|  |  |
| --- | --- |
|  | |
| 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 | «» 2019 г. |  |
|  |  |  |
| «Зачтено» | «\_\_»\_\_\_\_\_\_\_201\_\_ г. |  |

Москва 2019

**Содержание**

[Таблица успеваемости 3](#_Toc2269390)

[Практическая работа №1 4](#_Toc2269391)

[Практическая работа №2 6](#_Toc2269392)

[Практическая работа №3 11](#_Toc2269393)

## Таблица успеваемости

**Лабораторные занятия**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| №/р. | страница в отчете. | Студент выполнил:  (подпись) | Преподаватель принял:  (подпись) | Балл  (максимально возможный) | Балл (Фактический) |
| 1. |  |  |  | 2 |  |
| 2. |  |  |  | 2 |  |
| 3. |  |  |  | 4 |  |
| 4. |  |  |  | 6 |  |
| 5. |  |  |  | 6 |  |
| 6. |  |  |  | 8 |  |
| 7. |  |  |  | 8 |  |
| 8. |  |  |  | 8 |  |
|  |  |  | Итоговый балл: | 44 |  |

**Практические занятия**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| №/р. | страница в отчете. | Студент выполнил:  (подпись) | Преподаватель принял:  (подпись) | Балл  (максимально возможный) | Балл (Фактический) |
| 1. |  |  |  | 2 | … |
| 2. |  |  |  | 2 |  |
| 3. |  |  |  | 4 |  |
| 4. |  |  |  | 6 |  |
| 5. |  |  |  | 6 |  |
| 6. |  |  |  | 8 |  |
| 7. |  |  |  | 8 |  |
|  |  |  | Итоговый балл: | 36 |  |

## Практическая работа №1

**Изучение синтаксиса языка C++**

**Цель практической работы**

Целью данной практической работы является знакомство с синтаксисом, основными алгоритмическими конструкциями языка С++, а также приобретение практических навыков разработки объектно-ориентированных программ на языке программирования С++.

**Задачи**

Реализовать программу, которая считывает имя и здоровается.

Реализовать программу, производящую операции над числами. (Сложение/вычитание, целые числа/числа с плавающей запятой)

**Ход работы**

Реализовать программу, которая считывает имя и здоровается

Реализовать программу, производящую операции над числами. (Сложение/вычитание, целые числа/числа с плавающей запятой)

#include <iostream>

using namespace std;

int main()

{

setlocale(LC\_ALL, "");

double a, b;

cout << "Введите 2 числа: ";

cin >> a >> b;

cout << "a - b = " << (a - b) << endl;

cout << "a + b = " << (a + b) << endl;

system("pause");

return 0;

}

**Вывод**

В ходе выполнения практической работы была использована библиотека стандартного ввода и вывода, операторы ввода, вывода, сложения и вычитания. Были достигнуты результаты, изображённые на рисунках (рис. 1 и 2). Также исходный код программ был помещён на GitHub и доступен по ссылкам:

1. [https://github.com/ivankachura/OOP/tree/master/%D0%9F%D1%80%D0%B0%D0%BA%D1%82%D0%B8%D0%BA%D0%B0%20%E2%84%961](https://github.com/ivankachura/OOP/tree/master/Практика%20№1)

Рис. 1 Результат работы первой программы

Рис. 2 Результат работы второй программы

## Практическая работа №2

**Решение задач на нелинейные алгоритмические конструкции языка С++**

**Цель практической работы**

Целью данной практической работы является получение практических навыков по разработке нелинейных алгоритмов с использованием конструкции «if-else».

**Ход работы**

Вариант 1:

#include "stdafx.h"

#include <iostream>

#include<cmath>

using namespace std;

int main()

{

double a=-0.6, b=5.3, z=0, t=0;

if (a < b) {

z = sqrt(abs(a\*a - b\*b));

}

else z=1-2\*cos(a)\*sin(b);

if (z < b) {

t = cbrt(z + a\*a\*b);

}

else if (z == b) {

t = 1 - log10(z) + cos(a\*a\*b);

}

else t = 1 / cos(z\*a);

cout << a << endl;

cout << b << endl;

cout << z << endl;

cout << t << endl;

system("pause");

return 0;

}

Вариант 2:

#include <iostream>

#include<cmath>

using namespace std;

int main()

{

double a=4.6, b=1.5, y=0, t=0;

if (a <= b) {

y = a - b / a + b\*a + b / a\*a - a\*b + b\*b;

}

else y = a + 2 \* log(b);

if (y == b) {

t = (2 \* y + sqrt(y\*y - a)) / 2 \* b - sqrt(a\*a - y);

}

else if (y < b) {

t = sin(y)\*sin(y) + 1 / tan(a-b);

}

else t = cbrt(y\*sin(a))+ 1/sqrt(y\*cos(b));

cout << a << endl;

cout << b << endl;

cout << y << endl;

cout << t << endl;

system("pause");

return 0;

}

Вариант 3:

#include <iostream>

#include<cmath>

using namespace std;

int main()

{

double a=3.5, x=-0.7, y=0, t=0;

if (a <= x) {

y = a + log(x + a);

}

else y = sqrt(sin(a\*x));

if (a>y) {

t = y/a-x;

}

else if (a==y) {

t = y/(a-x)+a+x/y\*y;

}

else t = tan(a\*x) + cos(2 \* a\*y);

cout << a << endl;

cout << x << endl;

cout << y << endl;

cout << t << endl;

system("pause");

return 0;

}

Вариант 4:

#include <iostream>

#include<cmath>

using namespace std;

int main()

{

double a=4.8, b=-7.9, m=0, n=0;

if (a >= b) {

n = cbrt(a - b);

}

else (a\*a) + (a - b / sin(a\*b));

if (n < b) {

m = (n + a / -b) + sqrt(sin(a)\*sin(a) - cos(n));

}

else if (n == b) {

m = b\*b + tan(n\*a);

}

else m = pow(b, 3) + n\*a\*a;

cout << a << endl;

cout << b << endl;

cout << n << endl;

cout << m << endl;

system("pause");

return 0;

}

**Вывод**

В ходе выполнения практической работы была использована библиотека cmath, которая позволила вычислять сложные математические функции, а также быстро совершить проверку на ошибки, и конструкция ветвления «if-else». Были достигнуты результаты, изображённые на рисунках (рис. 3-6). Также код программ доступен по ссылкам:

1. [https://github.com/ivankachura/OOP/tree/master/%D0%9F%D1%80%D0%B0%D0%BA%D1%82%D0%B8%D0%BA%D0%B0%20%E2%84%962](https://github.com/ivankachura/OOP/tree/master/Практика%20№2)

Рис. 3 Результат практической работы №2 вариант 1

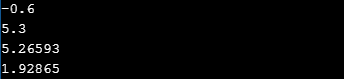


Рис. 4 Результат практической работы №2 вариант 2

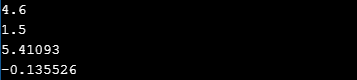


Рис. 5 Результат практической работы №2 вариант 3

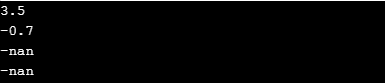
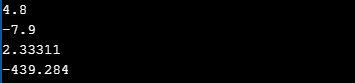


Рис. 6 Результат практической работы №2 вариант 4



## Практическая работа №3

**Классы в языке С++**

**Цель практической работы**

Целью данной практической работы является приобретение практических навыков при разработке классов на языке программирования С++.

