|  |  |
| --- | --- |
| для прик эмбл | |
| МИНОБРНАУКИ РОССИИ | |
| Федеральное государственное бюджетное образовательное учреждениевысшего профессионального образования"Московский технологический университет"МИРЭА | |
| Факультет информационных технологий (ИТ) | |
| Кафедра практической и прикладной информатики | |

|  |  |
| --- | --- |
| **ОТЧЕТ ПО ЛАБОРАТОРНЫМ РАБОТАМ** | |
| **по дисциплине** | |
| **«**Объектно-ориентированное программирование**»** | |
|  | |
| Выполнил студент группы ИНБО-04-18 | Качура И.А |
| Принял ассистент кафедры | Хлебникова В.Л. |

|  |  |  |
| --- | --- | --- |
| Лабораторные работы выполнены | «30» 2019 г. |  |
|  |  |  |
| «Зачтено» | «\_\_»\_\_\_\_\_\_\_201\_\_ г. |  |

Москва 2019

## Таблица успеваемости

**Лабораторные занятия**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| №/р. | страница в отчете. | Студент выполнил:  (подпись) | Преподаватель принял:  (подпись) | Балл  (максимально возможный) | Балл (Фактический) |
| 1. |  |  |  | 4 |  |
| 2. |  |  |  | 4 |  |
| 3. |  |  |  | 4 |  |
| 4. |  |  |  | 4 |  |
| 5. |  |  |  | 4 |  |
| 6. |  |  |  | 4 |  |
| 7. |  |  |  | 4 |  |
| 8. |  |  |  | 4 |  |
|  |  |  | Итоговый балл: | 32 |  |

## Лабораторная работа №1

**Создание многофайловых проектов**

**Цель практической работы**

Целью данной лабораторной работы освоить на практике создание многофайловых проектов а языке Си/Си++, познакомиться с директивами условной компиляции.

**Задачи**

1. Написать программу – калькулятор комплексных чисел.
2. На основе задания 1 написать программу, считывающую комплексные числа из файла complex.txt и находящуюю в них число с наибольшим модулем.

**Ход работы**

Программа №1:

Начало формы

Конец формы

Начало формы

Конец формы

|  |  |
| --- | --- |
|  | #include <iostream> |
|  | #include "Complex.h" |
|  | using namespace std; |
|  |  |
|  | int main() { |
|  | double re, im; |
|  | cout << "- - - - - - - - - - - - - - - - -" << endl; |
|  | cout << "Введите действительную часть А: "; |
|  | cout << "- -" << endl; |
|  | cin >> re; |
|  | cout << "Введите мнимую часть А: "; |
|  | cin >> im; |
|  | Complex c(re, im); |
|  | cout << "- - - - - - - - - - - - - - - - -" << endl; |
|  | cout << "Введите действительную часть Б: "; |
|  | cout << "- -" << endl; |
|  | cin >> re; |
|  | cout << "Введите мнимую часть Б: "; |
|  | cin >> im; |
|  | Complex Ь(re, im); |
|  | cout << "- - - - - - - - - - - - - - - - -" << endl; |
|  | cout << "a + b = " << c + b << endl; |
|  | cout << "- -" << endl; |
|  | cout << "a - b = " << c - b << endl; |
|  | cout << "- -" << endl; |
|  | cout << "a \* b = " << c \* b << endl; |
|  | cout << "- -" << endl; |
|  | cout << "a / b = " << c / b << endl; |
|  | cout << "- -" << endl; |
|  | system("pause"); |
|  | return 0; |

1. Программа №2:

#include "stdafx.h"

#include <iostream>

#include "Complex.h"

#include <vector>

#include <fstream>

using namespace std;

int main() {

ifstream file("TestCompl.txt");

int n, r, m;

vector<Complex> p;

file >> n;

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

file >> r >> m;

Complex el(r, m);

p.push\_back(el);

}

file.close();

Complex max;

double md = p[0].module();

for (int i = 0; i < n; i++)

if (p[i].module() > md) {

md = p[i].module();

max = p[i];

}

cout << max.printRe() << ' ' << max.printIm() << endl;

return 0;

}

**Вывод**

Все поставленные задачи были выполнены. Для достижения требуемого результата был создан класс, разбитый на файлы с реализацией и объявлением. Также были использованы некоторые дерективы препрцессора для защиты от многократного подключения файлов и перегружены некоторые операторы. Результаты работы программ приведены на скриншотах (рис. 1, 2). Исходный код также доступен по ссылкам на GitHub:

1. <https://github.com/ivankachura/OOP/blob/master/%D0%9B%D0%B0%D0%B1.%D0%A0%D0%B0%D0%B1%D0%BE%D1%82%D0%B0%201/%D0%9B%D0%B0%D0%B1.%D0%A0%D0%B0%D0%B1%D0%BE%D1%82%D0%B0%201.1>

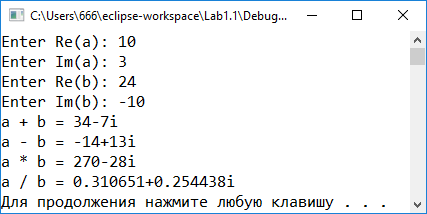
****

Рис. 1 Результат работы программы №1

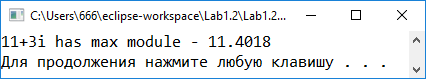


Рис. 2 Результат работы программы №2

## Лабораторная работа №2

**Указатели на функции**

**Цель практической работы**

Целью данной лабораторной работы освоить на практике вызов функции с использованием указателей.

**Задачи**

1. Напишите программу, которая вызывает различные виды функции в зависимости от заданного условия.
2. Напишите программу с использованием некой универсальной функции, которая в качестве возвращаемого значения возвращает указатель на функцию, зависящий от некоторого условия.v

**Ход работы**

1. Программа №1

#include <iostream>

using namespace std;

double summa(double a, double b) {

return a + b;

}

double proizv(double a, double b) {

return a \* b;

}

double raznost(double a, double b) {

return a - b;

}

double(\*p)(double a, double b);

int main()

{

double a, b;

cout << "- - - - - - - - - - -" << endl;

cout << "Введите А:" << endl;

cin >> a;

cout << "Введите B:" << endl;

cin >> b;

int z;

cout << "- - - - - - - - - - - - - - - -" << endl;

cout << "1-РАЗНОСТЬ 2-СУММА 3-ПРОИЗВ.:" << endl;

cin >> z;

switch (z)

{

case 1: p = raznost;

break;

case 2: p = summa;

break;

case 3: p = proizv;

break;

default:

break;

