

# ЛАБОРАТОРНАЯ РАБОТА №1

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

Студент Фаттахетдинов Сильвестр Динарович, 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: 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`,  $(a, b) \cdot (c, d) = (ac - bd, ad + bc)$ ;
- деления `div`,  $(a, b) / (c, d) = (ac + bd, bc - ad) / (c^2 + d^2)$ ;
- сравнение `eq`,  $(a, b) = (c, d)$ , если  $(a = c)$  и  $(b = d)$ ;
- сопряженное число `conj`,  $\text{conj}(a, b) = (a, -b)$ .

Реализовать операции сравнения модулей.

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 использовать ее) и создать репозиторий 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 3 файлах:

1. main.cpp - исполняемый код.
2. complex.h - специальный файл .h, содержащий прототипы используемых мною функций.
3. complex.cpp - реализация функций для моего задания.
4. CMakeLists.txt - специальный дополнительный файл типа CMakeLists.

# Дневник отладки

Программа нуждалась лишь в исправлении нескольких синтаксических ошибок, отладка не заняла много времени.

## Недочёты

Недочётов не было обнаружено.

## Выводы

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

## Исходный код

complex.h

```
#include <cmath>
#include <iostream>
class complex{
public:
    complex(double a1, double b1);
    complex(const complex &c);

    void add(complex c); //сложение
    void sub(complex c); //вычитание
    void mul(complex c); //умножение
    void div(complex c); //деление
    bool equ(complex c); //сравнение
    bool equm(complex c); //сравнение по модулю
    complex conj(); //сопряжённое

    complex& operator= (const complex &c);
    void print(){
        std::cout <<"(" << a <<", " << b << "i)";
    };
private:
    double a;
    double b;
```

```
};
```

## complex.cpp

```
#include "complex.h"

complex::complex(double a1, double b1){
    a = a1;
    b = b1;
}

complex::complex(const complex &c){
    a = c.a;
    b = c.b;
}

void complex::add(complex c){
    a = a + c.a;
    b = b + c.b;
}

void complex::sub(complex c){
    a = a - c.a;
    b = b - c.b;
}

void complex::mul(complex cc){
    double c = cc.a;
    double d = cc.b;
    double a1 = a*c - b*d;
    b = a*d + b*c;
    a = a1;
}

void complex::div(complex cc){
    double c = cc.a;
    double d = cc.b;
    double a1 = (a*c + b*d) / (c*c + d*d);
    b = (b*c - a*d) / (c*c + d*d);
    a = a1;
}

bool complex::equ(complex c){
    if (a == c.a && b == c.b) return true;
    return false;
}
```

```

bool complex::equm(complex c){
    if (abs(a) == abs(c.a) && abs(b) == abs(c.b)) return true;
    return false;
}

complex complex::conj(){
    complex ans(a, -b);
    return ans;
}

complex& complex::operator= (const complex &c){
    a = c.a;
    b = c.b;
    return *this;
}

```

## main.cpp

```

#include "complex.h"
void printop(complex a, complex b, complex res, char c){
    a.print();
    std::cout << " " << c << " ";
    b.print();
    std::cout << " = ";
    res.print();
    std::cout << "\n";
};

int main(){
    double real, img, real2, img2;
    char option = 'y';
    while(option != 'n'){
        std::cout << "Enter first number: ";
        std::cin >> real >> img;
        std::cout << "Enter second number: ";
        std::cin >> real2 >> img2;
        complex c(real, img);
        complex c2(real2, img2);
        complex c1(c);

        c1.add(c2);
        printop(c, c2, c1, '+');

        c1 = c;
        c1.sub(c2);
        printop(c, c2, c1, '-');

        c1 = c;
    }
}

```

```

    c1.mul(c2);
    printop(c, c2, c1, '*');

    c1 = c;
    c1.div(c2);
    printop(c, c2, c1, '/');

    c1 = c;
    if(c1.equ(c2)){
        c1.print();
        std::cout << " = " ;
        c2.print();
        std::cout << "\n";
    }
    else{
        c1.print();
        std::cout << " != " ;
        c2.print();
        std::cout << "\n";
    }

    c1 = c;
    if(c1.equm(c2)){
        c1.print();
        std::cout << " = " ;
        c2.print();
        std::cout << " (absolutely)\n";
    }
    else{
        c1.print();
        std::cout << " != " ;
        c2.print();
        std::cout << " (absolutely)\n";
    }

    std::cout << "Conj for ";
    c.print();
    std::cout << " is: ";
    complex cconj = c.conj();
    cconj.print();
    std::cout << "\n";
    std::cout << "Continue?\ny/n: ";
    std::cin >> option;
    while(option != 'y' && option != 'n'){
        std::cout << "Continue?\ny/n: ";
        std::cin >> option;
    };
    std::cout << "Shutting down...\n";
}
}

```

