

Московский Авиационный Институт
(Национальный исследовательский Университет)

Факультет: «Информационные технологии и прикладная математика»
Кафедра: 806 «Вычислительная математика и программирование»

**Лабораторная работа №1
по курсу «ООП»**

**Тема:
Простые классы.**

Студент:	Рыженко И.А.
Группа:	М80-208Б-18
Преподаватель:	Журавлев А. А.
Вариант:	22
Оценка:	
Дата:	11.01.2020

Москва
2020

1. Код программы на языке C++:

complex.h:

```
#pragma once
#include <iostream>

class Complex_number
{
private:
    double r;
    double phi;
public:
    Complex_number();//конструктор по умолчанию
    Complex_number(double r, double phi);//конструктор
    Complex_number(const Complex_number &complex);
    ~Complex_number();
    //getters и setters
    double get_module() const;
    double get_angle() const;
    static Complex_number sum(const Complex_number &,const Complex_number
&);
    static Complex_number sub(const Complex_number &,const Complex_number
&);
    static Complex_number mul(const Complex_number &,const Complex_number
&);
    static Complex_number div(const Complex_number &,const Complex_number
&);
    static bool equ(const Complex_number &,const Complex_number &);//сравне-
ние
    static bool equ_rational(const Complex_number &,const Complex_number
&);//сравнение по действительной части
    Complex_number conj() const;
    void print(std::ostream& out) const;
    void read(std::istream& in);
};
```

complex.cpp:

```
#include "complex.h"
#include <iostream>
#include <math.h>

Complex_number::~Complex_number(){}
Complex_number::Complex_number()
{
    this->r = 0;
    this->phi = 0;
}

Complex_number::Complex_number(double r, double phi) //угол в радианах
{
    this->r = r;
    while ((phi - 2*M_PI)>0)
        phi -= 2*M_PI;
    this->phi = phi;
}

double Complex_number::get_module() const
{
    return this->r;
}

double Complex_number::get_angle() const
{
    return this->phi;
}

Complex_number Complex_number::sum(const Complex_number &A,const Complex_number
&B)
{
    double Im = A.r*sin(A.phi)+B.r*sin(B.phi);//мнимая часть
    double Re = B.r*sin(B.phi)+B.r*sin(B.phi);//действительная часть
    double module = pow(Im,2) + pow(Re,2);//модуль комплексного числа
    double result_angle = atan2(Im,Re)/(2*M_PI);
    Complex_number result(module,result_angle);
    return result;
}

Complex_number Complex_number::sub(const Complex_number &A,const Complex_number
&B)
{
    double Im = A.r*sin(A.phi)-B.r*sin(B.phi);//мнимая часть
    double Re = B.r*sin(B.phi)-B.r*sin(B.phi);//действительная часть
```

```

        double module = pow(Im,2) + pow(Re,2); //модуль комплексного числа
        double result_angle = atan2(Im,Re)/(2*M_PI);
        Complex_number result(module,result_angle);
        return result;
    }

Complex_number Complex_number::mul(const Complex_number &A,const Complex_number
&B)
{
    double module = A.r*B.r; //модуль комплексного числа
    double result_angle = A.phi + B.phi;
    Complex_number result(module,result_angle);
    return result;
}

Complex_number Complex_number::div(const Complex_number &A,const Complex_number
&B)
{
    double module = A.r/B.r; //модуль комплексного числа
    double result_angle = A.phi - B.phi;
    Complex_number result(module,result_angle);
    return result;
}

bool Complex_number::equ(const Complex_number &A,const Complex_number &B)
{
    return (A.r == B.r && A.phi == B.phi) ? true: false;
}

Complex_number Complex_number::conj() const
{
    Complex_number result(this->r,-this->phi);
    return result;
}

bool Complex_number::equ_rational(const Complex_number &A, const Complex_number
&B)
{
    return (A.r == B.r) ? true: false;
}

void Complex_number::print(std::ostream& out) const
{
    std::cout << "Module of complex number: " << this->r << "\n";
    std::cout << "Angle of complex number: " << this->phi << "\n";
}

void Complex_number::read(std::istream& in)
{
    std::cout << "Module of complex number: ";
    in >> this->r;
    std::cout << "Angle of complex number: ";
    in >> this->phi;
}