}

cout << "- - - - - - - - - - - - - - - -" << endl;

cout<<p(a, b)<<endl;

system("pause");

return 0;

}

1. Программа №2

#include "pch.h"

#include <iostream>

#include <ctime>

#include <string>

using namespace std;

int compare(const void \* x1, const void \* x2)

{

return (\*(int\*)x1 - \*(int\*)x2);

}

int comp2(const void \* a, const void \* b)

{

return (\*(int\*)b - \*(int\*)a);

}

cout << "\_ \_ \_ \_ \_ \_ \_ \_ \_ \_ \_ \_ \_ \_ \_ \_ \_ " << endl;

void f1(int \*a, int n) { for (int i = n - 1; i > -1; i--)cout << a[i]; cout << endl; }

void f2(int \*a, int n) { qsort(a, n, sizeof(int), compare); };

void f3(int \*a, int n) { qsort(a, n, sizeof(int), comp2); };

void (\*func(int\*a,int n))(int\*a,int n){

int sum=0;

for (int i = 1; i < n; i++)sum += a[i];

if (sum == a[0])return f1;

else if (sum > a[0])return f2;

else return f3;

}

int main()

{

cout << "\_ \_ \_ \_ \_ \_ \_ \_ \_ \_ \_ \_ \_ \_ \_ \_ \_ " << endl;

srand(time(0));

int n;

cin >> n;

int \*a= new int[n];

a[0] = 1000;

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

a[i] = rand() % 100+1;

cout << a[i] << endl;

}

cout << "\_ \_ \_ \_ \_ \_ \_ \_ \_ \_ \_ \_ \_ \_ \_ \_ \_ " << endl;

func(a, n)(a, n);

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

cout << a[i] << " ";

}

cout << endl;

delete[] a;

system("pause");

return 0;

}

**Вывод**

Все поставленные задачи были выполнены. Для достижения поставленных целей были использованы указатели на функции с различными возвращаемыми значениями. Результаты работы программ можно увидеть на скриншотах (рис. 3, 4). Исходный код программ также доступен по ссылкам на GitHub:

1. <https://github.com/ivankachura/OOP/blob/master/%D0%9B%D0%B0%D0%B1.%D0%A0%D0%B0%D0%B1%D0%BE%D1%82%D0%B0%202/%D0%9B%D0%B0%D0%B1.%D0%A0%D0%B0%D0%B1%D0%BE%D1%82%D0%B0%202.1>
2. <https://github.com/ivankachura/OOP/blob/master/%D0%9B%D0%B0%D0%B1.%D0%A0%D0%B0%D0%B1%D0%BE%D1%82%D0%B0%202/%D0%9B%D0%B0%D0%B1.%D0%A0%D0%B0%D0%B1%D0%BE%D1%82%D0%B0%202.2>

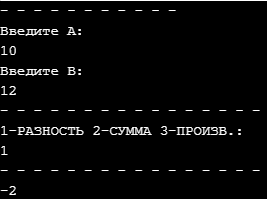


Рис. 3 Результат работы программы №1

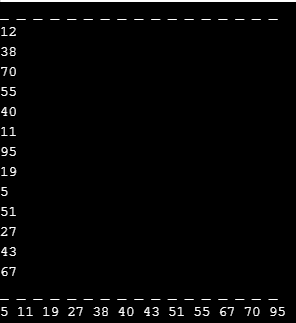


Рис. 4 Результат работы программы №2

## Лабораторная работа №3

**Классы**

**Цель практической работы**

Целью данной лабораторной работы является знакомство с классами в языке С++.

**Задачи**

1. Определить класс Child, который содержит такие поля (члены класса): закрытые — имя ребенка, фамилию и возраст, публичные — методы ввода данных и отображения их на экран. Объявить два объекта класса, внести данные и показать их.
2. Создать класс Tiles (кафель), который будет содержать поля с открытым доступом: brand, size\_h, size\_w, price и метод класса getData(). В главной функции объявить пару объектов класса и внести данные в поля. Затем отобразить их, вызвав метод getData().
3. Создать класс Complex, в котором реализовано комплексное число. В данном классе должны присутствовать методы, позволяющие рассчтать и вывеси модуль и аргументы данного числа.
4. Реализовать класс Vector, позволяющий хранить в себе математический вектор. В классе должно присутствовать метод позволяющей получить модуль вектора и методы, позволяющие складывать и вычитать разные векторы.

**Ход работы**

1. Программа №1

#include "pch.h"

#include <iostream>

#include <string>

using namespace std;

class Book

{

public:

string avtor;

string name;

double list;

void open();

};

int main()

{

Book Kniga;

Book.open();

return 0;

}

void Kniga::open()

{

double random;

cin >> avtor;

cin >> name;

random = 1 + rand() % 50;

cout << "Автор: " << avtor << " " << "Название: " << name << " " << "Страницы: " << random << endl;

}

1. Программы №2

#include <iostream>

#include <string>

using namespace std;

class Students

{

string \*members;

int n;

public:

void makestudents();

void getstudent();

};

void Students::makestudents()

{

cout << "Введите значение N:";

cin >> n;

members = new string[n];

for (int i = 0; i < n; i++)cin >> members[i];

}

void Students::getstudent()

{

for (int i = 0; i < n; i++)cout << members[i] << endl;

}

int main()

{

Students a;

a.makestudents();

a.getstudent();

system("pause");

return 0;

1. Программа №3

#include <iostream>

#include <string>

#include <ctime>

using namespace std;

class automobile

{

int speed;

int year;

string name;

public:

automobile()

{

year = 0;

speed = 0;

name = "Нету названия.";

}

void set\_automobile()

{

cout << "Введите год автомобиля: ";

cin >> year;

cout << "Введите скорость автомобиля: ";

cin >> speed;

cout << "Введите название автомобиля: ";

cin >> name;

}

void get\_automobile()

{

cout << "Год выпуска автомобиля: " << year << endl;

cout << "Скорость: " << speed << "КМ/Ч" << endl;

cout << "Название автомобиля: " << name << endl;

}

};

int main()

{

automobile a;

a.set\_automobile();

a.get\_automobile();

system("pause");

return 0;

}

1. Программа №4

#include "pch.h"

#include <iostream>

#include<string>

using namespace std;

class Vector {

double x, y, ugol;

public:

void set\_Vector()

{

cout << "- - - - - - - - - - - - - - - - - - - -" << endl;

cout << "Требуется ввести X и Y координаты: " << endl;

cin >> x;

cin >> y;

}

double modul()

