МИНИСТЕРСТВО ОБРАЗОВАНИЯ И НАУКИ РОССИЙСКОЙ ФЕДЕРАЦИИ

Федеральное государственное автономное образовательное учреждение высшего образования

«САНКТ-ПЕТЕРБУРГСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ

АЭРОКОСМИЧЕСКОГО ПРИБОРОСТРОЕНИЯ»

КАФЕДРА КОМПЬЮТЕРНЫХ ТЕХНОЛОГИЙ

И ПРОГРАММНОЙ ИНЖЕНЕРИИ (КАФЕДРА №43)

ОТЧЕТ

ЗАЩИЩЕН С ОЦЕНКОЙ

ПРЕПОДАВАТЕЛЬ:

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Старший преподаватель |  |  |  | Е. О. Шумова |
| должность, уч. степень, звание |  | подпись, дата |  | инициалы, фамилия |

ОТЧЕТ О ЛАБОРАТОРНОЙ РАБОТЕ №3

«Перегрузка операторов»

ПО ДИСЦИПЛИНЕ «ОБЪЕКТНО-ОРИЕНТИРОВАННОЕ ПРОГРАММИРОВАНИЕ»

РАБОТУ ВЫПОЛНИЛ

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| СТУДЕНТ ГР. | 4831 |  |  |  | К.А. Корнющенков |
| к |  |  | подпись, дата |  | инициалы, фамилия |

Санкт-Петербург 2020

Задание:

Изучить механизм перегрузки операторов для типов, определенных пользователем посредством использования методов класса и дружественных функций.

В работе необходимо реализовать класс в соответствии с вариантом задания и создать приложение. В классе должен быть предусмотрен конструктор для установки начальных значений полей, методы – члены класса и дружественные методы, обеспечивающие перегрузку операций для заданного класса.

Разработать класс «Комплексное число в тригонометрической форме». Определить в нем конструктор, перегрузить арифметические операции, операции сравнения, операцию преобразования в строку и метод получения комплексного числа из строки.

Листинг программы:

Main.cpp

#include <iostream>

#include <string>

#include "Complex.hpp"

using namespace std;

int main() {

Complex complex(2,3);

Complex complex2(1,2);

int choose;

bool check = true;

while (check) {

cin >> choose;

switch (choose) {

case 1:{

cout << "(" << complex.print() << ") - (" << complex2.print() << ") = " ;

Complex complex3 = complex - complex2;

cout << "(" << complex3.print() << ")"<< endl;

break;

}

case 2:{

cout << "(" << complex.print() << ") + (" << complex2.print() << ") = " ;

Complex complex3 = complex + complex2;

cout << "(" << complex3.print() << ")"<< endl;

break;

}

case 3:{

cout << "(" << complex.print() << ") / (" << complex2.print() << ") = " ;

Complex complex3 = complex / complex2;

cout << "(" << complex3.print() << ")"<< endl;

break;

}

case 4:{

cout << "(" << complex.print() << ") \* (" << complex2.print() << ") = " ;

Complex complex3 = complex \* complex2;

cout << "(" << complex3.print() << ")"<< endl;

break;

}

case 5:{

cout << "равенство: ";

cout << to\_string(complex = complex2) << endl;

break;

}

case 6:{

cout << "меньше: ";

cout << to\_string(complex < complex2) << endl;

break;

}

case 7:{

cout << "больше: ";

cout << to\_string(complex > complex2) << endl;

break;

}

case 8:{

cout << "вывод: " <<complex.print() << endl;

break;

}

case 9:{

string line;

cin >> line;

complex.fromString(line);

cout << complex.print() << endl;

break;

}

default:{

check = false;

}

}

}

return 0;

}

Complex.cpp

#include <cmath>

#include "Complex.hpp"

Complex operator/(Complex complex,Complex complex2){

double real = complex2.r/complex.r\*cos(complex2.coss-complex.coss);

double imag = complex2.r/complex.r\*sin(complex2.sinn-complex.sinn);

return Complex(real,imag);

}

void Complex::fromString(string value){

string line = "";

int count = 0;

for (int i=0;i<value.length();i++){

if (value[i]!=' ' && value[i]!='i' && value[i]!='+'){

line+= value[i];

}else{

if (count==0){

this->coss = atoi(line.c\_str());

count++;

}

line = "";