**Задачи**

Реализовать класс «Книга»

Реализовать класс «Группа студентов»

Реализовать класс «Автомобиль»

Реализовать класс «Вектор»

Реализовать класс «Библиотека»

**Ход работы**

Класс «Книга»:

#include <iostream>

#include <string>

using namespace std;

class Kniga

{

public:

string autor;

string Name;

double stranica;

void open();

};

int main()

{

Kniga Book;

Book.open();

system("pause");

return 0;

}

void Kniga::open()

{

double random;

cout << "Введите автора: " << endl;

cin >> autor;

cout << "Введите название: " << endl;

cin >> Name;

random = 1 + rand() % 50;

cout << autor << " " << Name << " " << "открыта на" << " "<< random << " "<< "странице." << endl;

}

Класс «Группа студентов»:

#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;

}

Класс «Автомобиль»:

#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;

}

Класс «Вектор»:

#include <iostream>

#include <string>

#include <cmath>

using namespace std;

class vector {

int x, y;

public:

vector(){

x = 0, y = 0;

}

vector(int x, int y) {

this->x = x;

this->y = y;

}

void set\_vector() {

int x1, y1, x2, y2;

cout << "Введите координаты точек: " << endl;

cout << "x1, y1: ";

cin >> x1 >> y1;

cout<<endl;

cout << "x2, y2: ";

cin >> x2 >> y2;

cout << endl;

x = x2 - x1;

y = y2 - y1;

}

double modul() {

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

}

};

int main()

{

vector a;

vector b(3,4);

a.set\_vector();

cout <<"Модуль вектора равен: "<< a.modul() << endl;;

system("pause");

return 0;

}

Класс «Библиотека»:

#include <iostream>

#include<string>

using namespace std;

class Biblioteka

{

string \*book;

bool \*knigi;

int n;

public :

Biblioteka()

{

cout << "Библеотека"<< endl;

cout << "Введите кол-во книг: ";

cin >> n;

book = new string[n+1];

cout << "Введите название книг:";

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

{

getline(cin, book[i]);

}

knigi = new bool[n + 1];

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

knigi[i] = false;

}

~Biblioteka() {

delete[] book;

cout << "Массив был удалён!" << endl;

}

string get\_book();

void take\_book();

void get\_books();

void drop\_book();

};

string Biblioteka ::get\_book()

{

cout << "Введите номер книги: ";

int d;

cin >> d;

return book[d];

}

void Biblioteka ::take\_book()

{

cout << "Введите номер книги, чтобы взять её: ";

int k;

do

{

cin >> k;

if (k > 0 && k < n + 1)knigi[k] = true;

} while (k != 0);

}

void Biblioteka::get\_books()

{

int i;

cout << "Имеющиеся книги:" << endl;

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

if (knigi[i] == true)cout << book[i] << endl;

}

void Biblioteka::drop\_book()

{

cout << "Введите номер книги, которую хотите отдать:";

int k;

do {

cin >> k;

if (k > 0 && k < n + 1)knigi[k] = false;

} while (k != 0);

}

int main()

{

Biblioteka a;

int z;

do {

cout << "1-Показ 2-Взять 3-Отдать 4-Показ";

cin >> z;

switch (z)

{

case 1: cout << a.get\_book() << endl;

break;

case 2: a.get\_book();

break;

case 3: a.drop\_book();

break;

case 4: a.get\_books();

break;

default:

break;

}

} while (z != 0);

system("pause");

return 0;

}

**Вывод**

Все поставленные задачи были выполнены. Для этого были использованы библиотека STL, перегрузка оператора равенства, шаблоны функций. Все задачи были выполнены в виде много файлового проекта с разбиением классов на файлы объявления и реализации класса. Результаты работы программ приведены в рисунках (рис. 7-11). Код программ также доступен на GitHub по ссылкам:

1. [https://github.com/ivankachura/OOP/tree/master/%D0%9F%D1%80%D0%B0%D0%BA%D1%82%D0%B8%D0%BA%D0%B0%20%E2%84%963](https://github.com/ivankachura/OOP/tree/master/Практика%20№3)

Рис. 7 Результат работы программы №1

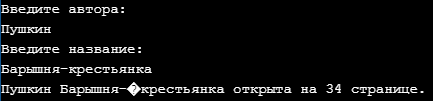


Рис. 8 Результат работы программы №2

Рис. 9 Результат работы программы №3

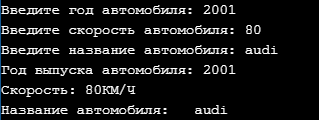


Рис 10 Результат работы программы №4

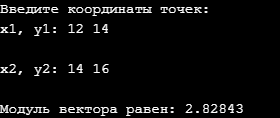
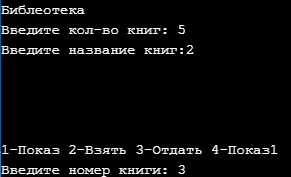


Рис. 11 Результат работы программы №5



## Практическая работа №4

**Динамическое выделение памяти для объектов в языке С++**

**Цель практической работы**

Целью данной практической работы является изучение динамического выделения и удаления памяти, а так же работа с указателями на языке С++.

**Задачи**

Объявите указатель на массив типа double и предложите пользователю выбрать его размер. Далее напишите четыре функции: первая должна выделить память для массива, вторая – заполнить ячейки данными, третья – показать данные на экран, четвёртая – освободить занимаемую память. Программа должна предлагать продолжать работу (создавать новые массивы) или выйти из программы.

Объявите указатель на массив типа int и выделите память для 12-ти элементов. Необходимо написать функцию, которая поменяет значения четных и нечетных ячеек массива.

Объявить и заполнить двумерный динамический массив случайными числами от 10 до 50. Показать его на экран. Для заполнения и показа на экран написать отдельные функции. Количество строк и столбцов выбирает пользователь

**Ход работы**

Программа №1:

Программа №2:

#include <iostream>

using namespace std;

int \*p;

void mass( int n);

void zap(int n);

void viv(int n);

void del() { delete[] p; }

void change(int n);

int main()

{

int n = 12;

mass(n);

zap(n);

viv(n);

change(n);

viv(n);

del();

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] ;

}

void change(int n) {

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

if (i % 2 == 0) swap(p[i], p[i + 1]);

else continue;

}

Программа №3:

#include <iostream>

#include<ctime>

#include <iomanip>

using namespace std;

void zzz(int \*\*ptarray, int m, int n);

void get(int \*\*ptarray, int m, int n );

int main()

{

int m, n;

srand(time(0));

cin >> m >> n;

int \*\*ptarray = new int\*[n];

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

ptarray[count] = new int[m];

zzz(ptarray, m, n);

get(ptarray, m, n);

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

delete[]ptarray[count];

system("pause");

return 0;

}

void get(int \*\*ptarray, int m, int n) {

for (int count\_row = 0; count\_row < n; count\_row++)

{

for (int count\_column = 0; count\_column < m; count\_column++)

cout << setw(4) << setprecision(2) << ptarray[count\_row][count\_column] << " ";

cout << endl;

}

}

void zzz(int \*\*ptarray, int m, int n) {

for (int count\_row = 0; count\_row < n; count\_row++)

for (int count\_column = 0; count\_column < m; count\_column++)

ptarray[count\_row][count\_column] = (rand() % 40 + 10);

}

**Вывод**

Все поставленные задачи были выполнены. При выполнении использовались массивы с динамическим выделением памяти, которые создавались с помощью генератора случайных чисел. Для генератора случайных чисел была подключена библиотека ctime. Результат работы программ приведён на рисунках (рис.12-14). Так же исходный код программ доступен на GitHub по ссылкам:

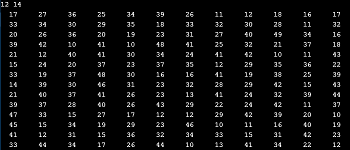
1. [https://github.com/ivankachura/OOP/tree/master/%D0%9F%D1%80%D0%B0%D0%BA%D1%82%D0%B8%D0%BA%D0%B0%20%E2%84%964](https://github.com/ivankachura/OOP/tree/master/Практика%20№4)

Рис. 12 Результат работы программы №1

Рис. 13 Результат работы программы №2

C:\Users\user\Desktop\Безымянный9.png

Рис. 14 Результат работы программы №3



## Практическая работа №5

**Перегрузка функций**

**Цель практической работы**

Целью данной практической работы является приобретение практических навыков по программированию перегрузки функций на языке C++.

