

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

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

Студент Ханнанов Руслан Маратович, группа М8О-208Б-20

Преподаватель Дорохов Евгений Павлович

Условие

Задание: Вариант 2: класс - 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. Исходный код лежит в файле main.cpp

Протокол работы

3 60

2 15

Complex num in trigonometric form: $3*(\cos 60 + i * \sin 60)$

Complex num in trigonometric form: $2*(\cos 15 + i * \sin 15)$

Сложение: Complex num in trigonometric form: $4.39362*(\cos -185.045 + i * \sin -185.045)$

Вычитание: Complex num in trigonometric form: $2.58769*(\cos -121.102 + i * \sin -121.102)$

Умножение: Complex num in trigonometric form: $6*(\cos 75 + i * \sin 75)$

Деление: Complex num in trigonometric form: $1.5*(\cos 45 + i * \sin 45)$

Проверка на равенство: 0

Сопряженное число: Complex num in trigonometric form: $3*(\cos -60 + i * \sin -60)$

Проверка на равенство по действительной части: 0

Дневник отладки

Проблем и ошибок при написании данной работы не возникло.

Недочёты

Выводы

В процессе выполнения данной лабораторной работы, я смог применить начальные знания о классах, реализовал простой класс тригонометрического числа. Мне кажется, что классы удобны для вынесения некоторых абстрактных объектов в структуру с определенным набором полей и методов. Это позволяет сделать код компактнее и более понятным для стороннего наблюдателя.

Исходный код:

main.cpp

```
#include <iostream>
#include <cmath>

class Complex{

public:
    Complex();
    Complex(double r, double j);

    Complex add(Complex x); //Сложение
    Complex sub(Complex x); //Вычитание
    Complex mul(Complex x); //Умножение
    Complex div(Complex x); //Деление
    bool equ(Complex x); //Сравнение
    bool equ_by_real(Complex x); //Сравнение по действительной части
    Complex conj(); //Сопряженное число
    Complex to_alg(Complex x); //Перевод в алгебраический вид
    Complex to_trig(Complex x); //Перевод в тригонометрический вид
    friend std::istream& operator>> (std::istream &in, Complex &num); //Перегрузка оператора
    friend std::ostream& operator<< (std::ostream &out, const Complex &num); //Перегрузка оператора

private:
    double r,j;
};

Complex::Complex(double r, double j) {
    this->r = r;
    this->j = j;
}

Complex Complex::to_alg(Complex x) {
    return {x.r * cos(x.j), x.r * sin(x.j)};
}

Complex Complex::to_trig(Complex x) {
    double a = x.r, b = x.j;
    double z = sqrt(a * a + b * b);
    double argZ = (-3.14 + atan(b / a)) * (180 / 3.14);
    return {z, argZ};
}
```

```

}

Complex Complex::add(Complex x) {
    Complex alg_form1 = to_alg(*this);
    Complex alg_form2 = to_alg(x);
    Complex result(alg_form1.r + alg_form2.r, alg_form1.j + alg_form2.j);
    return to_trig(result);
}

Complex Complex::sub(Complex x) {
    if(this->r == x.r && this->j == x.j) return {0, 0};
    Complex alg_form1 = to_alg(*this);
    Complex alg_form2 = to_alg(x);
    Complex result(alg_form1.r - alg_form2.r, alg_form1.j - alg_form2.j);
    return to_trig(result);
}

Complex Complex::mul(Complex x) {
    return {this->r * x.r, this->j + x.j};
}

Complex Complex::div(Complex x) {
    if(x.r == 0 && x.j == 0){
        std::cout << "На 0 делить нельзя!" << std::endl;
        return *this;
    }
    return {this->r / x.r, this->j - x.j};
}

bool Complex::equ(Complex x) {
    return this->r == x.r && this->j == x.j;
}

Complex Complex::conj() {
    return {this->r, -this->j};
}

bool Complex::equ_by_real(Complex x) {
    const double e = 1e-5;
    Complex alg_form1 = to_alg(*this);
    Complex alg_form2 = to_alg(x);
    return (std::abs(alg_form1.r - alg_form2.r) < e);
}

```

```

}

Complex::Complex() {
    this->r = 0;
    this->j = 0;
}

std::ostream &operator<<(std::ostream &out, const Complex &num) {
    out << "Complex num in trigonometric form: " << num.r << "*(cos" << num.j << " + i * s
    return out;
}

std::istream &operator>>(std::istream &in, Complex &num) {
    in >> num.r >> num.j;
    return in;
}

int main() {
    Complex test_num1, test_num2;

    std::cin >> test_num1;
    std::cin >> test_num2;

    std::cout << test_num1 << test_num2 << std::endl;

    //Все операции
    std::cout << "Сложение: " << test_num1.add(test_num2) << std::endl;
    std::cout << "Вычитание: " << test_num1.sub(test_num2) << std::endl;
    std::cout << "Умножение: " << test_num1.mul(test_num2) << std::endl;
    std::cout << "Деление: " << test_num1.div(test_num2) << std::endl;
    std::cout << "Проверка на равенство: " << test_num1.equ(test_num2) << std::endl;
    std::cout << "Сопряженное число: " << test_num1.conj() << std::endl;
    std::cout << "Проверка на равенство по действительной части: " << test_num1.equ_by_rea

    return 0;
}

```