МИНИСТЕРСТВО ОБРАЗОВАНИЯ И НАУКИ РОССИЙСКОЙ ФЕДЕРАЦИИ МОСКОВСКИЙ АВИАЦИОННЫЙ ИНСТИТУТ

(НАЦИОНАЛЬНЫЙ ИССЛЕДОВАТЕЛЬСКИЙ УНИВЕРСТИТЕТ)

ЛАБОРАТОРНАЯ РАБОТА №1 по курсу объектно-ориентированное программирование I семестр, 2021/22 уч. год

Студент *Борисов Ян Артурович, группа М80-208Б-20*

Преподаватель *Дорохов Евгений Павлович*

**Цель:**

● Изучение системы сборки на языке C++, изучение систем контроля версии.

● Изучение основ работы с классами в С++;

## Порядок выполнения работы

1. Ознакомиться с теоретическим материалом.

2. Получить у преподавателя вариант задания.

3. Реализовать задание своего варианта в соответствии с поставленными требованиями.

4. Подготовить тестовые наборы данных.

5. Создать репозиторий на GitHub.

6. Отправить файлы лабораторной работы в репозиторий.

7. Отчитаться по выполненной работе путём демонстрации работающей программы на тестовых наборах данных (как подготовленных самостоятельно, так и предложенных преподавателем) и ответов на вопросы преподавателя (как из числа контрольных, так и по реализации программы).

## Требования к программе

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** (в случае использования Windows **oop\_exercise\_01.exe)**

Необходимо зарегистрироваться на GitHub (если студент уже имеет регистрацию на GitHubто можно использовтаь ее) и создать репозитарий для задания лабораторной работы.

Преподавателю необходимо предъявить ссылку на публичный репозиторий на Github. Имя репозитория должно быть [https://github.com/*login*/oop\_exercise\_01](https://github.com/login/oop_exercise_01)

Где login – логин, выбранный студентом для своего репозитория на Github.

Репозиторий должен содержать файлы:

· main.cpp //файл с заданием работы

· CMakeLists.txt // файл с конфигураций CMake

· test\_xx.txt // файл с тестовыми данными. Где xx – номер тестового набора 01, 02 , … Тестовых наборов должно быть больше 1.

· report.doc // отчет о лабораторной работе

**Описание программы**

Исходный код лежит в файле: main.cpp  
  
  
  
**Дневник отладки**

Во время выполнения лабораторной работы программа не нуждалась в отладке, все ошибки компиляции были исправлены с первой попытки. После их исправления программа работала так, как было задумано изначально.

**Недочёты**  
Недочётов не было обнаружено.

**Выводы**

Данная лабораторная работа помогла мне использовать полученные на лекциях теоретические знания на практике, и я написал простенький полностью работающий класс.

**Исходный код**

#include <iostream>

#include <cmath>

class Complex{

public:

Complex();

Complex(double r, double j);

Complex add(Complex num2);

Complex sub(Complex num2);

Complex mul(Complex num2);

Complex div(Complex num2);

bool equ(Complex num2);

bool equ\_by\_real(Complex num2);

Complex conj();

Complex alg\_form(Complex num);

Complex alg\_form\_to\_geom(Complex num);

friend std::istream& operator>> (std::istream &in, Complex &num);

friend std::ostream& operator<< (std::ostream &out, const Complex &num);

private:

double r,j;

};

Complex::Complex(double r, double j) {

this->r = r;

this->j = j;

}

Complex Complex::add(Complex num2) {

Complex alg\_form1 = alg\_form(\*this);

Complex alg\_form2 = alg\_form(num2);

Complex result(alg\_form1.r + alg\_form2.r, alg\_form1.j + alg\_form2.j);

return alg\_form\_to\_geom(result);

}

Complex Complex::sub(Complex num2) {

if(this->r == num2.r && this->j == num2.j) return Complex(0, 0);

Complex alg\_form1 = alg\_form(\*this);

Complex alg\_form2 = alg\_form(num2);

Complex result(alg\_form1.r - alg\_form2.r, alg\_form1.j - alg\_form2.j);

return alg\_form\_to\_geom(result);

}

Complex Complex::mul(Complex num2) {

return Complex(this->r \* num2.r, this->j + num2.j);

}

Complex Complex::div(Complex num2) {

if(num2.r == 0 && num2.j == 0){

std::cout << "На 0 делить нельзя!" << std::endl;

return \*this;

}

return Complex(this->r / num2.r, this->j - num2.j);

}

bool Complex::equ(Complex num2) {

return this->r == num2.r && this->j == num2.j;

}

Complex Complex::conj() {

return Complex(this->r, -this->j);

}

bool Complex::equ\_by\_real(Complex num2) {

const double e = 1e-5;

Complex alg\_form1 = alg\_form(\*this);

Complex alg\_form2 = alg\_form(num2);

return (std::abs(alg\_form1.r - alg\_form2.r) < e);

}

Complex Complex::alg\_form(Complex num) {

return Complex(num.r \* cos(num.j), num.r \* sin(num.j));

}

Complex Complex::alg\_form\_to\_geom(Complex num) {

double a = num.r, b = num.j;

double z = sqrt(a \* a + b \* b);

double argZ = (-3.14 + atan(b / a)) \* (180 / 3.14);

return Complex(z, argZ);

}

Complex::Complex() {

this->r = 0;

this->j = 0;

}

std::istream &operator>>(std::istream &in, Complex &num) {

in >> num.r >> num.j;

return in;

}

std::ostream &operator<<(std::ostream &out, const Complex &num) {

out << "Complex num in trigonometric form: " << num.r << "\*(cos" << num.j << " + i \* sin" << num.j << ")" << std::endl;

return out;

}

int main() {

Complex test\_num1, test\_num2;

std::cin >> test\_num1;

std::cin >> test\_num2;

std::cout << test\_num1 << test\_num2 << std::endl;

std::cout << "Сложение**\n**" << test\_num1.add(test\_num2) << std::endl;

std::cout << "Вычитание**\n**" << test\_num1.sub(test\_num2) << std::endl;

std::cout << "Умножение**\n**" << test\_num1.mul(test\_num2) << std::endl;

std::cout << "Деление**\n**" << test\_num1.div(test\_num2) << std::endl;

std::cout << "Проверка на равенство**\n**" << test\_num1.equ(test\_num2) << std::endl;

std::cout << "Сопряженное число**\n**" << test\_num1.conj() << std::endl;

std::cout << "Проверка на равенство по действительной части**\n**" << test\_num1.equ\_by\_real(test\_num2) << std::endl;

return 0;

}