|  |  |
| --- | --- |
| Gerb-BMSTU_01 | **Министерство образования и науки Российской Федерации**  **Федеральное государственное бюджетное образовательное учреждение**  **высшего образования**  **«Московский государственный технический университет**  **имени Н.Э. Баумана**  **(национальный исследовательский университет)»**  **(МГТУ им. Н.Э. Баумана)** |

ФАКУЛЬТЕТ: Информатики и систем управления

КАФЕДРА: Компьютерные системы и сети

**ОТЧЕТ ПО УЧЕБНОЙ ПРАКТИКЕ**

Студент Боярских Никита Игоревич

*фамилия, имя, отчество*

Группа ИУ6-22

Тип практики Практикум по программированию

Название предприятия МГТУ им. Н.Э. Баумана

Студент **\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ \_\_\_\_БоярскихН. И. \_**

*подпись, дата фамилия, и.о.*

Руководитель практики **\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ \_\_Черноусова Т. Г. \_\_**

*подпись, дата фамилия, и.о.*

Оценка \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

*2017 г.*

**Оглавление**

Введение 3

1. Основная часть 4

1.1 Класс «Комплексное число»5

1.1.1 Алгоритм инициализации данных класса5

1.1.2 Алгоритмы перегруженных для класса операторов8

Заключение 9

Список литературы 9

Приложения10

1) Код тестирующей программы main.cpp10

2) Заголовочный файл класса «Комплексное число»Complex.h15

3) Реализация класса «Комплексное число» Complex.hpp16

4) Скриншот интерфейса программы 120

5) Скриншот интерфейса программы 220

**Введение**

**Задание:**

Выполнить структурную декомпозицию, разработать структурную схему и алгоритмы программ. Реализовать на С++ в консольном режиме.

Разработать программу, которая реализует операции над комплексными числами. Реализовать шаблон, позволяющий использовать целые и вещественные числа различного размера со следующими операциями: ввод чисел, их суммирование, вычитание и умножение на скаляр, а также вывод результатов операций на экран.

**Цель работы** – выполнение структурной декомпозиции, создание структурной схемы: диаграмм классов предметной области и схем алгоритмов программы, разработка, тестирование и отладка программы в редакторе vim.

**1. Основная часть**

Работа состоит из 2-х модулей: тестирующей программы и класса «Комплексное число».

(Скриншоты, подтверждающие корректность работы программы, представлены в приложениях №4 и №5)

**1.1 Класс «Комплексное число»**

Шаблонный класс. Позволяет работать со всеми целочисленными (int, long) и вещественными (float, double) типами данных языка C++. Реализует основные операции над комплексными числами:

* ввод
* сложение
* вычитание
* умножение на скаляр
* вывод результатов операций на экран

Поднимает исключение WrongFormat, конструктор которого принимает необязательным параметром строку для описания ошибки, в случае неверного формата строки, рассматриваемой в качестве комплексного чила.

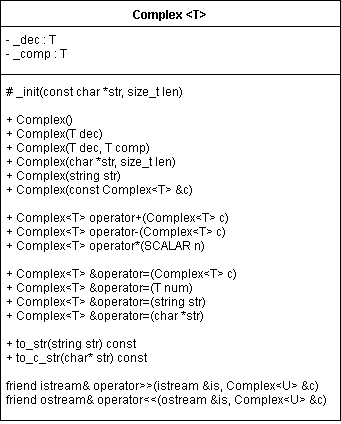


Рисунок 1 – диаграмма класса «Комплексное число» [1]

