Московский Авиационный Институт

(Национальный исследовательский Университет)

Факультет: «Информационные технологии и прикладная математика»

Кафедра: 806 «Вычислительная математика и программирование»

**Лабораторная работа**

**по курсу «ООП»**

**Тема:**

**Перегрузка операторов.**

|  |  |
| --- | --- |
| Студент: | Суворова С. А. |
| Группа: | М80-206Б-18 |
| Преподаватель: | Журавлев А.А. |
| Вариант: | 2 |
| Оценка: |  |
| Дата: |  |

Москва

2019

1.**Код на C++:**

complex.h:

#ifndef COMPLEX\_NEW

#define COMPLEX\_NEW

struct complex{

complex();

complex(double r1,double q1);

complex& operator\*=(const complex& o);

complex& operator/=(const complex& o);

double r;

double q;

};

std::istream& operator>> (std::istream& is,complex& m);

std::ostream& operator<< (std::ostream& os,const complex& m);

complex operator""\_angle(long double q);

complex operator+ (complex w1,complex w2);

complex operator- (complex w1,complex w2);

complex operator\* (complex w1,complex w2);

complex operator/ (complex w1,complex w2);

bool operator== (complex w1,complex w2);

complex cong(complex w1);

#endif

complex.cpp:

#include <iostream>

#include "complex.h"

#include <math.h>

complex::complex() {r=0;q=0;}

complex::complex(double r1, double q1){

r=r1;

q=q1;

}

complex& complex::operator\*=(const complex &o) {

r\*=o.r;

q+=o.q;

return \*this;

}

complex& complex::operator/=(const complex &o) {

r=r/o.r;

q=q-o.q;

return \*this;

}

std::istream& operator>> (std::istream& is,complex& m){

is >> m.r >> m.q;

return is;

}

std::ostream& operator<< (std::ostream& os,const complex& m){

os << m.r << " " << m.q <<"\n";

return os;

}

complex operator""\_angle(long double q){

complex d;

d.r=1;

d.q=q;

return d;

}

complex operator+ (complex w1,complex w2){

complex w3;

double a1,b1,a2,b2,a3,b3;

a1=w1.r\*cos(w1.q\*M\_PI);

b1=w1.r\*sin(w1.q\*M\_PI);

a2=w2.r\*cos(w2.q\*M\_PI);

b2=w2.r\*sin(w2.q\*M\_PI);

a3=a1+a2;

b3=b1+b2;

w3.r=sqrt(pow(a3,2)+pow(b3,2));

w3.q=(atan2(b3,a3))/M\_PI;

return w3;

}

complex operator- (complex w1,complex w2){

complex w3;

double a1,b1,a2,b2,a3,b3;

a1=w1.r\*cos(w1.q\*M\_PI);

b1=w1.r\*sin(w1.q\*M\_PI);

a2=w2.r\*cos(w2.q\*M\_PI);

b2=w2.r\*sin(w2.q\*M\_PI);

a3=a1-a2;

b3=b1-b2;

w3.r=sqrt(pow(a3,2)+pow(b3,2));

w3.q=(atan2(b3,a3))/M\_PI;

return w3;

}

complex operator\* (complex w1,complex w2){

complex w3;

w3=w1;

w3\*=w2;

return w3;

}

complex operator/ (complex w1,complex w2){

complex w3;

w3=w1;

w3/=w2;

return w3;

}

bool operator== (complex w1,complex w2){

if((w1.r==w2.r)&&(w1.q==w2.q)){

return true;

}else{

return false;

}

}

complex cong(complex w1){

complex w2;

w2.r=w1.r

w2.q=-w1.q;

return w2;

}

main.cpp:

#include <iostream>

#include "complex.h"

#include <math.h>

int main() {

complex w1;

complex w2;

std::cout << "Введите 2 комплексных числа" << std::endl;

std::cin >> w1 >> w2;

if((w1.r<0) || (w2.r<0)){

std::cout << "Модуль не может быть отрицательным " << std::endl;

}else {

std::cout << "+ :";

std::cout << w1+w2;

std::cout << "- :";

std::cout << w1-w2;

std::cout << "\* :";

std::cout << w1\*w2;

std::cout << "/ :";

std::cout << w1/w2;

std::cout << "Сравнение :";

if(w1==w2){

std::cout << "Равны" << "\n";

}else{

std::cout << "Не равны" << "\n";

}

std::cout << "Сопряженое первого комплексного числа :";

std::cout << -w1;

}

return 0;

}

2. **Ссылка на репозиторий в GitHub:**

https://github.com/Suvorova-Sofya/oop\_exercise\_02

3.**Набор testcases:**

test1:

Исходные данные:

1 2 3 4

Ожидаемый результат:

+ : 4 0

- : 2 0

\* : 3 6

/ : 0.333333 -2

Сравнение : Не равны

Сопряженое первого комплексного числа : 1 -2

test2:

Исходные данные:

0 1 0 1

Ожидаемый результат:

+ : 0 nan

- : 0 nan

\* : 0 2

/ : nan 0

Сравнение : Равны

Сопряженое первого комплексного числа : 0 -1

test3:

Исходные данные:

3 -3 4 -4

Ожидаемый результат:

+ : 1 0

- : 7 0

\* : 12 7

/ : 0.75 -1

Сравнение : Не равны

Сопряженое первого комплексного числа : 3 -3

test4:

Исходные данные:

-9 8 5 6

Ожидаемый результат:

Модуль не может быть отрицательным

4.**Результаты выполнения программы:**

test1:

Введите 2 комплексных числа  
+ :4 -1.36436e-16  
- :2 1  
\* :3 6  
/ :0.333333 -2  
Сравнение :Не равны  
Сопряженое первого комплексного числа :1 -2

test2:

Введите 2 комплексных числа  
0 1 0 1  
+ :0 1  
- :0 0  
\* :0 2  
/ :-nan 0  
Сравнение :Равны  
Сопряженое первого комплексного числа :0 -1test3:

Введите 2 комплексных числа  
Введите 2 комплексных числа  
3 -3 4 -4  
+ :1 2.72872e-16  
- :7 -1  
\* :12 -7  
/ :0.75 1  
Сравнение :Не равны  
Сопряженое первого комплексного числа :3 3

test4:

Введите 2 комплексных числа  
-9 8 5 6  
Модуль не может быть отрицательным

5**. Объяснение результатов работы программы:**

Программе задается 2 комплексных числа в тригонометрической форме, и она вычисляет сумму, разность, результат умножения, результат деления, равенство комплексных чисел и сопряженное первого комплексного числа.

6.**Вывод:**

В данной программе показывается каким образом можно перегрузить практически все существующие операторы в C++, чтобы потом их было более удобно использовать для специфичных типов.