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
&);
        Complex_number operator+(const Complex_number &);
        static Complex_number sub(const Complex_number &,const Complex_number
&);
        Complex_number operator-(const Complex_number &);
        static Complex_number mul(const Complex_number &,const Complex_number
&);
        Complex_number operator*(const Complex_number &);
        static Complex_number div(const Complex_number &,const Complex_number
&);
        Complex_number operator/(const Complex_number &);
        static bool equ(const Complex_number &,const Complex_number &);//сравне-
ние
        bool operator==(const Complex_number &);
        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_number operator ""_Complex (long double first);
```

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_r) //угол в радианах
{
    this->r = r;
    if (phi_r < 0)
        while ((phi_r + 2*M_PI) < 0)
            phi_r = phi_r + 2*M_PI;
    else if (phi_r > 0)
        while ((phi_r - 2*M_PI) > 0)
            phi_r = phi_r - 2*M_PI;
    if (phi_r < 0)
        phi_r = 2*M_PI + phi_r;
    this->phi = phi_r;
}

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 = A.r*cos(A.phi)+B.r*cos(B.phi);//действительная часть
    double module = sqrt(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::operator+(const Complex_number &B)
{
    return sum(*this,B);
}

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 = A.r*cos(A.phi)-B.r*cos(B.phi);//действительная часть
    double module = sqrt(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::operator-(const Complex_number &B)
{
    return sub(*this,B);
}

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::operator*(const Complex_number &B)
{
    return mul(*this,B);
}

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;
}

Complex_number Complex_number::operator/(const Complex_number &B)
{
    return div(*this,B);
}

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

bool Complex_number::operator==(const Complex_number &B)
{
    return equ(*this,B);
}

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: ";
    double phi_r = 0;
    in >> phi_r;
    if (phi_r < 0)
        while ((phi_r + 2*M_PI) < 0)
            phi_r = phi_r + 2*M_PI;
    else if (phi_r > 0)
        while ((phi_r - 2*M_PI) > 0)
            phi_r = phi_r - 2*M_PI;
    if (phi_r < 0)
        phi_r = 2*M_PI + phi_r;
    this->phi = phi_r;
}

Complex_number operator ""_Complex (long double first)
{
    Complex_number result(first,0.0);
    return result;
}
```

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);
  std::cout << "Complex conjugate to the first number: \n";
  a.conj().print(std::cout);
  std::cout << "\n\n\n";
  std::cout << "\n\n\n";
  if (a == b)
      std::cout << "Input numbers are equivalent\n";
  else
      std::cout << "Input numbers are not equivalent\n";
  std::cout << "\n\n\n";
  std::cout << "The sum of input complex numbers:\n";
  (a+b).print(std::cout);
  std::cout << "\n\n\n";
  std::cout << "The difference of input complex numbers:\n";
  (a-b).print(std::cout);
  std::cout << "\n\n\n";
  std::cout << "The multiple of input complex numbers:\n";
  (a*b).print(std::cout);
  std::cout << "\n\n\n";
  std::cout << "The private of input complex numbers:\n";
  (a/b).print(std::cout);
  std::cout << "\n\n\n";
  auto x = 2.4_Complex;
  std::cout << "The value of literal:\n";
  x.print(std::cout);
  return 0;
}
```

CmakeLists.txt:

```
cmake_minimum_required(VERSION 2.8)

project(lab_2)

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 complex)
```

2. Ссылка на репозиторий на GitHub.

https://github.com/DeZellt/oop_exercise_02

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
Complex conjugate to the first number:
Module of complex number: 20
Angle of complex number: 5.26548
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
The value of literal:
Module of complex number: 2.4
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
Complex conjugate to the first number:
Module of complex number: 34
Angle of complex number: 2.13274

```
Input numbers are not 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
The value of literal:
Module of complex number: 2.4
Angle of complex number: 0
```

5. Объяснение результатов работы программы.

В консоли пользователь вводит два комплексных числа. Программа рассчитывает комплексно-сопряжённое первого числа, проверяет два числа на равенство, выводит их сумму, разность, произведение и частное. В конце выводится значение литерала.

6. Вывод.

В процессе выполнения данной работы произошло знакомство с перегрузкой операторов, приобретены навыки создания пользовательских литералов.