**1.1.1 Алгоритм инициализации данных класса**

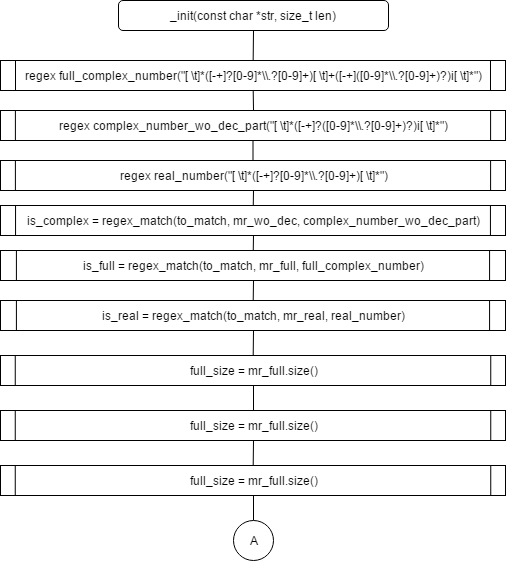
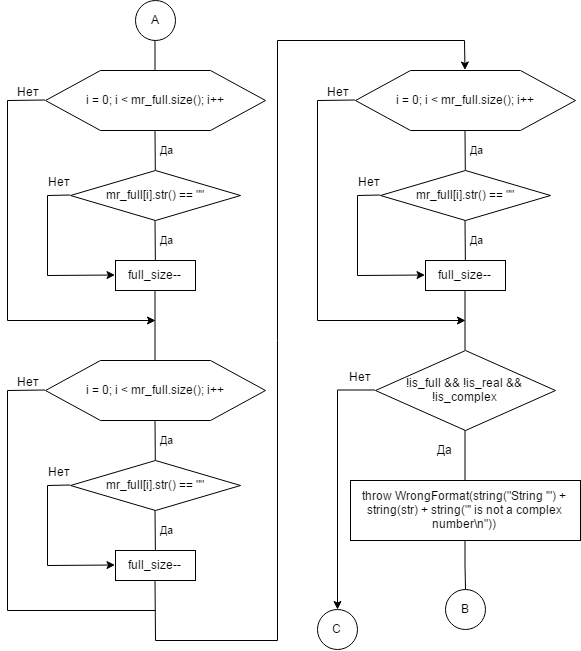
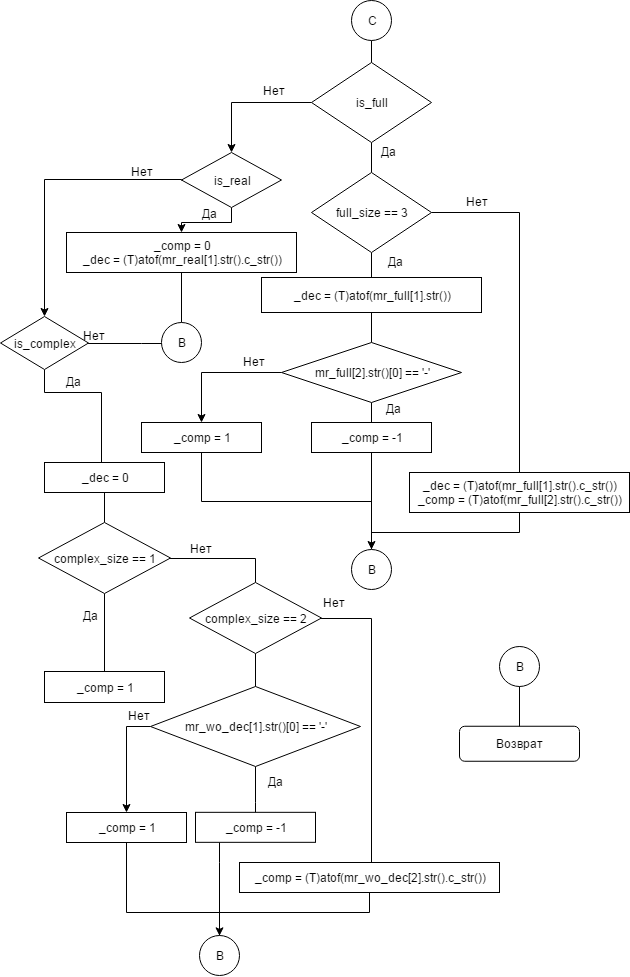


Рисунок 2 – Схема алгоритма инициализации данных класса (часть 1) [1]

Рисунок 3 – Схема алгоритма инициализации данных класса (часть 2) [1]

Рисунок 4 – Схема алгоритма инициализации данных класса (часть 3) [1]

**1.1.2 Алгоритмы перегруженных операторов класса**

|  |  |
| --- | --- |
| Рисунок 5 – Схема алгоритма оператора вычитания [1] | Рисунок 6 – Схема алгоритма оператора умножения на скаляр [1] |
| Рисунок 7 – Схема алгоритма оператора ввода [1] | Рисунок 8 – Схема алгоритма оператора вывода [1] |

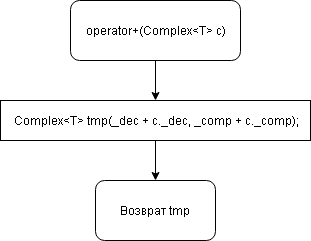


Рисунок 9 – Схема алгоритма оператора сложения [1]

**Заключение**

Была выполнена структурная декомпозиция, разработана структурная схема и схема алгоритмов программы. Разработана программа, которая реализует операции над комплексными числамив консольном режиме. Успешно проведено тестированиепрограммы, ошибок выявлено не было.

