Московский Авиационный Институт
(Национальный исследовательский Университет)

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

Лабораторная работа 2
по курсу «ООП»

Тема:
Операторы. Литералы.

Студент:	Болдырев А.К.
Группа:	М8О-206Б-18
Преподаватель:	Журавлев А.А.
Вариант:	2
Оценка:	
Дата:	

Москва
2019

1. Код программы на языке C++:

complex.hpp

```
#ifndef __COMPLEX__
#define __COMPLEX__
#include <iostream>

class complex {

public:
complex();
complex(double a, double b);
complex conj(const complex& o) const;

friend complex operator + (const complex& , const complex& );
friend complex operator - (const complex& , const complex& );
friend complex operator * (const complex& , const complex& );
friend complex operator / (const complex& , const complex& );
friend std::ostream& operator « (std::ostream& out, const complex& complex);
friend void operator » (std::istream &in, complex& complex);
friend bool operator == (const complex& , const complex& );

private:
double r;
double fi;
};

complex operator ""_xn (unsigned long long first);

#endif
```

complex.cpp

```
#include "complex.hpp"
#include <cmath>

complex::complex() {
r = 0;
```

```
fi = 0;
}

complex::complex(double a, double b) {
if ( b >= 2 ) {
b = fmod( b, 2 );
} else if ( b < 0 ) {
b = fmod( b, 2 ) + 2;
}
r = a;
fi = b;
}

complex complex::conj(const complex& o) const {
complex result;
result.r = r;
result.fi = -fi;
return result;
}

std::ostream& operator « (std::ostream& out, const complex& complex) {
return out « complex.r « " " « complex.fi « "p";
}

void operator » (std::istream &in, complex& complex) {
in » complex.r » complex.fi;
if ( complex.fi >= 2 ) {
complex.fi = fmod( complex.fi, 2 );
} else if ( complex.fi < 0 ) {
complex.fi = fmod( complex.fi, 2 ) + 2;
}
}
bool operator == (const complex& a, const complex& b) {
if (a.r == b.r and a.fi == b.fi) {
return true;
}
return false;
}

complex operator + (const complex& a, const complex& b) {
complex result;
double cos_1 = cos ( a.fi * M_PI );
double cos_2 = cos ( b.fi * M_PI );
double sin_1 = sin ( a.fi * M_PI );
double sin_2 = sin ( b.fi * M_PI );
```

```
double a_1 = a.r * cos_1;
double a_2 = b.r * cos_2;
double b_1 = a.r * sin_1;
double b_2 = b.r * sin_2;
double a_r = a_1 + a_2;
double b_r = b_1 + b_2;
result.r = sqrt( a_r * a_r + b_r * b_r );
//ПЕРВАЯ ЧЕТВЕРТЬ
if (a_r / result.r > 0 and b_r /result.r >= 0) {
result.fi = ( acos ( a_r / result.r) ) / M_PI;
}
//ВТОРАЯ
if (a_r / result.r <= 0 and b_r /result.r > 0) {
result.fi = ( acos ( a_r / result.r) ) / M_PI;
}
//ТРЕТЬЯ
if (a_r / result.r >= 0 and b_r /result.r < 0) {
result.fi = ( asin ( b_r / result.r) ) / M_PI;
}
//ЧЕТВЕРТАЯ
if (a_r / result.r < 0 and b_r /result.r <= 0) {
result.fi = - ( acos ( a_r / result.r) ) / M_PI;
}
return result;
}

complex operator - (const complex& a, const complex& b) {
complex result;
double cos_1 = cos ( a.fi * M_PI );
double cos_2 = cos ( (b.fi - 1) * M_PI );
double sin_1 = sin ( a.fi * M_PI );
double sin_2 = sin ( (b.fi - 1) * M_PI );
double a_1 = a.r * cos_1;
double a_2 = b.r * cos_2;
double b_1 = a.r * sin_1;
double b_2 = b.r * sin_2;
double a_r = a_1 + a_2;
double b_r = b_1 + b_2;
result.r = sqrt( a_r * a_r + b_r * b_r );
//ПЕРВАЯ ЧЕТВЕРТЬ
if (a_r / result.r > 0 and b_r /result.r >= 0) {
result.fi = ( acos ( a_r / result.r) ) / M_PI;
}
//ВТОРАЯ
if (a_r / result.r <= 0 and b_r /result.r > 0) {
```

```
result.fi = ( acos ( a_r / result.r) ) / M_PI;
}
//ТРЕТЬЯ
if (a_r / result.r >= 0 and b_r /result.r < 0) {
result.fi = ( asin ( b_r / result.r) ) / M_PI;
}
//ЧЕТВЕРТАЯ
if (a_r / result.r < 0 and b_r /result.r <= 0) {
result.fi = - ( acos ( a_r / result.r) ) / M_PI;
}

return result;
}

complex operator * (const complex& a, const complex& b) {
complex result;
result.r = a.r * b.r;
result.fi = a.fi + b.fi;
return result;
}

complex operator / (const complex& a, const complex& b) {
complex result;
result.r = a.r / b.r;
result.fi = a.fi - b.fi;
return result;
}

complex operator"" _xn(unsigned long long first) {
complex P(first, 2);
return P;
}
```