{

return sqrt(x\*x + y \* y);

}

void Set\_ugol(Vector &other)

{

ugol = this->x\*other.x + this->y\*other.y / (this->modul()\*other.modul());

}

string summa(Vector &other)

{

return to\_string(this->x + other.x) + "+" + to\_string(this->y + other.y);

}

string raznost(Vector &other)

{

return to\_string(this->x - other.x) + "-" + to\_string(this->y - other.y);

}

double proizvsk(Vector &other)

{

return this->modul()\*other.modul()\*cos(ugol);

}

double proizvvec(Vector &other)

{

return this->modul()\*other.modul()\*sin(ugol);

}

};

int main()

{

Vector a, b;

a.set\_Vector();

b.set\_Vector();

a.Set\_ugol(b);

cout << "- - - - - - - - - - -" << endl;

cout <<"Cумма = "<< a.summa(b) << endl;

cout << "- - - - - - - - - - -" << endl;

cout << "Разность = " << a.raznost(b) << endl;

cout << "- - - - - - - - - - -" << endl;

cout << "Скалярное произведение: " << a.proizvsk(b) << endl;

cout << "- - - - - - - - - - -" << endl;

cout << "Векторное произведение: " << a.proizvvec(b) << endl;

system("pause");

return 0;

}

**Вывод**

Все поставленные задачи были достигнуты. Для получения требуемого результата былииспользованы конструкции класса и перегрузка некоторых операторов. Результаты работы программ изображены на скриншотах (рис. 5-8). Исходные коды программ также доступны по ссылкам на GitHub:

1. <https://github.com/ivankachura/OOP/blob/master/%D0%9B%D0%B0%D0%B1.%D0%A0%D0%B0%D0%B1%D0%BE%D1%82%D0%B0%203/%D0%9B%D0%B0%D0%B1.%D0%A0%D0%B0%D0%B1%D0%BE%D1%82%D0%B0%203.1>
2. <https://github.com/ivankachura/OOP/blob/master/%D0%9B%D0%B0%D0%B1.%D0%A0%D0%B0%D0%B1%D0%BE%D1%82%D0%B0%203/%D0%9B%D0%B0%D0%B1.%D0%A0%D0%B0%D0%B1%D0%BE%D1%82%D0%B0%203.2>
3. <https://github.com/ivankachura/OOP/blob/master/%D0%9B%D0%B0%D0%B1.%D0%A0%D0%B0%D0%B1%D0%BE%D1%82%D0%B0%203/%D0%9B%D0%B0%D0%B1.%D0%A0%D0%B0%D0%B1%D0%BE%D1%82%D0%B0%203.3>
4. <https://github.com/ivankachura/OOP/blob/master/%D0%9B%D0%B0%D0%B1.%D0%A0%D0%B0%D0%B1%D0%BE%D1%82%D0%B0%203/%D0%9B%D0%B0%D0%B1.%D0%A0%D0%B0%D0%B1%D0%BE%D1%82%D0%B0%203.4>

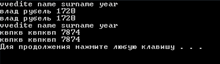
****

Рис. 5 Результат работы программы №1

**C:\Users\user\Desktop\Безымянный8.png**

Рис. 6 Результат работы программы №2

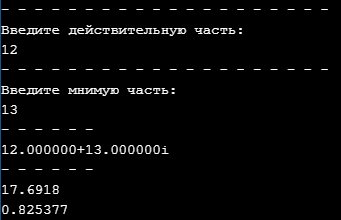
****

Рис. 7 Результат работы программы №3

**C:\Users\user\Desktop\Безымянный8.png**

Рис. 8 Результат работы программы №4

## Лабораторная работа №4

**Конструкторы и деструкторы**

**Цель практической работы**

Целью данной лабораторной работы является знакомство с конструкторами и деструкторами в языке С++.

**Задачи**

Из прошлой лабораторной работы дополнить всем видами конструкторов и деструкторами классы:

1. Complex
2. Vector
3. Tiles
4. Child

**Ход работы**

1. Программа №1

#include "stdafx.h"

#include <iostream>

using namespace std;

int \*p;

void mass( int n);

void zap(int n);

void viv(int n);

void del() { delete[] p; }

int main()

{

int n;

int ss;

do {

cout << "\_ \_ \_ \_ \_ \_ \_ \_ \_ \_ \_ \_ \_ \_ \_" << endl;

cout << "Введите S: " << endl;

cin >> ss;

switch (ss)

{

case 0:

break;

case 1:

cout << "\_ \_ \_ \_ \_ \_ \_ \_ \_ \_ \_ \_ \_ \_ \_" << endl;

cout << "Введите N: " << endl;

cin >> n;

mass(n);

break;

case 2: zap(n);

break;

case 3: viv(n);

break;

case 4: del();

break;

default:

break;

}

} while (ss != 0);

system("pause");

return 0;

}

void mass(int n) {

p = new int[n];

}

void zap(int n) {

for (int i = 0; i < n; i++)p[i] = i + 1;

}

void viv(int n) {

for (int i = 0; i < n; i++) cout << p[i] << endl;

}

1. Программа №2

#include "stdafx.h"

#include <iostream>

#include <string>

using namespace std;

class Tiles {

string brand;

int size\_h, size\_w, price;

public:

Tiles() {

cout << "- - - - - - - - - - - - - - - - - - - " << endl;

cout << "Введите Брэнд ширины и длины, цены:";

cin >> brand;

cin >> size\_h >> size\_w >> price;

}

Tiles(const Tiles &other) {

this->brand = other.brand;

this->size\_h = other.size\_h;

this->size\_w = other.size\_w;

this->price = other.price;

}

~Tiles() {};

string get\_Data() {

return brand + " " + to\_string(size\_h) + " " + to\_string(size\_w) + " " + to\_string(price);

}

};

int main()

{

Tiles a;

cout << "- - - - - - - - " << endl;

cout<<"a=" << a.get\_Data() << endl;

Tiles b = a;

cout << "- - - - - - - - " << endl;

cout << "b=" << b.get\_Data() << endl;

system("pause");

return 0;

}

1. Программа №3

#include "stdafx.h"

#include <iostream>

#include <string>

#include <cmath>

using namespace std;

class Complex {

double Re, Im;

public:

Complex() {

cout << "- - - - - - - - - - - - - - - " << endl;

cout << "Введите действительное часть:";

cin >> Re;

cout << "- - - - - - - - - - - - - - - " << endl;

cout << "Введите мнимое часть:";

cin >> Im;

}

Complex(const Complex &other) {

this->Re = other.Re;

this->Im = other.Im;