**Список литературы**

**Основная литература:**

* + - 1. д.т.н., проф. Иванова Галина Сергеевна: Лекции по объектно-ориентированному программированию МГТУ им. Н.Э. Баумана, «факультет Информатика и системы управления», кафедра «Компьютерные системы и сети». МОСКВА. 2017 год МГТУ им. Баумана

1. Подбельский В.В. Стандартный Си++: Учеб.пособие. – М.: Финансы и статистика, 2008.
2. Иванова Г.С., Ничушкина Т.Н. Объектно-ориентированное программирование. Учеб.для вузов. – М.: Изд-во МГТУ им. Н.Э. Баумана, 2014.

**Дополнительная литература:**

1. <http://www.cyberforum.ru/> (дата обращения 15.05.17)
2. <https://www.stackoverflow.com/> (дата обращения 15.05.17)

**Приложения**

1. **Код тестирующей программы main.cpp [4, 5]**

#include <iostream>

#include <cstdio>

#include <sstream>

#include <iomanip>

#include "Complex.h"

void clear\_stdin()

{

char c;

while( scanf("%c", &c) == 1 && c != '\n' );

}

int main() {

using std::cout;

using std::cin;

using std::endl;

using std::setw;

Complex <int> isum, idif, imul, ci1(-1);

Complex <float> fsum, fdif, fmul, cf1(0.0, 3.4);

Complex <double> dsum, ddif, dmul, cd1(999.99, -23.9402);

// Вывод значений средствами C++

cout<< "Инициализированы комплексные числа следующих типов:" <<endl;

cout << setw(20) << "Operation |" << setw(20) << "Int |"

<< setw(20) << "Float |" << setw(20) << "Double " << endl;

cout << setw(20) << std::setfill('-') << "+"<< setw(20) << "+" << setw(20)

<< "+" << setw(20) << "-" << endl << std::setfill(' ');

cout << setw(20) << "C1= |" << setw(18) << ci1 << " |" << setw(18) << cf1

<< " |" <<setw(20) <<cd1 <<endl;

cout<< "Введите ещё по 2 числа каждого из типов:" <<endl;

booldone = false;

// Ввод средствами C++

cout<< "Ввод первых чисел (C2=):" <<endl;

Complex<int> ci2;

Complex<float> cf2;

Complex<double> cd2;

while( ! done ) {

try {

cout<< "Введите комплексное число с целыми действительной и мнимой частью:" <<endl;

cin >> ci2;

done = true;

} catch( WrongFormat e ) {

(Продолжение кода на следующей странице)

(Продолжение кода)

cout<<e.msg<<endl;

done = false;

}

}

done = false;

while( ! done ) {

try {

cout<< "Введите комплексное число с рациональными действительной и мнимой частью:" <<endl;

cin >> cf2;

done = true;

} catch( WrongFormat e ) {

cout << e.msg << endl;

done = false;

}

}

done = false;

while( ! done ) {

try {

cout<< "Введите комплексное число с рациональными двойной точности действительной и мнимой частью:" <<endl;

cin >> cd2;

done = true;

} catch( WrongFormat e ) {

cout << e.msg << endl;

done = false;

}

}

// Ввод средствами C

char\* str = new char[255];

Complex<int>\* ci3;

done = false;

while( ! done ) {

try {

printf("Ввод вторых чисел (C3=):\n");

printf("Введите комплексное число с целыми действительной и мнимой частью:\n");

if( scanf("%255[^\n]", str) != 1 ) {

printf("Ошибка ввода!\n");

return 1;

}

clear\_stdin();

ci3 = new Complex<int>(str, (size\_t)255);

done = true;

} catch( WrongFormat e ) {

cout << e.msg << endl;

done = false;

}

}

(Продолжение кода на следующей странице)

(Продолжение кода)

Complex<float>\* cf3;

done = false;

while( ! done ) {

try {

cout<< "Введите комплексное число с рациональными действительной и мнимой частью:" <<endl;

if( scanf("%255[^\n]", str) != 1 ) {

printf("Ошибка ввода!\n");

return 1;

}

clear\_stdin();

cf3 = new Complex<float>(str, (size\_t)255);

done = true;