**Задачи**

Реализовать сортировку пузырьком для целых чисел, а затем перегрузить её для дробных.

Реализовать сортировку выбором для целых чисел, а затем перегрузить её для дробных.

Реализовать сортировку вставками для целых чисел, а затем перегрузить её для дробных.

Реализовать программу-калькулятор, работающую с разными типами данных.

**Ход работ**

Программа №1:

#include <iostream>

#include <ctime>

#include <iomanip>

using namespace std;

void sort(int \*a , int n );

void sort(double \*a , int n );

int main()

{

srand(time(0));

int n,z;

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

cout<<"Введите кол-во чисел: "<<endl;

cin >> n;

cout<<"Дробные числа (0), Целые числа (1): "<<endl;

cin>>z;

if (z == 0) {

double\*a = new double[n];

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

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

cout<< "Введите дробное число: " << endl;

cin >> a[i];

}

sort(a, n);

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

cout << a[i] << endl;

}

else {

int\*a = new int[n];

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

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

cout <<"Введите целое число: "<<endl;

cin >> a[i];

}

sort(a, n);

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

cout << a[i] << endl;

}

system("pause");

return 0;

}

void sort(double \*a, int n)

{

int i, j, tmp;

cout<<"Идёт сортировка дробных чисел..."<< endl;

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

for (j = n - 1; j >= 0; j--)

if (a[j] < a[j + 1])

{

tmp = a[j];

a[j] = a[j + 1];

a[j + 1] = tmp;

}

cout << "Сортировка закончилась!" << endl;

}

void sort(int \*a, int n)

{

int i, j, tmp;

cout<<"Идёт сортировка целых чисел..."<<endl;

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

for (j = n - 1; j >= 0; j--)

if (a[j] < a[j + 1])

{

tmp = a[j];

a[j] = a[j + 1];

a[j + 1] = tmp;

}

cout << "Сортировка закончилась!" << endl;

}

Программа №2:

#include <iostream>

#include<ctime>

#include <iomanip>

using namespace std;

void sort(int \*a , int n );

void sort(double \*a , int n );

int main() {

srand(time(0));

int n,z;

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

cout<<"Введите кол-во чисел: "<<endl;

cin >> n;

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

cout<<"Дробные - 0, Целые - 1. Введите цифру: "<<endl;

cin>>z;

if (z == 0) {

double\*a = new double[n];

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

{

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

cout<<"Введите дробное число: "<<endl;

cin >> a[i];

}

sort(a, n);

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

cout << a[i] << endl;

}

else {

int\*a = new int[n];

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

{

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

cout<<"Введите целове число: "<<endl;

cin >> a[i];

}

sort(a, n);

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

cout << a[i] << endl;

}

system("pause");

return 0;

}

void sort(double \*a, int n)

{

long i, j, k;

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

cout << "Идёт сортировка [Д] - ОЖИДАЙТЕ!" << endl;

double x;

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

{

k = i; x = a[i];

for (j = i + 1; j < n; j++)

if (a[j] < x) {

k = j; x = a[j];

}

a[k] = a[i]; a[i] = x;

}

cout << "Сортировка закончена..." << endl;

}

void sort(int \*a, int n)

{

long i, j, k;

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

cout << "Идёт сортировка [Ц] - ОЖИДАЙТЕ!" << endl;

double x;

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

{

k = i; x = a[i];

for (j = i + 1; j < n; j++)

if (a[j] < x) {

k = j; x = a[j];

}

a[k] = a[i]; a[i] = x;

}

cout << "Сортировка закончена..." << endl;

}

Программа №3:

#include <iostream>

#include <ctime>

using namespace std;

void FA(int \*\*, int, int);

void PA(int \*\*, int, int);

int main()

{

srand(time(NULL));

int m, n;

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

cout << "Введите количество строк и столбцов: ";

cin >> n >> m;

int \*\*arr = new int\*[n];

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

arr[i] = new int[m];

}

FA(arr, n, m);

PA(arr, n, m);

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

delete [] arr[i];

}

delete [] arr;

system("pause");

return 0;

}

void FA(int \*\*arr, int n, int m)

{

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

for (int j = 0; j < m; j++) {

arr[i][j] = rand() % 41 + 10;

}

}

}

void PA(int \*\*arr, int n, int m)

{

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

cout << "Полученный массив: " << endl;

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

for (int j = 0; j < m; j++) {

cout << arr[i][j] << ' ';

}

cout << endl;

}

}

Программа №4:

#include <iostream>

#include<ctime>

#include <iomanip

using namespace std;

double calc(int a, int b);

double calc(double a, double b);

int main() {

int z;

cout << "Выберите тип чисел(1-Целые,2-Дробные): " << endl;

cin >> z;

if (z == 1) {

int a , b ;

cout << "Введите ДВА числа (желательно целые): " << endl;

cin >> a >> b;

cout << "Ответ: " << endl;

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

}

else {

double a, b;

cout << "Введите ДВА числа (желательно дробные): " << endl;

cin >> a >> b;

cout << "Ответ: " << endl;

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

}

system("pause");

return 0;

}

double calc(int a, int b)

{

int k;

do {

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

cout << "Выберите операцию..." << endl;

cout << "{} = I1+2-3\*4/" << endl;

cin >> k;

switch (k)

{

case 1:

return a + b;

break;

case 2:

return a - b;

break;

case 3:

return a\*b;

break;

case 4:

return a / b;

default:

break;

}

} while (k != 0);

}

double calc(double a, double b)

{

int k;

do {

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

cout << "Выберите операцию..." << endl;

cout << "{} = D1+2-3\*d/" << endl;

cin >> k;

switch (k)

{

case 1:

return a + b;

break;

case 2:

return a - b;

break;

case 3:

return a\*b;

break;

case 4:

return a / b;

default:

break;

}

} while (k != 0);

}

**Вывод**

Все задачи были выполнены. В ходе выполнения была изучена техника перегрузки функций. Так же для заполнения массивов использовался генератор случайных чисел. Результат работы программы приведён на рисунках (рис. 15-18). Исходный код программ так же доступен на GitHub по ссылкам:

1. [https://github.com/ivankachura/OOP/tree/master/%D0%9F%D1%80%D0%B0%D0%BA%D1%82%D0%B8%D0%BA%D0%B0%20%E2%84%965](https://github.com/ivankachura/OOP/tree/master/Практика%20№5)

Рис. 15 Результат работы программы №1

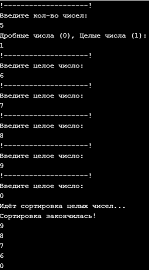


Рис. 16 Результат работы программы №2

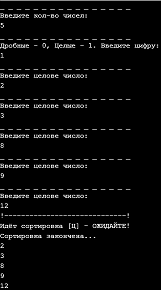


Рис. 17 Результат работы программы №3

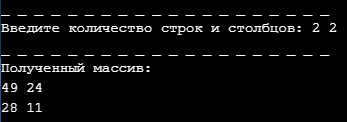
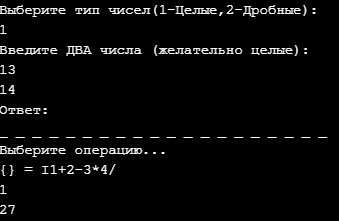


Рис. 18 Результат работы программы №4



## Практическая работа №6

**Решение задач на наследование на языке С++**

**Цель практической работы**

Целью данной практической работы является приобретение практических навыков использования принципа ООП – наследования для разработки программ на языке программирования C++.

