Отчёт по лабораторной работе №____02____ по курсу ___2

студента группы _____М80-208Б-18_____, № по списку _2

Адреса www, e-mail, jabber, skype _____alek.maria@yandex.ru

Работа выполнена: “7“ _____октября_____ 2019г.

1. **Тема**: ____Операции. Литералы_____

2. **Цель работы**: ____Изучение механизмов перегрузки операторов. Изучение механизмов работы с пользовательскими литералами._____

3. **Задание** (*вариант № 2*):

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

- сложения `add`, (r_1, φ_1) + (r_2, φ_2);
- вычитания `sub`, (r_1, φ_1) – (r_2, φ_2);
- умножения `mul`, (r_1, φ_1) * (r_2 φ_2);
- деления `div`, (r_1, φ_1) / (r_2, φ_2);
- операции сравнения `equ`, (r_1, φ_1) = (r_2, φ_2), если ($r_1 = r_2$) и ($\varphi_1 = \varphi_2$);
- сопряженное число `conj`, `conj`(r, φ) = (r, $-\varphi$).

Операции сложения, вычитания, умножения, деления, сравнения (на равенство, больше и меньше) должны быть выполнены в виде перегрузки операторов.
Необходимо реализовать пользовательский литерал для работы с константами типа `Complex`.

4. **Адрес репозитория на GitHub** https://github.com/PowerMasha/oop_exercise_02

5. **Код программы на C++**

main.cpp

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

int main(){

      Complex m1;
      Complex m2;
      Complex m3;
printf("Введите первое комплексное число\n");
   std::cin >> m1;


printf("Введите второе комплексное число\n");
   std::cin >> m2;


printf("Первое комплексное число\n");
std::cout << m1 <<std::endl;

printf("Второе комплексное число\n");
std::cout << m2<<std::endl;

std::cout << "Сумма:\n";
```

```
std::cout << m1 + m2 <<std::endl;


std::cout << "Разность:\n";
std::cout << m1 - m2<<std::endl ;

std::cout << "Произведение: \n";
std::cout << m1 * m2 <<std::endl;

std::cout << "Деление :\n";
std::cout << m1 / m2<<std::endl ;


std::cout << "Сравнение комплексных чисел по длине вектора и углу:\n";
if (m1 == m2)
	std::cout << "Комплексные числа равны\n";
else
	std::cout << "Комплексные числа не равны\n";
std::cout << "sopr_m1:\n";
std::cout << m1.sopr() <<std::endl;

std::cout << "sopr_m2:\n";
std::cout << m2.sopr()<<std::endl ;
std::cout << "Третье комплексное число:\n";
m3 = "[2:45]"_c;
  std::cout << m3 <<std::endl;

return (0);

}
```

Complex.cpp

```
#include "complex.h"
#include <cstring>
#include <sstream>
#include <cmath>

double PI=3.1415926535;


Complex::Complex(): arr{0,0} {}
Complex::Complex(double a,double b): arr{a, b} {}

double Complex::get(int i) const {
	return arr[i];
}

double Complex::cosi()const{
	double k;
	if (arr[1]==90 || arr[1]==270){
		k=0;}
			else{
			k=arr[0]*cos(arr[1]*PI/180);}
	return k;
}

double Complex::sini()const{
	double s;
	if (arr[1]==0 || arr[1]==180) {
		s=0;}
			else {
```

```
                    s=arr[0]*sin(arr[1]*PI/180);}
    return s;
}

Complex Complex::operator+ (const Complex& rhs) const
{
    Complex res= *this;
    double x1 = this->cosi();
    double y1 = this->sini();
    double x2 = rhs.cosi();
    double y2 = rhs.sini();
    double x=x1+x2;
    double y=y1+y2;

     res.arr[0]=std::sqrt(x*x+y*y);
    res.arr[1]=atan2(y,x);