} catch( WrongFormat e ) {

cout << e.msg << endl;

done = false;

}

}

Complex<double>\* cd3;

done = false;

while( ! done ) {

try {

cout<< "Введите комплексное число с рациональными двойной точности действительной и мнимой частью:" <<endl;

if( scanf("%255[^\n]", str) != 1 ) {

printf("Ошибка ввода!\n");

return 1;

}

clear\_stdin();

cd3 = new Complex<double>(str, (size\_t)255);

done = true;

} catch( WrongFormat e ) {

cout << e.msg << endl;

done = false;

}

}

delete[] str;

cout<< "С введёнными числами будут произведены операции сложения, вычитания и умножения на скаляр." <<endl;

// Операция сложения со всеми типами

isum = ci2 + \*ci3;

fsum = cf2 + \*cf3;

dsum = cd2 + \*cd3;

// Операция вычитания со всеми типами

idif = ci2 - \*ci3;

fdif = cf2 - \*cf3;

ddif = cd2 - \*cd3;

(Продолжение кода на следующей странице)

(Продолжение кода)

// Операция умножения на число со всеми типами

doublenum;

cout<< "Введите число, с которым будет производиться операция умножения: N = ";

cin >> num; cin.ignore();

imul = ci1 \* num;

fmul = cf1 \* num;

dmul = cd1 \* num;

// Вывод результатов средствами C

constchardelimiter[] = "-------------------+-------------------+-------------------+--------------------\n";

const char format[] = "%19s|%19s|%19s|%20s\n";

printf(format, "Operation ", "Int ", "Float ", "Double ");

printf(delimiter);

char \*si = new char[20];

char \*sf = new char[20];

char \*sd = new char[20];

ci1.to\_c\_str(si);

cf1.to\_c\_str(sf);

cd1.to\_c\_str(sd);

printf(format, "=C1", si, sf, sd);

printf(delimiter);

ci2.to\_c\_str(si);

cf2.to\_c\_str(sf);

cd2.to\_c\_str(sd);

printf(format, "=C2", si, sf, sd);

printf(delimiter);

ci3->to\_c\_str(si);

cf3->to\_c\_str(sf);

cd3->to\_c\_str(sd);

printf(format, "=C3", si, sf, sd);

printf(delimiter);

printf("%19s|%30lf\n", "N=", num);

printf(delimiter);

isum.to\_c\_str(si);

fsum.to\_c\_str(sf);

dsum.to\_c\_str(sd);

printf(format, "=C2+C3", si, sf, sd);

printf(delimiter);

idif.to\_c\_str(si);

fdif.to\_c\_str(sf);

ddif.to\_c\_str(sd);

printf(format, "=C2-C3", si, sf, sd);

printf(delimiter);

(Продолжение кода на следующей странице)

(Продолжение кода)

imul.to\_c\_str(si);

fmul.to\_c\_str(sf);

dmul.to\_c\_str(sd);

printf(format, "=C1\*N", si, sf, sd);

printf(delimiter);

delete ci3;

delete cf3;

delete cd3;

delete[] si;

delete[] sf;

delete[] sd;

return 0;

}

1. **Заголовочный файл класса «Комплексное число» Complex.h [4, 5]**

#ifndef \_COMPLEX\_HEADER

#define \_COMPLEX\_HEADER

#include <iostream>

#include <string>

using std::istream;

using std::ostream;

using std::string;

class WrongFormat : public std::exception {

public:

string msg;

WrongFormat() { msg = "WrongFormat"; };

WrongFormat(string str) { msg = str; };

};

template<typenameT>

class Complex {

private:

T \_dec; // Действительная часть

T \_comp; // Мнимая часть

protected:

void \_init(const char \*str, size\_t len);

public:

// Конструкторы

Complex();

Complex(T dec); // comp = 0

Complex(T dec, T comp);

Complex(char \*str, size\_t len);

Complex(string str);

Complex(const Complex<T>&c);

// Арифметические операторы

Complex<T> operator+(Complex<T> c);

Complex<T> operator-(Complex<T> c);

template <typename SCALAR>

Complex<T> operator\*(SCALAR n);

// Операторы присваивания

Complex<T>&operator=(Complex<T> c);

Complex<T>&operator=(T num);

Complex<T>&operator=(string str);

Complex<T>&operator=(char \*str);

// Методы приведения типов

void to\_str(string str) const;

void to\_c\_str(char\* str) const;