}

}

this->sinn = atoi(line.c\_str());

}

Complex::Complex(double a,double b){

this->a = a;

this->b = b;

this->r = sqrt(a\*a+b\*b);

this->coss = a/this->r;

this->sinn = b/this->r;

}

Complex::Complex(){

this->a = 0;

this->b = 0;

this->r = sqrt(a\*a+b\*b);

this->coss = a/this->r;

this->sinn = b/this->r;

}

string Complex::print(){

return to\_string(this->coss) + " + i" + to\_string(this->sinn);

}

Complex Complex::operator\*(Complex complex){

double real = this->r\*complex.r\*cos(this->coss+complex.coss);

double imag = r\*complex.r\*sin(this->sinn+complex.sinn);

return Complex(real,imag);

}

Complex Complex::operator-(Complex complex){

double real = this->r\*coss - complex.r\*complex.coss;

double imag = this->r\*sinn - complex.r\*complex.sinn;

return Complex(real,imag);

}

Complex Complex::operator+(Complex complex){

double real = this->r\*coss + complex.r\*complex.coss;

double imag = this->r\*sinn + complex.r\*complex.sinn;

return Complex(real,imag);

}

bool Complex::operator=(Complex complex){

if (this->r==complex.r){

return true;

}

return false;

}

bool Complex::operator>(Complex complex){

if (this->r>complex.r){

return true;

}

return false;

}

bool Complex::operator<(Complex complex){

if (this->r<complex.r){

return true;

}

return false;

}

Complex.hpp

#ifndef Complex\_hpp

#define Complex\_hpp

#include <stdio.h>

#include <string>

using namespace std;

class Complex{

private:

double a,b,r,coss,sinn;

public:

Complex(double a,double b);

Complex();

string print();

void fromString(string value);

Complex operator\*(Complex complex);

friend Complex operator/(Complex complex,Complex complex2);

Complex operator-(Complex complex);

Complex operator+(Complex complex);

bool operator=(Complex complex);

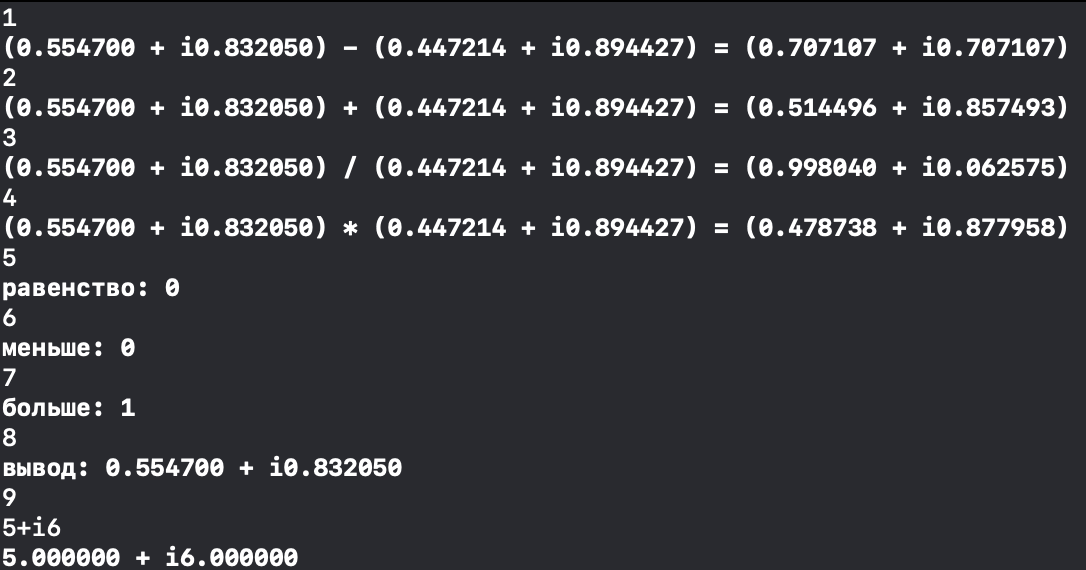
bool operator>(Complex complex);

bool operator<(Complex complex);

};

#endif /\* Complex\_hpp \*/

Скриншоты с результатами:



Вывод:

В ходе выполнения работы на практике были изучены операторы перегрузки функции и изучили механизм дружественных функций.