

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

ЛАБОРАТОРНАЯ РАБОТА №2
по курсу объектно-ориентированное программирование I семестр, 2021/22
уч. год

Студент: Сафонникова Анна Романовна, группа М8О-207Б-20

Преподаватель: Дорохов Евгений Павлович

Условие

Вариант 2

Комплексное число в тригонометрической форме представляются парой действительных чисел (r, j) , где r – радиус (модуль), j – угол. Реализовать класс `Complex` для работы с комплексными числами. Обязательно должны быть присутствовать операции:

- сложения $\text{add}, (r1, j1) + (r2, j2)$;
- вычитания $\text{sub}, (r1, j1) - (r2, j2)$;
- умножения $\text{mul}, (r1, j1) \cdot (r2, j2)$;
- деления $\text{div}, (r1, j1) / (r2, j2)$;
- сравнение $\text{eq}, (r1, j1) = (r2, j2)$, если $(r1 = r2)$ и $(j1 = j2)$;
- сопряженное число $\text{conj}, \text{conj}(r, j) = (r, -j)$.

Реализовать операции сравнения по действительной части. Реализовать пользовательский литерал для работы с константами объектов созданного класса.

Описание программы

Исходный код лежит в следующих файлах:

1. `main.cpp`: тестирование кода
2. `Complex.h`: описание класса комплексных чисел
3. `Complex.cpp`: реализация класса комплексных чисел

Дневник отладки

Ошибок не было.

Недочёты

Недочётов не заметила.

Вывод

В данной лабораторной работе я реализовала пользовательский литерал для работы с константами объектов созданного класса из предыдущей работы. Знакомство с ООП продолжается.

Исходный код

main.cpp

```
#include <iostream>
#include "Complex.h"

using namespace std;

Complex operator "" _i(long double i)
{
    return Complex(0,i);
};

Complex readFromConsole()
{
    double r;
    double j;
    cin>>r;
    cin>>j;
    return Complex(r,j);
}

void help()
{
    cout<< "Supported commands:\n"
    "a r1 j1 r2 j2 (add two complex numbers)\n"
    "s r1 j1 r2 j2 (sub two complex numbers)\n"
    "m r1 j1 r2 j2 (mul two complex numbers)\n"
    "d r1 j1 r2 j2 (div two complex numbers)\n"
    "e r1 j1 r2 j2 (check equality of two complex numbers)\n"
    "c r1 j1 (get conjugacy complex number)\n"
    "r r1 j1 r2 j2 (comparison by real part)\n"
    "q (quit program)\n";
}

void Test1()
{
    Complex arg1 = 1.0_i + 0.5;
    Complex arg2 = 0.5_i + 0.25;
    Complex res = arg1.Add(arg2);
    cout<<'('<<res.r<<','<<res.j<<')'<<endl;
}
```

```

int main(int argc, char *argv[])
{
    Test1();
    help();
    char run=1;
    while(run)
    {
        cout<<"> ";
        char cmd;
        cin>>cmd;
        switch(cmd)
        {
            case 'a':
            {
                Complex res = readFromConsole().Add(readFromConsole());
                cout<<'('<<res.r<<', '<<res.j<<')'<<endl;
                break;
            }
            case 's':
            {
                Complex res = readFromConsole().Sub(readFromConsole());
                cout<<'('<<res.r<<', '<<res.j<<')'<<endl;
                break;
            }
            case 'm':
            {
                Complex res = readFromConsole().Mul(readFromConsole());
                cout<<'('<<res.r<<', '<<res.j<<')'<<endl;
                break;
            }
            case 'd':
            {
                Complex res = readFromConsole().Div(readFromConsole());
                cout<<'('<<res.r<<', '<<res.j<<')'<<endl;
                break;
            }
            case 'e':
            {
                char res = readFromConsole().Equ(readFromConsole());
                cout<<(res ? "equals":"not equals")<<endl;
                break;
            }
        }
    }
}

```

```

    }
    case 'c':
    {
        Complex res = readFromConsole().Conj();
        cout<<'('<<res.r<<','<<res.j<<')'<<endl;
        break;
    }
    case 'r':
    {
        char res = readFromConsole().EquR(readFromConsole());
        cout<<(res ? "equals":"not equals")<<endl;
        break;
    }
    case 'q':
    {
        run=0;
        break;
    }
    default:
    {
        cout<<"There is no such command\n"<<endl;
        help();
    }
}

return 0;
}

```

Complex.h

```

#ifndef COMPLEX_H
#define COMPLEX_H

class Complex
{
public:
    double r;
    double j;
    Complex(double _r, double _j);
    Complex Add(Complex arg);
    Complex Sub(Complex arg);
    Complex Mul(Complex arg);
    Complex Div(Complex arg);

```

```

        char Equ(Complex arg);
        Complex Conj();
        char EquR(Complex arg);
        const Complex operator+ (const double another);
    private:
        const double PI = 3.14159265358979323846;
        Complex a2t(Complex arg);
        Complex t2a(Complex arg);
        double sign(double arg);
};

#endif

```

Complex.cpp

```

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

Complex::Complex(double _r, double _j)
{
    r = _r;
    j = _j;
}

Complex Complex::Add(Complex arg)
{
    Complex a1 = t2a(arg);
    Complex a2 = t2a(*this);
    return a2t(Complex(a1.r+a2.r,a1.j+a2.j));
}

Complex Complex::Sub(Complex arg)
{
    Complex a1 = t2a(arg);
    Complex a2 = t2a(*this);
    return a2t(Complex(a1.r-a2.r,a1.j-a2.j));
}

Complex Complex::Mul(Complex arg)
{
    Complex a1 = t2a(arg);
    Complex a2 = t2a(*this);
    return a2t(Complex(a1.r*a2.r-a1.j*a2.j,a1.r*a2.j+a1.j*a2.r));
}

```

```

}

Complex Complex::Div(Complex arg)
{
    Complex a1 = t2a(arg);
    Complex a2 = t2a(*this);
    double sqr = a2.r*a2.r+a2.j*a2.j;
    return a2t(Complex((a1.r*a2.r+a1.j*a2.j)/sqr,(a1.j*a2.r-a1.r*a2.j)/sqr));
}

char Complex::Equ(Complex arg)
{
    Complex a1 = t2a(arg);
    Complex a2 = t2a(*this);
    return a1.r==a2.r && a1.j==a2.j;
}

Complex Complex::Conj()
{
    return Complex(r,-j);
}

char Complex::EquR(Complex arg)
{
    Complex a1 = t2a(arg);
    Complex a2 = t2a(*this);
    return a1.r == a2.r;
}

Complex Complex::t2a(Complex arg)
{
    return Complex(arg.r*cos(arg.j),arg.r*sin(arg.j));
}

Complex Complex::a2t(Complex arg)
{
    double z = sqrt(arg.r*arg.r+arg.j*arg.j);
    double za = 0;
    double a = arg.r;
    double b = arg.j;
    if(a>0)
    {

```

```

        za = atan(b/a)*sign(b);
    }
    else if(a<0)
    {
        za = (PI-atan(b/a))*sign(b);
    }
    else
    {
        za = (PI/2)*sign(b);
    }
    return Complex(z,za);
}

double Complex::sign(double arg)
{
    if(arg>0)
        return 1;
    if(arg<0)
        return -1;
    return 0;
}

const Complex Complex::operator+ (const double real)
{
    return Complex(real, j);
}

```