    return res;
}

Complex Complex::operator- (const Complex& rhs)  const
{
    Complex res = *this;
    double x1 = this->cosi();
    double y1 = this->sini();
    double x2 = rhs.cosi();
    double y2 = rhs.sini();
    double x=x1-x2;
    double y=y1-y2;

     res.arr[0]=std::sqrt(x*x+y*y);
    res.arr[1]=atan2(y,x);
    return res;
}

Complex Complex::operator* (const Complex& rhs)  const
{
    Complex res = *this;
    res.arr[0] = arr[0]*rhs.arr[0];
    res.arr[1] = arr[1]+ rhs.arr[1];

    return res;
}

Complex Complex::operator/ (const Complex& rhs)  const
{
    Complex res = *this;
    if (rhs.arr[0]!=0){res.arr[0] = arr[0]/rhs.arr[0];}
    res.arr[1] = arr[1]-rhs.arr[1];
    return res;
}

bool Complex::operator== (const Complex& rhs) const
{
        return (arr[0]==rhs.arr[0] && arr[1]==rhs.arr[1]);
}


Complex Complex::sopr() const
{
    Complex sop{0,0};
    sop.arr[0]=arr[0];
```

```
        sop.arr[1]=-arr[1];
return sop;
}

Complex operator ""_c(const char* str, size_t size){
  std::istringstream is(str);
    char tmp;
    double c, z;
    is >> tmp >> c >> tmp >> z;
    return {c, z};
}

std::istream& operator>> (std::istream& in, Complex& rhs){
  in >> rhs.arr[0] >> rhs.arr [1];
  return in;
}

std::ostream& operator<< (std::ostream& out, const Complex& rhs)
{
        out << rhs.arr[0] <<"*(cos("<<rhs.arr[1]<<")+i*sin("<<rhs.arr[1]<<"))";
        return out;
}
```

Complex.h

```
#ifndef D_COMPLEX_H
#define D_COMPLEX_H

#include <iostream>

struct Complex{
  Complex();
  Complex(double a, double b);

  double get(int i);
  double cosi()const;
  double sini()const;
  Complex sopr() const;
  Complex operator+ (const Complex& rhs) const;
  Complex operator- (const Complex& rhs)  const;
  Complex operator* (const Complex& rhs)  const;
  Complex operator/ (const Complex& rhs) const;

  bool operator== (const Complex& rhs) const;
  friend std::istream& operator>> (std::istream& in, Complex& rhs);
  friend std::ostream& operator<< (std::ostream& out, const Complex& rhs);

public:
  double arr[2];

};

Complex operator ""_c(const char* str, size_t size);
#endif
```

CmakeLists.txt

```
project(2lab)

