1.**Код на C++:**

complex.h:

#ifndef COMPLEX\_NEW

#define COMPLEX\_NEW

struct complex{

complex();

complex(double r1,double q1);

void print\_complex();

void sum\_complex(complex a);

void diff\_complex(complex a);

void mult\_complex(complex a);

void divis\_complex(complex a);

void compar\_complex(complex a);

void conj\_complex();

private:

double r;

double q;

};

#endif

main.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;

}

void complex::print\_complex() {

std::cout << r << " " << q << std::endl;

}

void complex::sum\_complex(complex a) {

complex res;

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

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

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

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

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

a3=a1+a2;

b3=b1+b2;

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

res.q=atan(b3/a3);

std::cout << res.r << " " << res.q << std::endl;

}

void complex::diff\_complex(complex a) {

complex res;

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

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

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

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

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

a3=a1-a2;

b3=b1-b2;

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

res.q=atan(b3/a3);

std::cout << res.r << " " << res.q << std::endl;

}

void complex::mult\_complex(complex a) {

complex res;

res.r=a.r\*r;

res.q=a.q+q;

std::cout << res.r << " " << res.q << std::endl;

}

void complex::divis\_complex(complex a) {

complex res;

res.r=r/a.r;

res.q=q-a.q;

std::cout << res.r << " " << res.q << std::endl;

}

void complex::compar\_complex(complex a) {

if((r==a.r)&&(q==a.q)){

std::cout << "Равны" << std::endl;

} else{

std::cout << "Не равны" << std::endl;

}

}

void complex::conj\_complex() {

q=(-1)\*q;

}

int main() {

double a, b,c,d;

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

std::cin >> a >> b >> c >> d;

if((a<0) || (c<0)){

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

}else {

complex w1 = complex(a, b);

complex w2 = complex(c, d);

std::cout << "+ :";

w1.sum\_complex(w2);

std::cout << "- :";

w1.diff\_complex(w2);

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

w1.mult\_complex(w2);

std::cout << "/ :";

w1.divis\_complex(w2);

std::cout << "Сравнение :";

w1.compar\_complex(w2);

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

w1.conj\_complex();

w1.print\_complex();

}

return 0;

}

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

https://github.com/Suvorova-Sofya/oop\_exercise\_01

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  
1 2 3 4  
+ :4 -4.28626e-16  
- :2 -6.12323e-16  
\* :3 6  
/ :0.333333 -2  
Сравнение :Не равны  
Сопряженое первого комплексного числа :1 -2  
  
test2:

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

Введите 2 комплексных числа  
3 -3 4 -4  
+ :1 8.57253e-16  
- :7 4.37374e-16  
\* :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 работа с объектом: разделение прав доступа, создание и использование конструктора, способность с помощью методов класса работать с данными находящимися в private.