}

~Complex() {};

void set\_c() {

cout << "- - - - - - - - - - - - - - - " << endl;

cout << "Введите действительное часть:";

cin >> Re;

cout << "- - - - - - - - - - - - - - - " << endl;

cout << "Введите мнимое часть:";

cin >> Im;

}

string get\_c() {

return to\_string(Re) + "+" + to\_string(Im) + "i";

}

double modul() {

return sqrt(Re\*Re + Im\*Im);

}

double arg() {

return atan(Im / Re);

}

};

int main()

{

Complex a;

cout << "- - - - - - -" << endl;

cout << a.get\_c() << endl;;

cout << "- - - - - - -" << endl;

cout << a.modul() << endl;

cout << "- - - - - - -" << endl;

cout << a.arg() << endl;

system("pause");

return 0;

}

1. Программа №4

#include "stdafx.h"

#include <iostream>

#include<string>

using namespace std;

class Vector {

double x, y, ugol;

public:

Vector() {

cout << "- - - - - - - - - - - - - - - - - - - " << endl;

cout << "Введите X и Y координаты векторов:";

cin >> x >> y;

}

Vector(const Vector &other) {

this->x = other.x;

this->y = other.y;

}

~Vector(){}

void set\_Vector() {

cout << "- - - - - - - - - - - - - - - - - - - " << endl;

cout << "Введите X и Y координаты векторов:";

cin >> x >> y;

}

double modul() {

return sqrt(x\*x + y \* y);

}

void Set\_ugol(Vector &other) {

ugol = this->x\*other.x + this->y\*other.y / (this->modul()\*other.modul());

}

string summa(Vector &other) {

return to\_string(this->x + other.x) + "+" + to\_string(this->y + other.y);

}

string raznost(Vector &other) {

return to\_string(this->x - other.x) + "-" + to\_string(this->y - other.y);

}

double proizvsk(Vector &other) {

return this->modul()\*other.modul()\*cos(ugol);

}

double proizvvec(Vector &other) {

return this->modul()\*other.modul()\*sin(ugol);

}

};

int main()

{

Vector a, b ;

a.Set\_ugol(b);

cout << "! - - - - - - - - - - - - - - - - - - - !" << endl;

cout << "Cумма равна:" << a.summa(b) << endl;

cout << "Разность равна:" << a.raznost(b) << endl;

cout << "Произведение скалярных вел.равно:" << a.proizvsk(b) << endl;

cout << "Произведение векторных вел.равно:" << a.proizvvec(b) << endl;

cout << "! - - - - - - - - - - - - - - - - - - - !" << endl;

system("pause");

return 0;

}

**Вывод**

Были выполнены все задачи. В процессе работы были более подробно изучены разные виды конструкторов и деструкторы. Для выполнения работы были использованы простые конструкторы, конструкторы копирования, а также перегруженный оператор присваивания. Результаты работы программ можно увидеть на скриншотах (рис. 9-12). Исходный код программ также доступен по ссылкам на GitHub:

1. <https://github.com/ivankachura/OOP/blob/master/%D0%9B%D0%B0%D0%B1.%D0%A0%D0%B0%D0%B1%D0%BE%D1%82%D0%B0%204/%D0%9B%D0%B0%D0%B1.%D0%A0%D0%B0%D0%B1%D0%BE%D1%82%D0%B0%204.1>
2. <https://github.com/ivankachura/OOP/blob/master/%D0%9B%D0%B0%D0%B1.%D0%A0%D0%B0%D0%B1%D0%BE%D1%82%D0%B0%204/%D0%9B%D0%B0%D0%B1.%D0%A0%D0%B0%D0%B1%D0%BE%D1%82%D0%B0%204.2>
3. <https://github.com/ivankachura/OOP/blob/master/%D0%9B%D0%B0%D0%B1.%D0%A0%D0%B0%D0%B1%D0%BE%D1%82%D0%B0%204/%D0%9B%D0%B0%D0%B1.%D0%A0%D0%B0%D0%B1%D0%BE%D1%82%D0%B0%204.3>
4. <https://github.com/ivankachura/OOP/blob/master/%D0%9B%D0%B0%D0%B1.%D0%A0%D0%B0%D0%B1%D0%BE%D1%82%D0%B0%204/%D0%9B%D0%B0%D0%B1.%D0%A0%D0%B0%D0%B1%D0%BE%D1%82%D0%B0%204.4>

C:\Users\user\Desktop\Безымянный8.png

Рис. 9 Результат работы программы №1

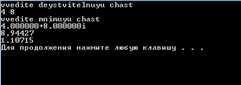


Рис. 10 Результат работы программы №2

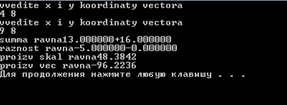


Рис. 11 Результат работы программы №3

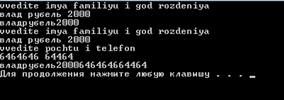


Рис. 12 Результат работы программы №4

## Лабораторная работа №5

**Перегрузка операторов в языке программирования C++**

**Цель практической работы**

Целью данной лабораторной работы является изучениеперегрузки операторов в языке С++ и использование перегруженныхоператоров на практике.

**Задачи**

1. Для класса Complex перегрузить операторы присваивания, инкремента, декремента, сравнения, ввода и вывода.
2. Для класса Vector перегрузить операторы присваивания, сравнения, ввода и вывода.

**Ход работы**

1. Программа №1

#include "stdafx.h"

#include <iostream>

#include <string>

#include <cmath>

using namespace std;

class Complex {

double Re, Im;

public:

Complex() {

cout << "- - - - - - - - - - - - - - - -" << endl;

cout << "Введите действительную часть: ";

cin >> Re;

cout << "- - - - - - - - - - - - - - - -" << endl;

cout << "Введите мнимую часть: ";

cin >> Im;

}

Complex(const Complex &other) {

this->Re = other.Re;

this->Im = other.Im;

}

~Complex() {};

void set\_c() {

cout << "- - - - - - - - - - - - - - - -" << endl;

cout << "Введите действительную часть: ";

cin >> Re;

cout << "- - - - - - - - - - - - - - - -" << endl;

cout << "Введите мнимую часть: ";

cin >> Im;

}

string get\_c() {

return to\_string(Re) + "+" + to\_string(Im) + "i";

}

double modul() {

return sqrt(Re\*Re + Im\*Im);

}

double arg() {

return atan(Im / Re);

}

friend istream& operator>> (std::istream &in, Complex &point)