**Задачи**

Написать иерархию классов, описывающих имущество налогоплательщиков. Она должна состоять из абстрактного базового класса Property и производных от него классов Appartment, Car и CountryHouse. Базовый класс должен иметь поле worth (стоимость), конструктор с одним параметром, заполняющий это поле, и чисто виртуальный метод рассчета налога, переопределенный в каждом из производных классов. Налог на квартиру вычисляется как 1/1000 ее стоимости, на машину – 1/200, на дачу – 1/500. Также, каждый производный класс должен иметь конструктор с одним параметром, передающий свой параметр конструктору базового класса. В функции main завести массив из 7 указателей на Property и заполнить его указателями на динамические объекты производных классов (первые 3 – Appartment, следующие 2 – Car и последние 2 – CountryHouse). Вывести на экран величину налога для всех 7 объектов. Не забудь также уничтожить динамические объекты перед завершением программы.

Написать набор классов, представляющий выражения. В этом наборе должен быть один абстрактный базовый тип, а также набор производных от него типов по видам выражений (константа, переменная, сумма, разность, произведение, частное, sin, cos, exp, ln). У каждого из классов должны быть следующие виртуальные функции: напечатать выражение (без параметров), вычислить выражение (параметр – значение переменной, результат – значение выражения), вернуть производную выражения (без параметров), создать копию выражения (тоже без параметров).

**Ход работы**

Программа №1:

#include "pch.h"

#include <iostream>

using namespace std;

class Property {

protected:

int worth;

public:

virtual double nalog() = 0;

Property(int worth) {

this->worth = worth;

}

virtual ~Property() {};

};

class Car: public Property

{

public: double nalog() {

return worth / 200;

}

Car(int worth) :Property(worth) {

}

};

class CountryHouse : public Property

{

public: double nalog() {

return worth / 500;

}

CountryHouse(int worth) :Property(worth) {

}

};

class Appartment : public Property

{

public: double nalog() {

return worth / 1000;

}

Appartment(int worth) :Property(worth) {

}

};

int main()

{

Property\* a[7];

for (int i = 0; i < 7; i++) {

int worth;

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

cout << "Введите стоимость объекта: " << i + 1;

cin >> worth;

if (i < 3)a[i] = new Appartment(worth);

else if (i < 5)a[i] = new Car(worth);

else a[i] = new CountryHouse(worth);

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

cout << "Стоимость налога: " << i + 1 << a[i]->nalog() << endl;

}

delete[] \*a;

system("pause");

return 0;

}

Программа №2:

#include "pch.h"

#include <iostream>

#include <cmath>

using namespace std;

class Function {

public:

virtual void printFunction() = 0;

virtual double count(double x) = 0;

virtual void takeDerivative() = 0;

};

class Const : public Function{

double c;

public:

Const(double c) { this->c = c; }

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

void printFunction() { cout << "Функция: f(x) = " << c << endl; }

double count(double x) { return c; }

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

void takeDerivative() { cout << "Директива: 0" << endl; }

};

class Variable : public Function {

public:

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

void printFunction() { cout << "Функция: f(x) = x" << endl; }

double count(double x) { return x; }

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

void takeDerivative() { cout << "Директиква: 1" << endl; }

};

class Sum : public Function {

double y;

public:

Sum(double y) { this->y = y; }

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

void printFunction() { cout << "Функция: f(x) = x + y" << endl; }

double count(double x) { return x + y; }

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

void takeDerivative() { cout << "Директива: 2" << endl; }

};

class Difference : public Function {

double y;

public:

Difference(double y) { this->y = y; }

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

void printFunction() { cout << "Функция: f(x) = x - y" << endl; }

double count(double x) { return x - y; }

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

void takeDerivative() { cout << "Директива: 0" << endl; }

};

class Product : public Function {

double y;

public:

Product(double y) { this->y = y; }

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

void printFunction() { cout << "Функция: f(x) = x \* y" << endl; }

double count(double x) { return x \* y; }

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

void takeDerivative() { cout << "Директива: y \* dx" << endl; }

};

class Quotient : public Function {

double y;

public:

Quotient(double y) { this->y = y; }

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

void printFunction() { cout << "Функция: f(x) = x / y" << endl; }

double count(double x) { return x / y; }

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

void takeDerivative() { cout << "Директива: dx / y" << endl; }

};

class Sin : public Function {

public:

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

void printFunction() { cout << "Функция: f(x) = sin(x)" << endl; }

double count(double x) { return sin(x); }

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

void takeDerivative() { cout << "Дирекетива: cos(x)" << endl; }

};

class Cos : public Function {

public:

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

void printFunction() { cout << "Директива: f(x) = cos(x)" << endl; }

double count(double x) { return cos(x); }

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

void takeDerivative() { cout << "Директива: -sin(x)" << endl; }

};

class Exp : public Function {

public:

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

void printFunction() { cout << "Директива: f(x) = e^x" << endl; }

double count(double x) { return exp(x); }

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

void takeDerivative() { cout << "Директива: e^x" << endl; }

};

class Ln : public Function {

public:

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

void printFunction() { cout << "Функция: f(x) = ln(x)" << endl;

double count(double x) { return log(x); }

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

void takeDerivative() { cout << "Директива: 1/x" << endl; }

};

int main() {

double y, x;

cout << "Constant" << endl << "Введите аргумент: ";

cin >> y;

Const c(y);

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

cout << "Введите аргумент: ";

cin >> x;

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

c.printFunction();

cout << "f(" << x << ") = " << c.count(x) << endl;

c.takeDerivative();

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

cout << endl << "Variable" << endl << "Введите аргумент: ";

cin >> x;

Variable v;

v.printFunction();

cout << "f(" << x << ") = " << v.count(x) << endl;

v.takeDerivative();

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

cout << endl << "Sum" << endl << "Введите значение X и Y: ";

cin >> x >> y;

Sum s(y);

s.printFunction();

cout << "f(" << x << ") = " << s.count(x) << endl;

s.takeDerivative();

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

cout << endl << "Difference" << endl << "Введите значение X и Y: ";

cin >> x >> y;

Difference d(y);

d.printFunction();

cout << "f(" << x << ") = " << d.count(x) << endl;

d.takeDerivative();

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

cout << endl << "Product" << endl << "Введите аргумент: ";

cin >> x >> y;

Product p(y);

p.printFunction();

cout << "f(" << x << ") = " << p.count(x) << endl;

p.takeDerivative();

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

cout << endl << "Quotient" << endl << "Введите значение X и Y: ";

cin >> x >> y;

Quotient q(y);

q.printFunction();

cout << "f(" << x << ") = " << q.count(x) << endl;

q.takeDerivative();

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

cout << endl << "Sin" << endl << "Введите аргумент: ";

cin >> x;

Sin si;

si.printFunction();

cout << "f(" << x << ") = " << si.count(x) << endl;

si.takeDerivative();

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

cout << endl << "Cos" << endl << "Введите аргумент: ";

cin >> x;

Cos co;

co.printFunction();

cout << "f(" << x << ") = " << co.count(x) << endl;

co.takeDerivative();

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

cout << endl << "Опыт: " << endl << "Введите аргумент: ";

cin >> x;

Exp e;

e.printFunction();

cout << "f(" << x << ") = " << e.count(x) << endl;

e.takeDerivative();

cout << endl << "" << endl << "Enter argument: ";

cin >> x;

Ln l;

l.printFunction();

cout << "f(" << x << ") = " << l.count(x) << endl;

l.takeDerivative();

system("pause");

return 0;

}

**Вывод**

Все поставленные задачи были выполнены. В ходе выполнения работы был изучен механизм наследования классов в C++. Также для выполнения работы были использованы абстрактные классы и чисто виртуальные методы базовых абстрактных классов. Результаты работы программ представлены на рисунках (рис 19, 20). Исходный код программ также доступен на GitHub по ссылкам:

1. [https://github.com/ivankachura/OOP/tree/master/%D0%9F%D1%80%D0%B0%D0%BA%D1%82%D0%B8%D0%BA%D0%B0%20%E2%84%966](https://github.com/ivankachura/OOP/tree/master/Практика%20№6)

Рис. 19 Результат работы программы №1

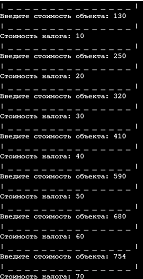


Рис. 20 Результат работы программы №2

## C:\Users\user\Desktop\Безымянный16.png

## C:\Users\user\Desktop\Безымянный17.png

## Практическая работа №7

**Абстрактные классы. Множественное наследование**

**Цель практической работы**

Целью данной практической работы является знакомство с абстрактными классами и механизмом виртуальных функций, а так же множественным наследованием на языке программирования C++.

