = _real * a._imag + _imag * a._real;
    return b;
}

Complex Complex::operator/(const Complex &a) const {
    Complex b;
    b._real = (_real * a._real + _imag * a._imag) / (a._real *
a._real + a._imag * a._imag);
    b._imag = (_imag * a._real - _real * a._imag) / (a._real *
a._real + a._imag * a._imag);
    return b;
}

Complex Complex::operator/(Complex &a) const {
    Complex b;
    b._real = (_real * a._real + _imag * a._imag) / (a._real *
a._real + a._imag * a._imag);
    b._imag = (_imag * a._real - _real * a._imag) / (a._real *
a._real + a._imag * a._imag);
    return b;
```

```
}

Complex& Complex::operator/=(const Complex &a) {
    *this = *this / a;
    return *this;
}

Complex& Complex::operator/=(Complex &a) {
    *this = *this / a;
    return *this;
}

Complex &Complex::operator*=(const Complex &a) {
    *this = *this * a;
    return *this;
}

Complex& Complex::operator*=(Complex &a) {
    *this = *this * a;
    return *this;
}

Complex Complex::conj() const {
    Complex a;
    a._real = _real;
    a._imag = (-1) * _imag;
    return a;
}

float Complex::abs() const {
    return sqrt(_real * _real + _imag * _imag);
}

bool Complex::operator<(const Complex &a) const {
    return this->abs() < a.abs();
}

bool Complex::operator<(Complex &a) const {
    return this->abs() < a.abs();
}

bool Complex::operator>(const Complex &a) const {
    return this->abs() > a.abs();
}

bool Complex::operator>(Complex &a) const {
    return this->abs() > a.abs();
}

bool Complex::operator>=(const Complex &a) const {
    return this->abs() >= a.abs();
}

bool Complex::operator>=(Complex &a) const {
```

```
    return this->abs() >= a.abs();
}

bool Complex::operator<=(const Complex &a) const {
    return this->abs() <= a.abs();
}

bool Complex::operator<=(Complex &a) const {
    return this->abs() <= a.abs();
}

Complex Complex::add(Complex &a) const {
    return *this + a;
}

Complex Complex::add(const Complex &a) const {
    return *this + a;
}

Complex Complex::sub(Complex &a) const {
    return *this - a;
}

Complex Complex::sub(const Complex &a) const {
    return *this - a;
}

Complex Complex::mul(Complex &a) const {
    return *this * a;
}

Complex Complex::mul(const Complex &a) const {
    return *this * a;
}

Complex Complex::div(Complex &a) const {
    return *this / a;
}

Complex Complex::div(const Complex &a) const {
    return *this / a;
}
```

Complex.h

```
//
// Created by Илья Рожков on 28.11.2021.
//

#ifndef OOP_EXERCISE_01_COMPLEX_H
#define OOP_EXERCISE_01_COMPLEX_H
#include "iostream"
```

```
class Complex {
    public:
    Complex();
    explicit Complex(float a);
    Complex(float a, float b);
    friend std::istream& operator>> (std::istream& is, Complex&
a);
    friend std::ostream& operator<< (std::ostream& out,
Complex& a);
    bool operator==(Complex& a) const;
    bool operator==(const Complex& a) const;
    Complex& operator=(const Complex& a);
    Complex operator+(Complex& a) const;
    Complex operator+(const Complex& a) const;
    Complex operator-(Complex& a) const;
    Complex operator-(const Complex& a) const;
    bool operator!=(const Complex& a) const;
    bool operator!=(Complex& a) const;
    Complex& operator+=(const Complex& a);
    Complex& operator+=(Complex& a);
    Complex& operator-=(const Complex& a);
    Complex& operator-=(Complex& a);
    Complex operator*(const Complex& a) const;
    Complex operator/(const Complex& a) const;
    Complex operator*(Complex& a) const;
    Complex operator/(Complex& a) const;
    Complex& operator/=(const Complex& a);
    Complex& operator/=(Complex& a);
    Complex& operator*=(const Complex& a);
    Complex& operator*=(Complex& a);
    Complex conj() const;
    float abs() const;
    bool operator<(const Complex& a) const;
    bool operator<(Complex& a) const;
    bool operator>(const Complex& a) const;
    bool operator>(Complex& a) const;
    bool operator>=(const Complex& a) const;
    bool operator>=(Complex& a) const;
    bool operator<=(const Complex& a) const;
    bool operator<=(Complex& a) const;
    Complex add(Complex& a) const;
    Complex add(const Complex& a) const;
    Complex sub(Complex& a) const;
    Complex sub(const Complex& a) const;
    Complex mul(Complex& a) const;
    Complex mul(const Complex& a) const;
    Complex div(Complex& a) const;
    Complex div(const Complex& a) const;


protected:
    float _real, _imag;
};
```

```
#endif //OOP_EXERCISE_01_COMPLEX_H
```