{

cout << "- - - - - - - - - - - - - - - - " << endl;

cout << "Введите действительную и мнимую часть: ";

in >> point.Re;

in >> point.Im;

return in;

}

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

{

out << point.Re<<"+i\*"<<point.Im;

return out;

}

bool operator <(Complex &other) {

if (this->modul() < other.modul())return true;

else return false;

}

bool operator >(Complex &other) {

if (this->modul() > other.modul())return true;

else return false;

}

bool operator ==(Complex &other) {

if (this->modul() == other.modul())return true;

else return false;

}

Complex& operator ++() {

this->Re++;

return \*this;

}

Complex& operator --() {

this->Re--;

return \*this;

}

};

int main()

{

Complex a;

Complex b = a;

cout << "- - - - - - - - - - - - - - - -" << endl;

cout <<"a-constructora" << a<<endl;

cout << "b=a" << b << endl;

cin >> a;

cout<<"a-izmenennoye" << a << endl ;

cout <<"Модуль А"<< a.modul() << endl;

cout << "Аргумент А" << a.arg() << endl;

cout << "a++ " << ++a << endl;

cout << "a-- " << --a << endl;

if (a > b)cout << a << "А больше Б" << endl;

else if (a < b)cout << b << "А больше Б" << endl;

else if (a == b)cout << "А равно Б";

cout << "- - - - - - - - - - - - - - - -" << endl;

system("pause");

return 0;

}

1. Программа №2

#include "stdafx.h"

#include <iostream>

#include <string>

using namespace std;

class Vector {

double x, y, ugol;

public:

Vector() {

cout << "- - - - - - - - - - - - - - - - - -" << endl;

cout << "Введите X и Y координаты векторов: ";

cin >> x;

cin >> y;

}

Vector(const Vector &other) {

this->x = other.x;

this->y = other.y;

}

~Vector(){}

void set\_Vector() {

cout << "- - - - - - - - - - - - - - - - - -" << endl;

cout << "Введите X и Y координаты векторов: ";

cin >> x;

cin >> y;

}

double modul() {

return sqrt(x\*x + y \* y);

}

void Set\_ugol(Vector &other) {

ugol = this->x\*other.x + this->y\*other.y / (this->modul()\*other.modul());

}

Vector summa(Vector &other) {

this->x = this->x + other.x;

this->y = this->y + other.y;

return \*this;}

Vector raznost(Vector &other) {

this->x = this->x - other.x;

this->y = this->y - other.y;

return \*this;

}

double proizvsk(Vector &other) {

return this->modul()\*other.modul()\*cos(ugol);

}

double proizvvec(Vector &other) {

return this->modul()\*other.modul()\*sin(ugol);

}

friend ostream& operator<< (std::ostream &out, const Vector &vector)

{

out <<"("<<vector.x<<","<< vector.y<<")";

return out;

}

friend istream& operator>> (std::istream &in, Vector &vector)

{

cout << "- - - - - - - - - - - - - - - - - -" << endl;

cout << "Введите X и Y координаты: ";

in >> vector.x;

in >> vector.y;

return in;

}

bool operator > (Vector & other) {

if (this->modul() > other.modul())return true;

else return false;

}

bool operator < (Vector & other) {

if (this->modul() < other.modul())return true;

else return false;

}

bool operator == (Vector & other) {

if (this->modul() == other.modul())return true;

else return false;

}

Vector& operator ++ () {

this->x = x + 1;

this->y = y + 1;

return \*this;

}

Vector& operator -- () {

this->x = x - 1;

this->y = y - 1;

return \*this;

}

};

int main()

{

Vector a, b ;

if (a > b)cout << a <<"Больше"<< endl;

else if (a < b)cout << b << "Больше" << endl;

else if (a == b)cout << "Равно";

cout << "incr a" << ++a<<endl;

cout << "decr a" << --a<<endl;

a.Set\_ugol(b);

cout << "Сумма равна" << a.summa(b) << endl;

cout << "Разность равна" << a.raznost(b) << endl;

cout << "Произведение скал.вел.равно: " << a.proizvsk(b) << endl;

cout << "Произведение век.вел.равно: " << a.proizvvec(b) << endl;

system("pause");

return 0;

}

**Вывод**

Все поставленные задачи были выполнены. В процессе выполнения работы был более плотно изучен механизм перегрузки. Для выполнения поставленных задач была использована перегрузка некоторых бинарных и унарных операторов. Результаты работы программ можно увидеть на скриншотах (рис. 13, 14). Исходный код программ также доступен по ссылкам на GitHub:

1. <https://github.com/ivankachura/OOP/blob/master/%D0%9B%D0%B0%D0%B1.%D0%A0%D0%B0%D0%B1%D0%BE%D1%82%D0%B0%205/%D0%9B%D0%B0%D0%B1.%D0%A0%D0%B0%D0%B1%D0%BE%D1%82%D0%B0%205.1>
2. <https://github.com/ivankachura/OOP/blob/master/%D0%9B%D0%B0%D0%B1.%D0%A0%D0%B0%D0%B1%D0%BE%D1%82%D0%B0%205/%D0%9B%D0%B0%D0%B1.%D0%A0%D0%B0%D0%B1%D0%BE%D1%82%D0%B0%205.2>

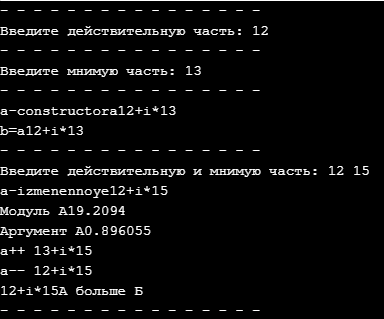


Рис. 13 Результат работы программы №1

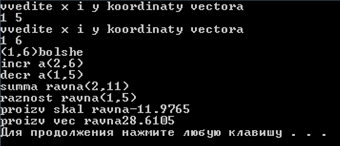


Рис. 14 Результат работы программы №2

## Лабораторная работа №6

**Наследование**

**Цель практической работы**

Целью данной лабораторной работы является изучение наследованная классов в языке С++.

**Задачи**

1. Создать класс «Староста», производный от класса «Студент». Новый класс должен содержать несколько дополнительных методов и полей.
2. Создать класс Alive и расширить его до Bird, Fish, Animal.
3. Создать класс Animal, и расширить его до Dog, Cat.