**Задачи**

Реализовать абстрактный класс «Животное» и путём наследования от него получить классы «Кошка», «Собака», «Попугай».

Реализовать абстрактный класс «Фигура» и путём наследования от него получить абстрактный класс «Четырёхугольник», и затем путём наследования получить классы «Ромб», «Прямоугольник».

Реализовать абстрактный класс «Транспортное средство» и путём наследования от него получить классы «Автомобиль», «Автобус», «Велосипед».

Реализовать абстрактные классы «Экран» и «Клавиатура», путём наследования от них получить классы «Ноутбук», «Телефон», «Стационарный компьютер».

**Ход работы**

Программа №1:

#include <iostream>

#include<string>

using namespace std;

class Animal {

protected:

string name;

int year;

public:

Animal(string name, int year) {

this->name = name;

this->year = year;

}

Animal() {

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

cout<<"Введите кол-во лет: "<<endl;

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

cin >> name;

cin >> year;

}

virtual void golos() = 0;

void get\_Data() {

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

cout << "Ya" << name << " " << year << endl;

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

}

};

class Cat: public Animal {

public:

Cat(string name, int year):Animal(name,year){}

Cat():Animal(){}

void golos() {

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

cout << "Мяу."<<endl;

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

}

};

class Dog : public Animal {

public:

Dog(string name, int year) :Animal(name, year) {}

Dog() :Animal() {}

void golos() {

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

cout << "Гав, Гав." << endl;

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

}

};

class Parrot : public Animal {

public:

Parrot(string name, int year) :Animal(name, year) {}

Parrot() :Animal() {}

void golos() {

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

cout << "Карррр." << endl;

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

}

};

int main(){

Cat a;

a.get\_Data();

a.golos();

Dog b;

b.get\_Data();

b.golos();

Parrot c;

c.get\_Data();

c.golos();

system("pause");

return 0;

}

Программа №2:

#include <iostream>

#include <cmath>

using namespace std;

class Forug{

protected:

int a1, a2, a3, a4;

public:

Forug() {

cout << "Введите длину сторон: " << endl;

cin >> a1;

cin >> a2;

cin >> a3;

cin >> a4;

}

virtual void sqaire() = 0;

};

class Romb:public Forug {

public:

double sinus;

Romb() :Forug() {

cout << "Введите меньший угол: " << endl;

cin >> sinus;

}

void sqaire() {

cout << a1\*a1\*sin(sinus)<<endl;;

}

};

class Pryamougolnic :public Forug {

public:

Pryamougolnic() :Forug() {

}

void sqaire() {

cout<< a1\*a3<<endl;

}

};

class Figure {

public:

void sqaire(Forug \*forug) {

cout << "Площадь равна: ";

forug->sqaire();

}

};

int main()

{

Pryamougolnic c;

Romb b;

Figure a;

a.sqaire(&b);

a.sqaire(&c);

system("pause");

return 0;

}

Программа №3:

#include <iostream>

#include <cmath>

#include <string>

using namespace std;

class Vehicle {

protected:int massa;

int skorost;

string name;

public:Vehicle() {

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

cout << "Введите массу и скорость машины:" << endl;

cin >> massa >> skorost >> name;

}

virtual void kto\_ya() = 0;

};

class Bycicle:public Vehicle {

public:

Bycicle() :Vehicle() { }

void kto\_ya() {

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

cout << "Велосипед:" << name << ' ' << skorost << "km/h " << massa << "kg"<<endl;

}

};

class Bus :public Vehicle {

public:

Bus() :Vehicle() {}

void kto\_ya() {

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

cout << "Автобус:" << name << ' ' << skorost << "km/h " << massa << "kg" << endl;

}

};

class Avto :public Vehicle {

public:

Avto() :Vehicle() { }

void kto\_ya() {

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

cout << "Машина:" << name << ' ' << skorost << "km/h " << massa << "kg" << endl;

}

};

int main()

{

string z;

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

cout << "Введите транспортное средство (Автобус, Автомобиль, Велосипед): " << endl;

cin >> z;

if(z=="Автобус"){ Bus a; a.kto\_ya();

}

else if (z == "Автомобиль") {

Avto a; a.kto\_ya();

}

else if (z == "Велосипед") {

Bycicle a; a.kto\_ya();

}

system("pause");

return 0;

}

Программа №4:

#include <iostream>

using namespace std;

class Screen {

protected:

double razr;

public:

Screen(double size) { this->razr = size; }

virtual void izmScreen(double size) = 0;

};

class Keyboard {

protected:

int buttons;

public:

Keyboard(int buttons) { this->buttons = buttons; }

virtual void izmKeyboard(int buttons) = 0;

};

class Computer : public Screen, Keyboard {

public:

Computer(double razr, int buttons) : Screen(razr), Keyboard(buttons) {};

void izmScreen(double size) {

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

cout << "ЭКРАН" << razr <<

"БЫЛ ИЗМЕНЁН" << size << "РАЗМЕР" << endl;

razr = size;

}

void izmKeyboard(int buttons) {

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

cout << "КЛАВИАТУРА С" << this->buttons << " КНОПКАМИ" <<

"БЫЛА ИЗМЕНЕНА" << buttons << " КНОПОК" << endl;

this->buttons = buttons;

}

~Computer() {}

};

class Notebook : public Screen, Keyboard {

public:

Notebook(double razr, int buttons) : Screen(razr), Keyboard(buttons) {};

void izmScreen(double size) {

if (razr == size) {

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

cout << "Экран был изменён." << endl;

}

else {

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

cout << "Экран не изменён." << endl;

}

}

void izmKeyboard(int buttons) {

if (this->buttons == buttons) {

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

cout << "Клавиатура была изменена." << endl;

}

else {

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

cout << "Клавиатура не была изменена." << endl;

}

}

~Notebook() {}

};

class Phone : public Screen, Keyboard {

public:

Phone(double razr, int buttons) : Screen(razr), Keyboard(buttons) {};

void izmScreen(double size) {

if (razr == size) {

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

cout << "ЭКРАН ТЕЛЕФОНА ИЗМЕНЁН" << endl;

}

else {

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

cout << "ЭКРАН ТЕЛЕФОНА НЕ БЫЛ ИЗМЕНЕН" << endl;

}

}

void izmKeyboard(int buttons) {

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

cout << "Клавиатура не может быть изменена!" << endl;

}

~Phone() {}

};

int main() {

int z=1;

do {

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

cout << "Какое устройсто будет рабоать: 1-telefon, 2- notebook, 3- computer" << endl;

int e, r;

cin >> z;

switch (z)

{

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

case 1: {cout << "Введите размер экрана и кол-во кнопок: " << endl;

cin >> e >> r;

Phone a(e, r);

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

cout << "На сколько вы хотите изменить размер экрана и кол-во кнопок: " << endl;

cin >> e >> r;

a.izmKeyboard(e);

a.izmScreen(r);

break; }

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

case 2: {cout << "Введите размер экрана и кол-во кнопок: " << endl;

cin >> e >> r;

Notebook b(e, r);

