МИНИСТЕРСТВО ОБРАЗОВАНИЯ И НАУКИ РОССИЙСКОЙ ФЕДЕРАЦИИ

Федеральное государственное автономное образовательное учреждение

высшего профессионального образования

**«Дальневосточный федеральный университет»**

**ШКОЛА ЕСТЕСТВЕННЫХ НАУК**

**Кафедра информационных систем управления**

**ЛАБОРАТОРНАЯ РАБОТА № 8 ЗАДАЧА № 3**

По дисциплине «Объектно-ориентированное программирование»

На тему «Динамические структуры в языке C++»

|  |
| --- |
| Выполнил студент гр. Б8219 |
| О.В. Константинов |
|  |
| Проверил старший преподаватель |
| Г.Л. Берёзкина |
|  |
| (зачтено/не зачтено) |

г. Владивосток

2016

# Аннотация

Данный отчет подготовлен в рамках задания по дисциплине «Объектно-ориентированное программирование». Отчет призван систематизировать знания об использовании и реализации классов в языке C++. Отчет содержит список выполненных заданий с реализацией на языке C++ в среде программирования MS Visual Studio 2016.

# Задача 3

## Файл *“complex\_lib.h”*

#pragma once

#include <iostream>

#include <cmath>

class Complex {

private:

double re, im;

public:

Complex(double r = 0, double i = 0) : re(r), im(i) {}

Complex(const Complex &c) : re(c.re), im(c.im) {}

double abs() const { return sqrt(re \* re + im \* im); }

double real() const { return re; }

double imag() const { return im; }

Complex& operator = (const Complex &c) { re = c.re; im = c.im; return \*this; }

Complex& operator += (const Complex &c) { re += c.re; im += c.im; return \*this; }

Complex& operator -= (const Complex &c) { re -= c.re; im -= c.im; return \*this; }

Complex operator + (const Complex &c) const { return Complex(re + c.re, im + c.im); }

Complex operator - (const Complex &c) const { return Complex(re - c.re, im - c.im); }

Complex operator \* (const Complex &c) const {

return Complex(re \* c.re - im \* c.im, re \* c.im + im \* c.re);

}

Complex operator / (const Complex &c) const {

Complex temp;

double r = c.re \* c.re + c.im \* c.im;

temp.re = (re \* c.re + im \* c.im) / r;

temp.im = (im \* c.re - re \* c.im) / r;

return temp;

}

Complex pow(int n) const {

Complex temp;

double R = powl(sqrt(re \* re + im \* im), n);

double Phi = n \* atan2(im, re);

temp.re = R \* cos(Phi);

temp.im = R \* sin(Phi);

return temp;

}

friend std::ostream& operator << (std::ostream &, const Complex &);

friend std::istream& operator >> (std::istream &, Complex &);

};

std::ostream& operator << (std::ostream &out, const Complex &c) {

out << "(" << c.re << "," << c.im << ")";

return out;

}

std::istream& operator >> (std::istream &in, Complex &c) {

in >> c.re >> c.im;

return in;

}

## Файл *“main.cpp”*

/\*

Задача 3. Комплексное число, задаваемое своей вещественной и мнимой частями (например, 5+3i задается парой (5, 3)). Обязательны функции-члены класса для ввода/вывода числа, сложения, произведения двух чисел, а также возведения числа в целую степень.

\*/

#include "complex\_lib.h"

#include <complex>

using namespace std;

typedef bool(\*TEST\_FuncT)();

void TEST\_FUNCTION(TEST\_FuncT funcs[], int size) {

cout << "TEST COUNT " << size << endl;

for (int i = 0; i < size; i++) {

cout << "TEST " << i + 1 << " ";

funcs[i]() ? cout << "OK" : cout << "FAIL";

cout << endl;

}

cout << "TEST IS FINISHED" << endl;

}

inline double round(double x) { return floor(x + 0.5); }

bool operator == (const Complex &lhs, const complex<double> &rhs) {

#if \_DEBUG

cout << "A = " << lhs << " C = " << rhs << endl;

#endif

return round(lhs.real()) == round(rhs.real()) && round(lhs.imag()) == round(rhs.imag());

}

bool TEST\_Create() {

Complex c1, c2(1), c3(1, 2);

return c1 == complex<double>()

&& c2 == complex<double>(1)

&& c3 == complex<double>(1, 2);

}

bool TEST\_Assign() {

Complex c1(3, 8), c2(7, 5), c3(5, 6);

c1 = c1;

c1 = c2;

c2 = c3;

return c1 == complex<double>(7, 5)

&& c2 == complex<double>(5, 6);

}

bool TEST\_Pow() {

Complex c1(3, 8), c2(7, 5);

complex<double> d1(3, 8), d2(7, 5);

return c1.pow(3) == pow(d1, 3)

&& c2.pow(4) == pow(d2, 4);

}

bool TEST\_AbsRealImag() {

Complex c1(4, 5), c2(6, 8);

c1 = c1.abs();

return c1 == abs(complex<double>(4, 5))

&& c2.real() == 6 && c2.imag() == 8;

}

bool TEST\_AddSubtract() {

Complex c1(4, 5), c2(6, 8);

complex<double> d1(4, 5), d2(6, 8);

return c1 + c2 == d1 + d2

&& c1 - c2 == d1 - d2;

}

bool TEST\_MultiplyDivide() {

Complex c1(4, 5), c2(6, 8);

complex<double> d1(4, 5), d2(6, 8);

return c1 \* c2 == d1 \* d2

&& c1 / c2 == d1 / d2;

}

int main() {

TEST\_FuncT funcs[] = {

TEST\_Create,

TEST\_Assign,

TEST\_Pow,

TEST\_AbsRealImag,

TEST\_AddSubtract,

TEST\_MultiplyDivide

};

TEST\_FUNCTION(funcs, sizeof(funcs) / sizeof(TEST\_FuncT));

system("PAUSE");

return 0;

}