**Ход работы**

1. Программа №1

#include "stdafx.h"

#include <iostream>

#include <string>

#include <vector>

using namespace std;

class Student {

protected:

string name;

string surname;

int year;

public:

Student(string &name, string &surname, int year) {

this->name = name;

this->surname = surname;

this->year = year;

}

Student() {

cout << "- - - - - - - - - - - -" << endl;

cout << "Введите имя: ";

cin >> name;

cout << "- - - - - - - - - - - -" << endl;

cout << "Введите фамилию: ";

cin >> surname;

cout << "- - - - - - - - - - - -" << endl;

cout << "Введите год рождения: ";

cin >> year;

}

string get\_Data() {

return name + surname + to\_string(year);

}

};

class Headman:public Student {

string email;

int phone;

public:

Headman(string& email, int phone):Student(name,surname,year){

this->email = email;

this->phone = phone;

}

Headman() :Student() {

cout << "- - - - - - - - - - - -" << endl;

cout << "Введите почту: ";

cin >> email;

cout << "- - - - - - - - - - - -" << endl;

out << "Введите номер телефона: ";

cin>> phone;

}

string get\_HData() {

return get\_Data() + email + to\_string(phone);

}

};

int main() {

Student a;

cout << a.get\_Data() << endl;

Headman b;

cout << b.get\_HData() << endl;

system("pause");

return 0;

}

1. Программа №2

#include <iostream>

#include <string>

#include <Windows.h>

using namespace std;

class Alive {

protected:

string name;

int age;

public:

Alive(string &name, int age) : name(name), age(age) {};

string &getName() { return name; }

int getAge() { return age; }

virtual void move() = 0;

};

class Animal : public Alive {

public:

Animal(string &name, int age) : Alive(name, age) {};

cout << "- - - - - - - - - - - - - - -" << endl;

void move() { cout << name << " Он бежит!" << endl; }

};

class Fish : public Alive {

public:

Fish(string &name, int age) : Alive(name, age) {};

cout << "- - - - - - - - - - - - - - -" << endl;

void move() { cout << name << " Он плавает!" << endl; }

};

class Bird : public Alive {

public:

Bird(string &name, int age) : Alive(name, age) {};

cout << "- - - - - - - - - - - - - - -" << endl;

void move() { cout << name << " Он летает!"; }

};

int main() {

SetConsoleCP(1251);

SetConsoleOutputCP(1251);

string name;

int age;

cout << "- - - - - - - - - - - - - - -" << endl;

cout << "Рыба!" << endl;

cout << "Введите имя: ";

cin >> name;

cout << "EВведите возвраст: ";

cin >> age;

Fish f(name, age);

cout << "- - - - - - - - - - - - - - -" << endl;

cout << endl << "Животное!" << endl;

cout << "Eвведите имя: ";

cin >> name;

cout << "Введите возвраст: ";

cin >> age;

Animal a(name, age);

cout << "- - - - - - - - - - - - - - -" << endl;

cout << endl << "Птица" << endl;

cout << "Введите имя: ";

cin >> name;

cout << "Введите возвраст: ";

cin >> age;

Bird b(name, age);

cout << "- - - - - - - - - - - - - - -" << endl;

cout << endl << "Ваша ПТИЦА:" << endl << "ИМЯ: " << b.getName() << endl << "ВОЗВРАСТ: " <<

b.getAge() << endl;

b.move();

cout << endl << "Ваша РЫБА:" << endl << "ИМЯ: " << f.getName() << endl << "ВОЗВРАСТ: " <<

f.getAge() << endl;

f.move();

cout << endl << "Ваше ЖИВОТНОЕ: " << endl << "ИМЯ: " << a.getName() << endl << "ВОЗВРАСТ: " <<

a.getAge() << endl;

cout << "- - - - - - - - - - - - - - -" << endl;

a.move();

system("pause");

return 0;

}

1. Программа №3

#include "stdafx.h"

#include <iostream>

#include <string>

using namespace std;

class Animal {

protected:

string name;

int age;

public:

Animal(string name, int age) : name(name), age(age) {};

string getName() { return name; }

int getAge() { return age; }

virtual void golos() = 0;

};

class Dog : public Animal {

public:

Dog(string name, int age) : Animal(name, age) {};

cout << "- - - - - - - - - - - - - - - - - -"<< endl;

void golos() { cout << "ГАВ ГАВ ГАВ ГАВ ГАВ ГАВ ГАВ ГАВ ГАВ ГАВ ГАВ ГАВ ГАВ" << endl; }

};

class Cat : public Animal {

public:

Cat(string name, int age) : Animal(name, age) {};

cout << "- - - - - - - - - - - - - - - - - -"<< endl;

void golos() { cout << "ММММЯЯЯЯЯУУУ" << endl; }

};

int main() {

string name;

int age;

cout << "- - - - - - - - - - - - - - - - - -"<< endl;

cout << "КОТ" << endl;

cout << "ВВЕДИТЕ ИМЯ: ";

cin >> name;

cout << "ВВЕДИТЕ ВОЗВРАСТ: ";

cin >> age;

Cat c(name, age);

cout << "- - - - - - - - - - - - - - - - - -"<< endl;

cout << endl << "СОБАКА" << endl;

cout << "ВВЕДИТЕ ИМЯ: ";

cin >> name;

cout << "ВВЕДИТЕ ВОЗВРАСТ: ";

cin >> age;

Dog d(name, age);

cout << "- - - - - - - - - - - - - - - - - -"<< endl;

cout << endl << "Твоя собака: " << endl << "imya: " << d.getName() << endl << "god: " <<

d.getAge() << endl;

d.golos();

cout << endl << "Твой кот: " << endl << "imya: " << c.getName() << endl << "god: " <<

c.getAge() << endl;

cout << "- - - - - - - - - - - - - - - - - -"<< endl;

c.golos();

system("pause");

return 0;

**Вывод**

Все поставленные задачи были выполнены. В процессе выполнения работы было изучено и использовано наследование. Результат работы программ изображены на скриншотах (рис. 15-17). Исходный код программ также можно просмотреть на GitHub:

1. <https://github.com/ivankachura/OOP/blob/master/%D0%9B%D0%B0%D0%B1.%D0%A0%D0%B0%D0%B1%D0%BE%D1%82%D0%B0%206/%D0%9B%D0%B0%D0%B1.%D0%A0%D0%B0%D0%B1%D0%BE%D1%82%D0%B0%206.1>
2. <https://github.com/ivankachura/OOP/blob/master/%D0%9B%D0%B0%D0%B1.%D0%A0%D0%B0%D0%B1%D0%BE%D1%82%D0%B0%206/%D0%9B%D0%B0%D0%B1.%D0%A0%D0%B0%D0%B1%D0%BE%D1%82%D0%B0%206.2>
3. <https://github.com/ivankachura/OOP/blob/master/%D0%9B%D0%B0%D0%B1.%D0%A0%D0%B0%D0%B1%D0%BE%D1%82%D0%B0%206/%D0%9B%D0%B0%D0%B1.%D0%A0%D0%B0%D0%B1%D0%BE%D1%82%D0%B0%206.3>