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

cout << "На сколько вы хотите изменить размер экрана и кол-во кнопок: " << endl;

cin >> e >> r;

b.izmKeyboard(e);

b.izmScreen(r);

break; }

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

case 3: {cout << "Введите размер экрана и кол-во кнопок: " << endl;

cin >> r >> e;

Computer c(e, r);

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

cout << "На сколько вы хотите изменить размер экрана и кол-во кнопок: " << endl;

cin >> e >> r;

c.izmKeyboard(e);

c.izmScreen(r);

break; }

default:

break;

}

} while (z != 0);

system("pause");

return 0;

}

**Вывод**

Все задачи были выполнены. В ходе выполнения работы были более плотно изучены множественное наследование и абстрактные классы. Для достижения поставленной цели были использованы различные STL контейнеры и библиотека cmath. Результаты работы программ представлены на изображениях (рис. 21-24). Исходный кодвыполненных програм также доступе но ссылка на GitHub:

1. [https://github.com/ivankachura/OOP/tree/master/%D0%9F%D1%80%D0%B0%D0%BA%D1%82%D0%B8%D0%BA%D0%B0%20%E2%84%967](https://github.com/ivankachura/OOP/tree/master/Практика%20№7)

Рис. 21 Результат работы программы №1

Рис. 22 Результат работы программы №2

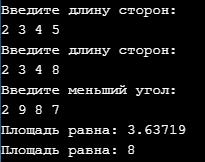


Рис. 23 Результат работы программы №3

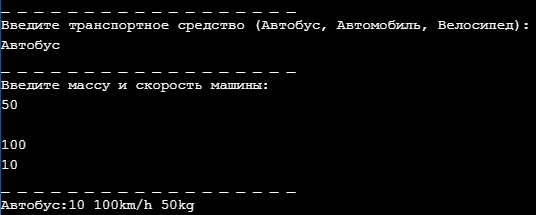
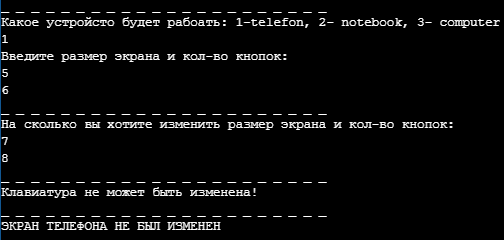


Рис. 24 Результат работы программы №4



## Практическая работа №8

**Работа с файлами в языке С++**

**Цель практической работы**

Целью данной практической работы является приобретение практических навыков по работе с файлами на языке программирования языке C++.

**Задачи**

Реализуйте программу, считывающую текст из файла и выводящую каждое слово с новой строки.

Реализуйте программу, считывающую текст с клавиатуры и записывающую его в файл.

**Ход работы**

Программа №1:

#include "stdafx.h"

#include <fstream>

#include <iostream>

#include <string>

using namespace std;

int main(int argc, char\* argv[])

{

string a;

cin >> a;

ofstream fout("TEST.txt");

fout << a;

fout.close();

system("pause");

return 0;

}

Программа №2:

#include "stdafx.h"

#include <fstream>

#include <iostream>

#include <string>

using namespace std;

int main()

{

setlocale(LC\_ALL, "rus");

string s;

ifstream file("TEST.txt");

if (file.is\_open()){

cout << "Файл открылся!" << endl;

for (file >> s; !file.eof(); file >> s)

cout << s << endl;

}

else

{

cout << "Файл не открыт ;(" << endl;

return -1;

}

system("pause");

return 0;

}

**Вывод**

Все задачи были выполнены успешно. В ходе работы была исследована работа с файлами. Для достижения результата использовалась библиотек fstream, позволяющая производить работу с файлами. Результаты работ программ приведены на рисунках (рис. 25, 26). Также исходный код програм доступенн по ссылкам на GitHub:

1. [https://github.com/ivankachura/OOP/tree/master/%D0%9F%D1%80%D0%B0%D0%BA%D1%82%D0%B8%D0%BA%D0%B0%20%E2%84%968](https://github.com/ivankachura/OOP/tree/master/Практика%20№8)

Рис. 25 Результат работы программы №1

Рис. 26 Результат работы программы №2

## 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**

Написать программу – калькулятор комплексных чисел.

На основе задания 1 написать программу, считывающую комплексные числа из файла complex.txt и находящуюю в них число с наибольшим модулем.

**Ход работы**

Файл заголовков Complex.h:

Программа №1

Программа №2

**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 рисунках (рис. 27, 28). Исходный код также доступен по ссылкам на GitHub:

1. <https://github.com/IvLaptev/OOP/tree/master/Lab1.1>

Рис. 27 Результат работы программы №1

Рис. 28 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**

Напишите программу, которая вызывает различные виды функции в зависимости от заданного условия.

Напишите программу с использованием некой универсальной функции, которая в качестве возвращаемого значения возвращает указатель на функцию, зависящий от некоторого условия.v

**Ход работы**

Программа №1

Программа №2

**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 рисунках (рис. 29, 30). Исходный код программ также доступен по ссылкам на GitHub:

1. <https://github.com/IvLaptev/OOP/tree/master/Lab2.1>

Рис. 29 Результат работы программы №1

Рис. 30 Результат работы программы №2

## Лабораторная работа №3

**Классы**

**Цель практической работы**

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

**Задачи**

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.

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().

Создать класс Complex, в котором реализовано комплексное число. В данном классе должны присутствовать методы, позволяющие рассчтать и вывеси модуль и аргументы данного числа.

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**

Программа №1

**#include** <iostream>

**#include** <string>

**#include** <Windows.h>

**using** **namespace** std;

**class** Child {

string name;

string surname;

**int** age;

**public**:

**Child**(string&, string&, **int**);

**void** **setName**(string& name) { **this**->name = name; } // Методы ввода данных

**void** **setSurname**(string& surname) { **this**->surname = surname; }

**void** **setAge**(**int** age) { **this**->age = age; }

**void** **display**(); // Вывод данных на экран

};

**int** **main**() {

**SetConsoleCP**(1251);

**SetConsoleOutputCP**(1251);

string name, surname;

**int** age;

cout << "Enter name of the first child: ";

cin >> name;

cout << "Enter surname of the first child: ";

cin >> surname;

cout << "Enter age of the first child: ";

cin >> age;

Child c1(name, surname, age);

cout << "Enter name of the second child: ";

cin >> name;

cout << "Enter surname of the second child: ";

cin >> surname;

cout << "Enter age of the second child: ";

cin >> age;

Child c2(name, surname, age);

c1.display();

c2.display();

**system**("pause");

**return** 0;

}

**Child::Child**(string& name, string& surname, **int** age) {

setName(name);

setSurname(surname);

setAge(age);

}

**void** **Child::display**() {

cout << "Name: " << name << **endl**;

cout << "Surname: " << surname << **endl**;

cout << "Age: " << age << **endl**;

}

1. Программы №2

**#include** <iostream>

**#include** <Windows.h>

**#include** <string>

**using** **namespace** std;

**class** Tiles {

**public**:

string brand;

**double** size\_w;

**double** size\_h;

**double** price;

**void** **getData**(); // Вывод данных на экран

};

**int** **main**() {

**SetConsoleCP**(1251);

**SetConsoleOutputCP**(1251);

Tiles t1;

cout << "Enter brand of the first tiles: ";

cin >> t1.brand;

cout << "Enter width of the first tiles: ";

cin >> t1.size\_w;

cout << "Enter height of the first tiles: ";

cin >> t1.size\_h;

cout << "Enter price of the first tiles: ";

cin >> t1.price;

Tiles t2;

cout << "Enter brand of the first tiles: ";

cin >> t2.brand;

cout << "Enter width of the first tiles: ";

cin >> t2.size\_w;

cout << "Enter height of the first tiles: ";

cin >> t2.size\_h;

cout << "Enter price of the first tiles: ";

cin >> t2.price;

cout << "The first tiles" << **endl**;

t1.getData();

cout <<**endl** << "The second tiles" << **endl**;

t2.getData();