// Дружественные операторы ввода-вывода

template<typenameU>

friend istream& operator>>(istream &is, Complex<U>&c);

template <typename U>

friend ostream& operator<<(ostream &os, Complex<U>&c);

};

#include "Complex.hpp"

#endif // \_COMPLEX\_HEADER

1. **Реализация класса «Комплексное число» Complex.hpp [4, 5]**

#ifndef \_COMPLEX\_

#define \_COMPLEX\_

//------------------------Подключение заголовочных файлов-----------------------

#include<string>

#include <iostream>

#include <sstream>

#include <regex>

#include<iomanip>

//-------------------------Импорт из пространства имён std----------------------

using std::istream;

using std::ostream;

using std::string;

using std::regex;

//---------------------------Реализация шаблонных методов-----------------------

//----------------------------------Конструкторы--------------------------------

template<classT>

void Complex<T>::\_init(const char \*str, size\_t len) {

regex full\_complex\_number(

"[ \t]\*([-+]?[0-9]\*\\.?[0-9]+)[ \t]+([-+]([0-9]\*\\.?[0-9]+)?)i[ \t]\*"

);

regex real\_number(

"[ \t]\*([-+]?[0-9]\*\\.?[0-9]+)[ \t]\*"

);

regex complex\_number\_wo\_dec\_part(

"[ \t]\*([-+]?([0-9]\*\\.?[0-9]+)?)i[ \t]\*"

);

std::smatch mr\_full;

std::smatch mr\_real;

std::smatch mr\_wo\_dec;

string to\_match(str);

bool is\_full = regex\_match(to\_match, mr\_full, full\_complex\_number);

bool is\_real = regex\_match(to\_match, mr\_real, real\_number);

bool is\_complex = regex\_match(to\_match, mr\_wo\_dec, complex\_number\_wo\_dec\_part);

size\_t full\_size = mr\_full.size();

size\_t real\_size = mr\_real.size();

size\_t complex\_size = mr\_wo\_dec.size();

for(size\_t i = 0; i < mr\_full.size(); i++) if(mr\_full[i].str() == "") full\_size--;

for(size\_t i = 0; i < mr\_real.size(); i++) if(mr\_real[i].str() == "") real\_size--;

for(size\_t i = 0; i < mr\_wo\_dec.size(); i++) if(mr\_wo\_dec[i].str() == "") complex\_size--;

(Продолжение кода на следующей странице)

(Продолжение кода)

if( !is\_full && !is\_real && !is\_complex ) {

throw WrongFormat(string("String '") + string(str) + string("' is not a complex number\n"));

} else {

if(is\_full) {

if(full\_size == 3) {

\_dec = (T)atof(mr\_full[1].str().c\_str());

\_comp = (mr\_full[2].str()[0] == '-') ? -1 : 1;

} else {

\_dec = (T)atof(mr\_full[1].str().c\_str());

\_comp = (T)atof(mr\_full[2].str().c\_str());

}

} else if (is\_real) {

\_comp = 0;

\_dec = (T)atof(mr\_real[1].str().c\_str());

} else if (is\_complex) {

\_dec = 0;

if(complex\_size == 1) {

\_comp = 1;

} else if(complex\_size == 2) {

\_comp = (mr\_wo\_dec[1].str()[0] == '-') ? -1 : 1;

} else {

\_comp = (T)atof(mr\_wo\_dec[2].str().c\_str());

}

}

}

}

template <class T>

Complex<T>::Complex() {

\_dec = 0;

\_comp = 0;

}

template <class T>

Complex<T>::Complex(T dec) {

\_dec = dec;

\_comp = 0;

}

template <class T>

Complex<T>::Complex(T dec, T comp) {

\_dec = dec;

\_comp = comp;

}

template <class T>

Complex<T>::Complex(string str) {

\_init(str.c\_str(), str.size());

}

(Продолжение кода на следующей странице)

(Продолжение кода)

template <class T>

Complex<T>::Complex(char \*str, size\_t len) {

\_init(str, len);

}

template <class T>

Complex<T>::Complex(const Complex<T>&c) {

\_dec = c.\_dec;

\_comp = c.\_comp;

}

//------------------------------Операторы------------------------------

template <class T>

Complex<T> Complex<T>::operator+(Complex<T> c) {

Complex<T> tmp(\_dec + c.\_dec, \_comp + c.\_comp);

return tmp;