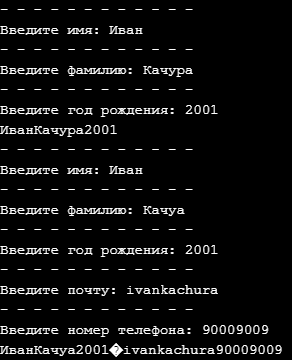


Рис. 15 Результат работы программы №1

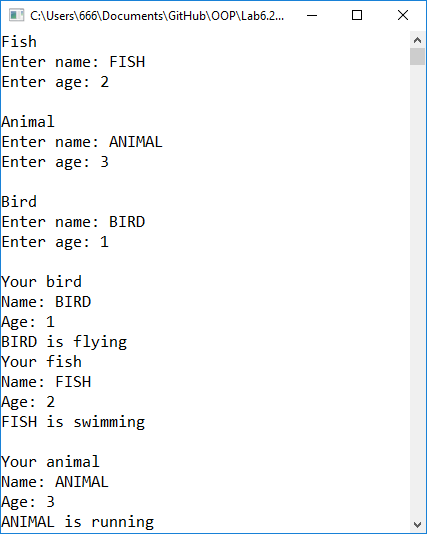


Рис. 16 Результат работы программы №2

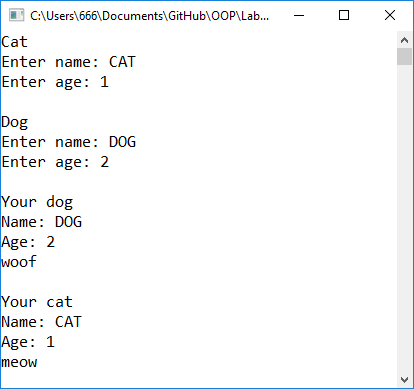


Рис. 17 Результат работы программы №3

## Лабораторная работа №7

**Создание абстрактных классов**

**Цель практической работы**

Целью данной лабораторной работы является изучение и создание абстрактных классов в языке С++.

**Задачи**

Реализовать систему из классов, изображённую на UML диаграмме.

**Ход работы**

#include <iostream>

#include <string>

#include <Windows.h>

using namespace std;

class Human {

protected:

string name;

string surname;

string midname;

int age;

public:

Human() : name(""), surname(""), midname(""), age(0) {};

Human(string surname, string name, string midname, int age) :

name(name), surname(surname), midname(midname), age(age) {};

virtual void print() = 0;

~Human() {}

};

class Student : public Human {

bool on\_lesson;

public:

Student() : Human(), on\_lesson(false) {};

Student(string surname, string name, string midname, int age, bool on\_lesson) :

Human(surname, name, midname, age), on\_lesson(on\_lesson) {};

void print();

~Student() {};

};

class Boss : public Human {

int number\_of\_workers;

public:

Boss() : Human(), number\_of\_workers(0) {};

Boss(string surname, string name, string midname, int age, int number\_of\_workers) :

Human(surname, name, midname, age), number\_of\_workers(number\_of\_workers) {};

void print();

~Boss() {};

};

int main() {

SetConsoleCP(1251);

SetConsoleOutputCP(1251);

string surname, name, midname;

int age, number\_of\_workers;

bool on\_lesson;

cout.setf(ios::boolalpha);

cout << "- - - - - - - - -" << endl;

cout << "Студент- " << endl;

cout << "Введите Фамилию: ";

cin >> surname;

cout << "Введите Имя: ";

cin >> name;

cout << "- - - - - - - - -" << endl;

cout << "Введите отчество: ";

cin >> midname;

cout << "Введите возвраст: ";

cin >> age;

cout << "- - - - - - - - -" << endl;

cout << "Есть ли на уроке (1 - да, 0 - нет): ";

cin >> on\_lesson;

Student s(surname, name, midname, age, on\_lesson);

cout << endl << "Boss" << endl;

cout << "- - - - - - - - -" << endl;

cout << "Введите фамилию: ";

cin >> surname;

cout << "Введите имя: ";

cin >> name;

cout << "Введите отчество: ";

cin >> midname;

cout << "Введите возвраст: ";

cin >> age;

cout << "Введите номер ученика: ";

cout << "- - - - - - - - -" << endl;

cin >> number\_of\_workers;

Boss b(surname, name, midname, age, number\_of\_workers);

s.print();

b.print();

system("pause");

return 0;

}

void Student::print() {

cout << "- - - - - - - - -" << endl;

cout << "ИМЯ: " << surname << " " << name << " " << midname << endl;

cout << "ВОЗВРАСТ: " << age << endl;

cout << "Есть ли на уроке: " << on\_lesson << endl;

cout << "- - - - - - - - -" << endl;

}

void Boss::print() {

cout << "- - - - - - - - -" << endl;

cout << "ИМЯ: " << surname << " " << name << " " << midname << endl;

cout << "ВОЗВРАСТ: " << age << endl;

cout << "Номер ученика: " << number\_of\_workers << endl;

cout << "- - - - - - - - -" << endl;

}

**Вывод**

Все поставленные задачи были выполнены. При выплнении работы были изучены основные элементы UML диаграмм. Для достижения поставленной цели были использованы механизмы наследования, переопределения функций и абстрактные классы. Результат работы программы изображён на скриншоте (рис. 18). Код программы также доступен на GitHub по ссылке:

1. <https://github.com/ivankachura/OOP/blob/master/%D0%9B%D0%B0%D0%B1.%D0%A0%D0%B0%D0%B1%D0%BE%D1%82%D0%B0%207/%D0%9B%D0%B0%D0%B1.%D0%A0%D0%B0%D0%B1%D0%BE%D1%82%D0%B0%207.1>

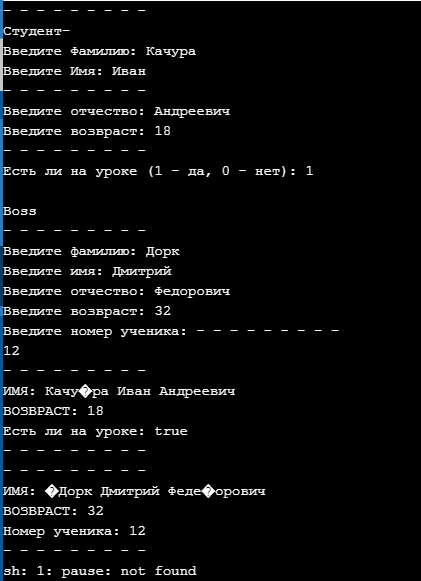


Рис. 18 Результат работы программы

## Лабораторная работа №8

**Бибилиотека STL**

**Цель практической работы**

Целью данной лабораторной работы является знакомство с библиотекой STL – стандартной библиотекой шаблонов - в языке С++, а также показать ее использование на примерах.