add_executable(oop_exercise_02
        main.cpp
        complex.cpp)
```

```
set(CMAKE_CXX_FLAGS
    "${CMAKE_CXX_FLAGS} -Wall -Wextra")
```

6. Набор testcases

test_01.txt	Ожидаемое действие	Ожидаемый результат
1 30 3.4 45	add((1, 30) ,(3.4, 45))	4.37 0.72
	sub((1, 30) ,(3.4, 45))	2.45 -2.25
	multiply((1, 30) ,(3.4, 45))	3.4*(cos(75)+i*sin(75))
	div((1, 30) ,(3.4, 45))	0,29*(cos(-15)+i*sin(-15))
	sravn((1, 30) ,(3.4, 45))	Длины не равны Углы не равны
	Сопряженные числа	(1, -30) (3.4, -45)

test_02.txt	Ожидаемое действие	Ожидаемый результат
2 90 1 30	add((2,90),(1,30))	2.64 1. 24
	sub((2,90),(1,30))	1.73 2.09
	multiply ((2,90),(1,30))	2*(cos(120)+i*sin(120))
	div((2,90),(1,30))	2*(cos(60)+i*sin(60))
	sravn((2,90),(1,30))	Длины не равны Углы не равны
	Сопряженные числа	(2, -90), (1, -30)

test_03.txt	Ожидаемое действие	Ожидаемый результат
5 45 3 180	add((5,45),(3,180))	3.57 1.42
	sub((5,45),(3,180))	7.43 0.49
	multiply ((5,45),(3,180))	15*(cos(225)+i*sin(225))
	div((5,45),(3,180))	1.66*(cos(-135)+i*sin(-135))
	sravn((5,45),(3,180))	Длины не равны Углы не равны
	Сопряженные числа	(5, -45) (3,-180)

7. Результаты выполнения тестов

masha@masha-VirtualBox:~/2kurs/oop_exercise_02/tmp$ ./oop_exercise_02 < ~/2kurs/oop_exercise_02/test_01.txt
Введите первое комплексное число
Введите второе комплексное число
Первое комплексное число
1*(cos(30)+i*sin(30))
Второе комплексное число
3.4*(cos(45)+i*sin(45))
Длина и угол(в радианах) вектора суммы:
4.37359*(cos(0.726186)+i*sin(0.726186))
Длина и угол(в радианах) вектора разности:
2.4478*(cos(-2.25026)+i*sin(-2.25026))
Произведение:
3.4*(cos(75)+i*sin(75))
Деление :
0.294118*(cos(-15)+i*sin(-15))
Сравнение комплексных чисел по длине вектора и углу:
Комплексные числа не равны
sopr_m1:
1*(cos(-30)+i*sin(-30))
sopr_m2:
3.4*(cos(-45)+i*sin(-45))
Третье комплексное число:
2*(cos(45)+i*sin(45))
masha@masha-VirtualBox:~/2kurs/oop_exercise_02/tmp$ ./oop_exercise_02 < ~/2kurs/oop_exercise_02/test_02.txt
Введите первое комплексное число
Введите второе комплексное число
Первое комплексное число
2*(cos(90)+i*sin(90))
Второе комплексное число
1*(cos(30)+i*sin(30))
Длина и угол(в радианах) вектора суммы:
2.64575*(cos(1.23732)+i*sin(1.23732))
Длина и угол(в радианах) вектора разности:
1.73205*(cos(2.0944)+i*sin(2.0944))
Произведение:

2*(cos(120)+i*sin(120))
Деление :
2*(cos(60)+i*sin(60))
Сравнение комплексных чисел по длине вектора и углу:
Комплексные числа не равны
sopr_m1:
2*(cos(-90)+i*sin(-90))
sopr_m2:
1*(cos(-30)+i*sin(-30))
Третье комплексное число:
2*(cos(45)+i*sin(45))
masha@masha-VirtualBox:~/2kurs/oop_exercise_02/tmp$ ./oop_exercise_02 < ~/2kurs/oop_exercise_02/test_03.txt
Введите первое комплексное число
Введите второе комплексное число
Первое комплексное число
5*(cos(45)+i*sin(45))
Второе комплексное число
3*(cos(180)+i*sin(180))
Длина и угол(в радианах) вектора суммы:
3.57586*(cos(1.42047)+i*sin(1.42047))
Длина и угол(в радианах) вектора разности:
7.43056*(cos(0.495885)+i*sin(0.495885))
Произведение:
15*(cos(225)+i*sin(225))
Деление :
1.66667*(cos(-135)+i*sin(-135))
Сравнение комплексных чисел по длине вектора и углу:
Комплексные числа не равны
sopr_m1:
5*(cos(-45)+i*sin(-45))
sopr_m2:
3*(cos(-180)+i*sin(-180))
Третье комплексное число:
2*(cos(45)+i*sin(45))

8. Объяснение результатов работы программы - вывод

В complex.h были заданы методы и свойства этого класса, а в complex.cpp они были описаны. Описанные методы использовались в файле main.cpp .

Применение перегрузки операторов в классах может существенно облегчить и ускорить процесс написания кода, однако, при неосторожном обращении, может запутать код и затруднить его чтение. Пользовательские литералы позволяют создавать объекты пользовательского типа посредством суффикса. Их использование может как повысить читаемость кода и упростить его написание, так и наоборот, при неумелом обращении.