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 C++;

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

Вариант задания: 2, Комплексное число в тригонометрической форме

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

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 полностью работающий класс, определил его поля.

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;
}
```