}

template <class T>

Complex<T> Complex<T>::operator-(Complex<T> c) {

Complex<T> tmp(\_dec - c.\_dec, \_comp - c.\_comp);

return tmp;

}

template <typename T>

template <class SCALAR>

Complex<T> Complex<T>::operator\*(SCALAR n) {

Complex<T> tmp(\_dec \* n, \_comp \* n);

return tmp;

}

template <class T>

Complex<T>&Complex<T>::operator=(Complex<T> c) {

\_dec = c.\_dec;

\_comp = c.\_comp;

}

template <class T>

Complex<T>&Complex<T>::operator=(T num) {

\_dec = num;

\_comp = 0;

}

template <class T>

Complex<T>&Complex<T>::operator=(string str) {

\_init(str.c\_str(), str.size());

}

template <class T>

Complex<T>&Complex<T>::operator=(char \*str) {

\_init(str, strlen(str));

}

(Продолжение кода на следующей странице)

(Продолжение кода)

//---------------------Операторы приведения типов----------------------

template <class T>

void Complex<T>::to\_str(string str) const {

if(str.size() == 0) return;

char\* c\_str = new char[str.size()];

to\_c\_str(c\_str, str.size());

str = (const char\*)(c\_str);

delete[] c\_str;

}

template <>

void Complex<float>::to\_c\_str(char \*str) const {

sprintf(str, "%4.2f %+4.2fi", \_dec, \_comp);

}

template <>

void Complex<double>::to\_c\_str(char \*str) const {

sprintf(str, "%4.2lf %+4.2lfi", \_dec, \_comp);

}

template <>

void Complex<int>::to\_c\_str(char \*str) const {

sprintf(str, "%4d %+4di", \_dec, \_comp);

}

template <>

void Complex<long>::to\_c\_str(char \*str) const {

sprintf(str, "%4ld %+4ldi", \_dec, \_comp);

}

//------------------------Операторы ввода-вывода------------------------

template<classT>

istream &operator>>(istream &is, Complex<T>&c) {

char \*c\_str = new char[255];

is.getline(c\_str, 255);

string str(c\_str);

c = str;

is.clear();

return is;

}

template <class T>

ostream &operator<<(ostream &os, Complex<T>&c) {

char str[255] = {};

c.to\_c\_str(str);

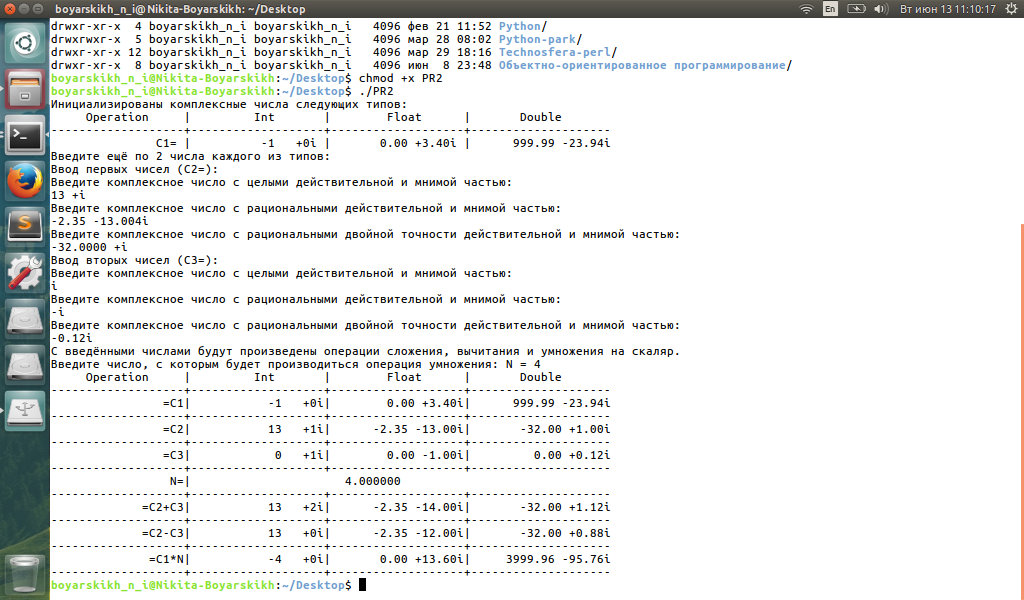
os << str;

return os;

}

#endif // \_COMPLEX\_

1. **Скриншот интерфейса программы 1**



1. **Скриншот интерфейса программы 2**