**Задачи**

1. Используйте шаблон vector для массива данных о студентах.
2. Используйте шаблон list для двусвязного списка данных класса Complex.
3. Используйте шаблон queue для очереди авто на мойке.

**Ход работы**

1. Программа №1

#include <iostream>

#include <string>

#include <vector>

using namespace std;

int main()

{

string name;

vector<string> students;

int n;

cout << "- - - - - - - - - - - - -"<<endl;

cout << "Введите кол-во студентов:";

cin >> n;

for (int i = 0; i < n; i++)

{

cout << "- - - - - - - - - - - - - - -"<<endl;

cout << "Введите фамилию,имя и группу:" << endl;

cin >> name;

students.push\_back(name);

}

for (int i = 0; i < n; i++)

{

cout << students[i] << endl;

}

system("pause");

return 0;

}

1. Программа №2

#include <iostream>

#include <string>

#include <queue>

using namespace std;

class Client

{

string name;

string car;

public:

Client(string& name, string& car) : name(name), car(car) {};

void getData()

{

cout << "- - - - - - - - - - - - - - - -" << endl;

cout << endl << "Имя: " << name << endl;

cout << "Авто: " << car << endl;

}

};

int main()

{

queue<Client> q;

for (int i = 0; i < 3; i++)

{

string name, car;

cout << "- - - - - - - - - - - - - - - -" << endl;

cout << "Клиент " << i + 1 << endl;

cout << "- -" << endl;

cout << "Введите имя: ";

cout << "- -" << endl;

cin >> name;

cout << "Введите модель автомобиля: ";

cin >> car;

q.push(Client(name, car));

}

cout << "- - - - - - - - - - - - - - - -" << endl;

cout << "Queue" << endl;

for (int i = 1; !q.empty(); i++)

{

cout << i;

q.front().getData();

q.pop();

}

system("pause");

return 0;

}

1. Программа №3

#include <iostream>

#include <cmath>

#include <list>

#include <string>

using namespace std;

class Complex

{

double re;

double im;

public:

Complex(double re, double im) : re(re), im(im)

{

}

Complex(const Complex& c)

{

re = c.re;

im = c.im;

}

Complex operator= (const Complex& c)

{

return Complex(c.re, c.im);

}

void setNumber(double re, double im)

{

this->re = re;

this->im = im;

}

double getModule()

{

return sqrt(re \* re + im \* im);

}

void displayData()

{

cout << re;

if (im < 0)

{

cout << im << "i" << endl;

}

else

{

cout << "+" << im << "i" << endl;

}

cout << "- -" << endl;

cout << "Модуль: " << getModule() << endl;

}

~Complex() {}

};

int main()

{

list<Complex> l;

int n;

cout << "- - - - - - - - - - - - - - - -" << endl;

cout << "Введите кол-во комплексных чисел:";

cin >> n;

for (int i = 0; i < n; i++)

{

double re, im;

cout << "- -" << endl;

cout << "Введите Re(x" << i + 1 << "): ";

cin >> re;

cout << "- -" << endl;

cout << "Введите Im(x" << i + 1 << "): ";

cin >> im;

l.push\_back(Complex(re, im));

}

for (auto a : l)

{

cout << endl;

a.displayData();

}

cout << "- - - - - - - - - - - - - - - -" << endl;

system("pause");

return 0;

}

**Вывод**

Все поставленные задачи были выполнены. Была изучена библиотека STL. Для выполнения работы были использованы коллекции queue, string, list и vector. Результат работы программ изображён на скриншотах (рис. 19-21). Исходный код программ так же доступен на GitHub по ссылкам:

1. <https://github.com/ivankachura/OOP/blob/master/%D0%9B%D0%B0%D0%B1.%D0%A0%D0%B0%D0%B1%D0%BE%D1%82%D0%B0%208/%D0%9B%D0%B0%D0%B1.%D0%A0%D0%B0%D0%B1%D0%BE%D1%82%D0%B0%208.1>
2. <https://github.com/ivankachura/OOP/blob/master/%D0%9B%D0%B0%D0%B1.%D0%A0%D0%B0%D0%B1%D0%BE%D1%82%D0%B0%208/%D0%9B%D0%B0%D0%B1.%D0%A0%D0%B0%D0%B1%D0%BE%D1%82%D0%B0%208.2>
3. <https://github.com/ivankachura/OOP/blob/master/%D0%9B%D0%B0%D0%B1.%D0%A0%D0%B0%D0%B1%D0%BE%D1%82%D0%B0%208/%D0%9B%D0%B0%D0%B1.%D0%A0%D0%B0%D0%B1%D0%BE%D1%82%D0%B0%208.3>

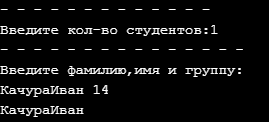


Рис. 19 Результат работы программы №1

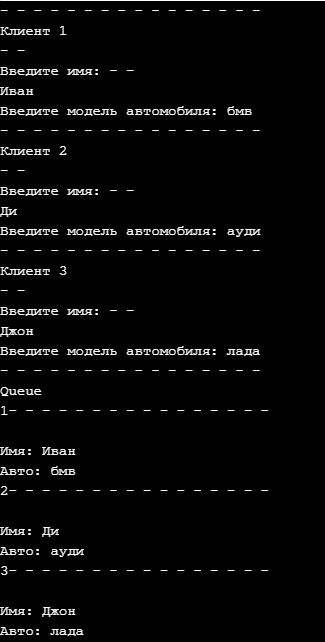


Рис. 20 Результат работы программы №2

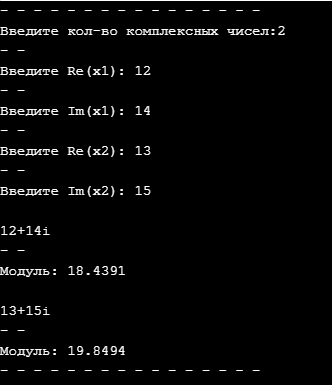


Рис. 21 Результат работы программы №3