main.cpp:

```
#include "complex.hpp"
#include <cmath>

int main() {
complex a, b, ans;
std::cin » a;
std::cin » b;
```

```
std::cout « "First number : " « a « "\n";
std::cout « "second number : " « b « "\n";
std::cout « " + = " « a + b « "\n";
std::cout « " - = " « a - b « "\n";
std::cout « " * = " « a * b « "\n";
std::cout « " / = " « a / b « "\n";
ans = a.conj(a);
std::cout « "conj = " « ans « "\n";
ans = b.conj(b);
std::cout « "conj = " « ans « "\n";
if ( a == b ) {
std::cout « "a == b\n";
} else {
std::cout « "a != b\n";
}
}
```

2. Ссылка на репозиторий на GitHub.

https://github.com/Anton-Boldyrev/oop_exercise_02

3. Набор test

Test_1.txt

```
1 0.2 1 0.2
First number : 1 0.2p
second number : 1 0.2p
+ = 2 0.2p
- = 1.57009e-16 -0.25p
* = 1 0.4p
/ = 1 0p
conj = 1 -0.2p
conj = 1 -0.2p
a == b
```

Test_2.txt

```
2 1.2 1 0.3
First number : 2 1.2p
second number : 1 0.3p
+ = 1.09351 -0.891194p
- = 2.96719 -0.76679p
* = 2 1.5p
/ = 2 0.9p
conj = 2 -1.2p
```

conj = 1 -0.3p
a != b

Test_3.txt
1 0.8 1.5 1.4
First number : 1 0.8p
second number : 1.5 1.4p
+ = 1.52412 -0.814495p
- = 2.04378 0.554068p
* = 1.5 2.2p
/ = 0.666667 -0.6p
conj = 1 -0.8p
conj = 1.5 -1.4p
a != b

Test_4.txt
1 -0.6 2 2.8
First number : 1 1.4p
second number : 2 0.8p
+ = 1.94009 0.963081p
- = 2.49721 -0.324368p
* = 2 2.2p
/ = 0.5 0.6p
conj = 1 -1.4p
conj = 2 -0.8p
a != b

Test_5.txt
1 -5.8 1 0.1
First number : 1 0.2p
second number : 1 0.1p
+ = 1.97538 0.15p
- = 0.312869 0.65p
* = 1 0.3p
/ = 1 0.1p
conj = 1 -0.2p
conj = 1 -0.1p
a != b

Test_6.txt
0.2 -0.025 1.5 3.02
First number : 0.2 1.975p
second number : 1.5 1.02p
+ = 1.3023 -0.973112p
- = 1.69824 0.0147178p

* = 0.3 2.995p
/ = 0.133333 0.955p
conj = 0.2 -1.975p
conj = 1.5 -1.02p
a != b

Test_7.txt
0.1 0.1 0.1 0.2
First number : 0.1 0.1p
second number : 0.1 0.2p
+ = 0.197538 0.15p
- = 0.0312869 -0.35p
* = 0.01 0.3p
/ = 1 -0.1p
conj = 0.1 -0.1p
conj = 0.1 -0.2p
a != b

4. Объяснение результатов работы программы.

При запуске нужно вписать два комплексных числа состоящих из пары чисел (радиус (r) и угол (fi)). Далее программа выводит сложение, вычитание, умножение, деление и сравнение, при помощи перегрузки операторов, то есть я просто пишу a + b и программа понимает что с этим надо делать. Также для объектов a и b вычисляется сопряженное число с помощью функции conj класса comlex.

5. Вывод.

Выполняя данную лабораторную я получил опыт работы с простыми классами, с системой сборки Cmake, с системой контроля версий Git, а также изучил основы работы с классами в C++. Перегрузил операторы сложения, вычитания, умножения, деления, ввода и вывода.