```

main.cpp:

```

#include <iostream>
#include "complex.h"

int main()
{
    Complex_number a;
    Complex_number b;
    std::cout << "Input the first complex number: \n";
    a.read(std::cin);
    a.print(std::cout);
    std::cout << "Input the second complex number: \n";
    b.read(std::cin);
    b.print(std::cout);
    if (Complex_number::equ(a,b))
        std::cout << "Input numbers are equivalent\n";
    else
        std::cout << "Input numbers are not equivalent\n";
    if (Complex_number::equ_rational(a,b))
        std::cout << "Rational part of input numbers are equivalent\n";
    else
        std::cout << "Rational part of input numbers are not equivalent\n";
    std::cout << "The sum of input complex numbers:\n";
    Complex_number::sum(a,b).print(std::cout);
    std::cout << "The difference of input complex numbers:\n";
    Complex_number::sub(a,b).print(std::cout);
    std::cout << "The multiple of input complex numbers:\n";
    Complex_number::mul(a,b).print(std::cout);
    std::cout << "The private of input complex numbers:\n";
    Complex_number::div(a,b).print(std::cout);
}

```

```
} return 0;
```

CmakeLists.txt:

```
cmake_minimum_required(VERSION 2.8)
project(lab1)
set(SOURCE_EXE main.cpp)
set(SOURCE_LIB complex.cpp)
add_library(complex STATIC ${SOURCE_LIB})
add_executable(main ${SOURCE_EXE})
target_link_libraries(main bitstring)
```

2. Ссылка на репозиторий на GitHub.

https://github.com/THEproVANO/oop_exercise_01

3. Набор testcases.

Test 1:

Input the first complex number:
Module of complex number: 20
Angle of complex number: 45
Module of complex number: 20
Angle of complex number: 1.0177
Input the second complex number:
Module of complex number: 20
Angle of complex number: 45
Module of complex number: 20
Angle of complex number: 1.0177

Input numbers are equivalent

Rational part of input numbers are equivalent

The sum of input complex numbers:
Module of complex number: 40
Angle of complex number: 0.161972

The difference of input complex numbers:
Module of complex number: 0
Angle of complex number: 0

The multiple of input complex numbers:
Module of complex number: 400
Angle of complex number: 2.03541

The private of input complex numbers:
Module of complex number: 1
Angle of complex number: 0

Test 2:

Input the first complex number:
Module of complex number: 34
Angle of complex number: 23
Module of complex number: 34
Angle of complex number: 4.15044
Input the second complex number:
Module of complex number: 34
Angle of complex number: 78
Module of complex number: 34
Angle of complex number: 2.60178
Input numbers are not equivalent
Rational part of input numbers are equivalent
The sum of input complex numbers:
Module of complex number: 48.6123
Angle of complex number: 5.82051
The difference of input complex numbers:
Module of complex number: 47.5483
Angle of complex number: 6.07051
The multiple of input complex numbers:
Module of complex number: 1156
Angle of complex number: 0.469035
The private of input complex numbers:
Module of complex number: 1
Angle of complex number: 1.54867

Test 3:

Input the first complex number:
Module of complex number: 12
Angle of complex number: 45
Module of complex number: 12
Angle of complex number: 1.0177
Input the second complex number:
Module of complex number: 34
Angle of complex number: 56
Module of complex number: 34

Angle of complex number: 5.73452
Input numbers are not equivalent
Rational part of input numbers are not equivalent
The sum of input complex numbers:
Module of complex number: 36.1056
Angle of complex number: 6.24978
The difference of input complex numbers:
Module of complex number: 36.0054
Angle of complex number: 0.358599
The multiple of input complex numbers:
Module of complex number: 408
Angle of complex number: 0.469035
The private of input complex numbers:
Module of complex number: 0.352941
Angle of complex number: 1.56637

5. Объяснение результатов работы программы.

Со стандартного ввода программа считывает два комплексных числа в виде двух пар действительных чисел. Далее преобразует в «стандартный вид», чтобы угол φ был ≥ 0 и $< 2\pi$. Далее вызываются различные классы, в них мы передаем оба числа представленных double парами. Сами методы, такие как `sum()`, `mul()`, `div()`, `sub()`, реализованы согласно правилам арифметики. Преобразования происходят согласно правилам операций с комплексными числами представленными в тригонометрической форме. После завершения работы программы на стандартный вывод подается результат выполнения арифметических операций, сопряженные числа и проверка на их равенство.

6. Вывод.

Выполняя данную лабораторную я получил опыт работы с простыми классами, с системой сборки Cmake, с системой контроля версий GitHub, а также изучил основы работы с классами в C++. Создал класс, соответствующий варианту моего задания, реализовал для него арифметические операции сложения, вычитания, умножения, деления, а также операции сравнения.