**system**("pause");

**return** 0;

}

**void** **Tiles::getData**() {

cout << "Brand: " << brand << **endl**;

cout << "Width: " << size\_w << **endl**;

cout << "Height: " << size\_h << **endl**;

cout << "Price: " << price << **endl**;

}

1. Программа №3

**#include** <iostream>

**#include** <cmath>

**using** **namespace** std;

**class** Complex {

**double** re;

**double** im;

**public**:

**Complex**(**double** re, **double** im) { setNumber(re, im); }

**void** **setNumber**(**double**, **double**); // Получает мнимую и действительную части числа

**double** **getModule**() { **return** **sqrt**(re \* re + im \* im); } // Возвращает модуль

**void** **displayData**(); // Выводит информацию о числе на экран

};

**int** **main**() {

**double** re, im;

cout << "Enter Re(a): ";

cin >> re;

cout << "Enter Im(a): ";

cin >> im;

Complex c1(re, im);

cout << "Enter Re(b): ";

cin >> re;

cout << "Enter Im(b): ";

cin >> im;

Complex c2(re, im);

c1.displayData();

c2.displayData();

**system**("pause");

**return** 0;

}

**void** **Complex::setNumber**(**double** re, **double** im) {

**this**->re = re;

**this**->im = im;

}

**void** **Complex::displayData**() {

cout << "Number: " << re;

**if** (im < 0) {

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

} **else** {

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

}

cout << "Module: " << getModule() << **endl**;

}

1. Программа №4

**#include** <iostream>

**#include** <cmath>

**using** **namespace** std;

**class** Vector {

**double** x;

**double** y;

**double** z;

**public**:

**Vector**(**double** x, **double** y, **double** z) { setVector(x, y, z); }

**void** **setVector**(**double** x, **double** y, **double** z); // Получает координаты вектора

**double** **getModule**() { **return** **sqrt**(x \* x + y \* y + z \* z); } // Возвращает модуль

**void** **printData**(); // Выводит данные на экран

Vector **operator=** (**const** Vector& v) { **return** Vector(v.x, v.y, v.z); }

**friend** Vector **operator+** (**const** Vector& v1, **const** Vector& v2) {

**return** Vector(v1.x + v2.x, v1.y + v2.y, v1.z + v2.z);

}

**friend** Vector **operator-** (**const** Vector& v1, **const** Vector& v2) {

**return** Vector(v1.x - v2.x, v1.y - v2.y, v1.z - v2.z);

}

};

**int** **main**() {

**double** x, y, z;

cout << "Set coordinates of the vector:" << **endl** << "x = ";

cin >> x;

cout << "y = ";

cin >> y;

cout << "z = ";

cin >> z;

Vector v1(x, y, z);

Vector v2 = v1 + v1 - v1;

v2.printData();

**system**("pause");

**return** 0;

}

**void** **Vector::setVector**(**double** x, **double** y, **double** z) {

**this**->x = x;

**this**->y = y;

**this**->z = z;

}

**void** **Vector::printData**() {

cout << "Coordinates: (" << x << ", " << y << ", " << z << ")" << **endl**;

cout << "Module: " << getModule() << **endl**;

}

**Вывод**

Все поставленные задачи были достигнуты. Для получения требуемого результата былииспользованы конструкции класса и перегрузка некоторых операторов. Результаты работы программ изображены на рисунках (рис. 31-34). Исходные коды программ также доступны по ссылкам на GitHub:

1. <https://github.com/IvLaptev/OOP/tree/master/Lab3.1>
2. <https://github.com/IvLaptev/OOP/tree/master/Lab3.2>
3. <https://github.com/IvLaptev/OOP/tree/master/Lab3.3>
4. <https://github.com/IvLaptev/OOP/tree/master/Lab3.4>

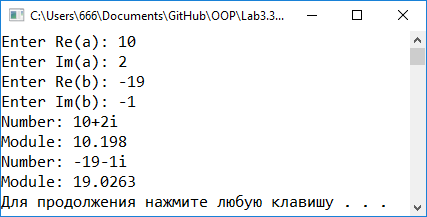


Рис. 31 Результат работы программы №1

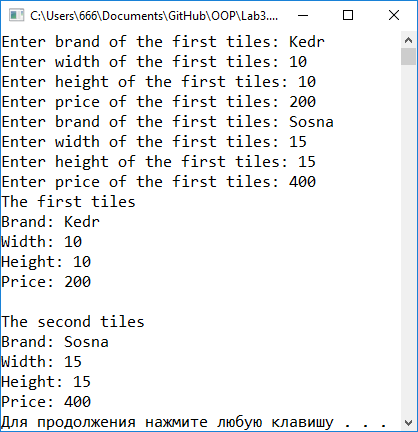


Рис. 32 Результат работы программы №2

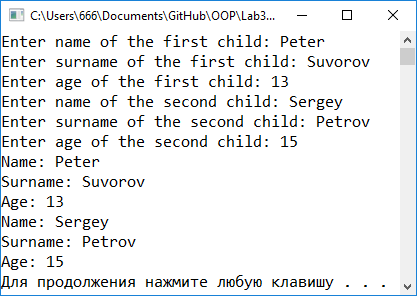


Рис. 33 Результат работы программы №3

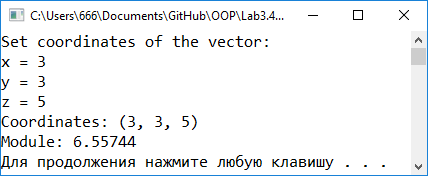


Рис. 34 Результат работы программы №4

## 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** <iostream>

**#include** <string>

**#include** <Windows.h>

**using** **namespace** std;

**class** Child {

string name;

string surname;

**int** age;

**public**:

**Child**(string&, string&, **int**); // Конструкторы класса

**Child**(**const** Child&);

Child **operator=** (**const** Child&);

**void** **setName**(string& name) { **this**->name = name; } // Методы ввода данных

**void** **setSurname**(string& surname) { **this**->surname = surname; }

**void** **setAge**(**int** age) { **this**->age = age; }

**void** **display**(); // Выводит данные на экран

**~Child**() {}

};

**int** **main**() {

**SetConsoleCP**(1251);

**SetConsoleOutputCP**(1251);

string name, surname;

**int** age;

cout << "Enter name of the first child: ";

cin >> name;

cout << "Enter surname of the first child: ";

cin >> surname;

cout << "Enter age of the first child: ";

cin >> age;

Child c1(name, surname, age);

Child c2(c1);

Child c3 = c2;

c1.display();

c2.display();

c3.display();

**system**("pause");

**return** 0;

}

Child **Child::operator=** (**const** Child& c) {

**this**->name = c.name;

**this**->age = c.age;

**this**->surname = c.surname;

**return** \***this**;

}

**Child::Child**(**const** Child& c) {

name = c.name;

surname = c.surname;

age = c.age;

}

**Child::Child**(string& name, string& surname, **int** age) {

setName(name);

setSurname(surname);

setAge(age);

}

**void** **Child::display**() {

cout << "Name: " << name << **endl**;

cout << "Surname: " << surname << **endl**;

cout << "Age: " << age << **endl**;

}

1. Программа №2

**#include** <iostream>

**#include** <Windows.h>

**#include** <string>

**using** **namespace** std;

**class** Tiles {

**public**:

string brand;

**double** size\_w;

**double** size\_h;

**double** price;

**Tiles**(string& brand, **double** size\_w, **double** size\_h, **double** price) :

brand(brand), size\_w(size\_w), size\_h(size\_h), price(price) {}

**Tiles**(**const** Tiles&);

Tiles **operator=** (**const** Tiles&);

**void** **getData**();

**~Tiles**() {}

};

**int** **main**() {

**SetConsoleCP**(1251);

**SetConsoleOutputCP**(1251);

string brand;

**double** size\_w, size\_h, price;

cout << "Enter brand of the first tiles: ";

cin >> brand;

cout << "Enter width of the first tiles: ";

cin >> size\_w;

cout << "Enter height of the first tiles: ";

cin >> size\_h;

cout << "Enter price of the first tiles: ";

cin >> price;

Tiles t1(brand, size\_w, size\_h, price);

Tiles t2(t1);

Tiles t3 = t1;

cout << "The first tiles" << **endl**;

t1.getData();

cout <<**endl** << "The second tiles" << **endl**;

t2.getData();

cout <<**endl** << "The third tiles" << **endl**;

t3.getData();

**system**("pause");

**return** 0;

}

**Tiles::Tiles**(**const** Tiles& t) {

**this**->brand = t.brand;

**this**->price = t.price;

**this**->size\_h = t.size\_h;

**this**->size\_w = t.size\_w;

}

Tiles **Tiles::operator=** (**const** Tiles& t) {

**this**->brand = t.brand;

**this**->price = t.price;

**this**->size\_h = t.size\_h;

**this**->size\_w = t.size\_w;

**return** \***this**;

}

**void** **Tiles::getData**() {

cout << "Brand: " << brand << **endl**;

cout << "Width: " << size\_w << **endl**;

cout << "Height: " << size\_h << **endl**;

cout << "Price: " << price << **endl**;

}

1. Программа №3

**#include** <iostream>

**#include** <cmath>

**using** **namespace** std;

**class** Complex {

**double** re;

**double** im;

**public**:

**Complex**(**double** re, **double** im) : re(re), im(im) {}

**Complex**(**const** Complex& c);

Complex **operator=** (**const** Complex& c);

**void** **setNumber**(**double** re, **double** im);

**double** **getModule**();

**void** **displayData**();

**~Complex**() {}

};

**int** **main**() {

**double** re, im;

cout << "Enter Re(a): ";

cin >> re;

cout << "Enter Im(a): ";

cin >> im;

Complex c1(re, im);

Complex c2(c1);

Complex c3 = c1;

c1.displayData();

c2.displayData();

c3.displayData();

**system**("pause");

**return** 0;

}

**Complex::Complex**(**const** Complex& c) {

re = c.re;

im = c.re;

}

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

**void** **Complex::setNumber**(**double** re, **double** im) {

**this**->re = re;

**this**->im = im;

}

**double** **Complex::getModule**() { **return** **sqrt**(re \* re + im \* im); }

**void** **Complex::displayData**() {

cout << "Number: " << re;

**if** (im < 0) {

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

} **else** {

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

}

cout << "Module: " << getModule() << **endl**;

}

1. Программа №4

**#include** <iostream>

**#include** <cmath>

**using** **namespace** std;

**class** Vector {

**double** x;

**double** y;

**double** z;

**public**:

**Vector**(**double** x, **double** y, **double** z);

**Vector**(**const** Vector& v);

**void** **setVector**(**double** x, **double** y, **double** z);

**double** **getModule**();

**void** **printData**();

Vector **operator=** (**const** Vector& v);

**friend** Vector **operator+** (**const** Vector& v1, **const** Vector& v2);

**friend** Vector **operator-** (**const** Vector& v1, **const** Vector& v2);

};

**int** **main**() {

**double** x, y, z;

cout << "Set coordinates of the vector:" << **endl** << "x = ";

cin >> x;

cout << "y = ";

cin >> y;

cout << "z = ";

cin >> z;

Vector v1(x, y, z);

Vector v2 = v1;

Vector v3(v2);

v1.printData();

v2.printData();

v3.printData();

**system**("pause");

**return** 0;

}

**Vector::Vector**(**double** x, **double** y, **double** z) { setVector(x, y, z); }

**Vector::Vector**(**const** Vector& v) {

x = v.x;

y = v.y;

z = v.z;

}

**void** **Vector::setVector**(**double** x, **double** y, **double** z) {

**this**->x = x;

**this**->y = y;

**this**->z = z;

}

**double** **Vector::getModule**() { **return** **sqrt**(x \* x + y \* y + z \* z); }

**void** **Vector::printData**() {

cout << "Coordinates: (" << x << ", " << y << ", " << z << ")" << **endl**;

cout << "Module: " << getModule() << **endl**;

}

Vector **Vector::operator=** (**const** Vector& v) { **return** Vector(v.x, v.y, v.z); }

Vector **operator+** (**const** Vector& v1, **const** Vector& v2) {

**return** Vector(v1.x + v2.x, v1.y + v2.y, v1.z + v2.z);

}

Vector **operator-** (**const** Vector& v1, **const** Vector& v2) {

**return** Vector(v1.x - v2.x, v1.y - v2.y, v1.z - v2.z);

}

**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 рисунках (рис. 35-38). Исходный код программ также доступен по ссылкам на GitHub:

1. <https://github.com/IvLaptev/OOP/tree/master/Lab4.1>
2. <https://github.com/IvLaptev/OOP/tree/master/Lab4.2>
3. <https://github.com/IvLaptev/OOP/tree/master/Lab4.3>
4. <https://github.com/IvLaptev/OOP/tree/master/Lab4.4>

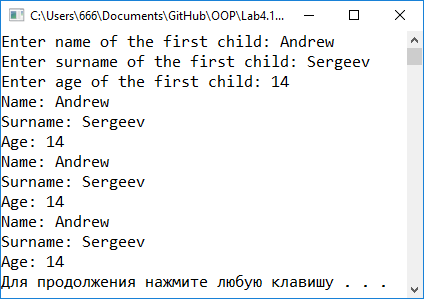


Рис. 35 Результат работы программы №1

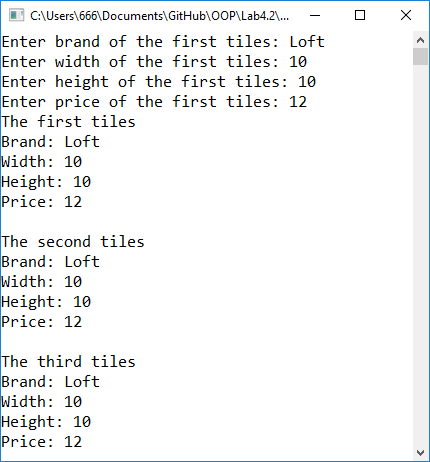


Рис. 36 Результат работы программы №2

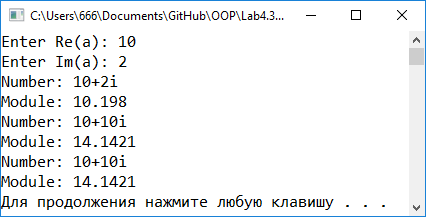


Рис. 37 Результат работы программы №3

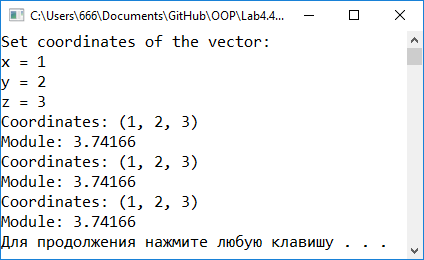


Рис. 38 Результат работы программы №4

## 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** <iostream>

**#include** <cmath>

**using** **namespace** std;

**class** Complex {

**double** re;

**double** im;

**public**:

**Complex**() {}

**Complex**(**double** re, **double** im) : re(re), im(im) {}

Complex **operator=** (**const** Complex& c);

**friend** **const** Complex **operator++** (Complex& c, **int**);

**friend** **const** Complex **operator--** (Complex& c, **int**);

**friend** **const** **bool** **operator==** (**const** Complex& c1, **const** Complex& c2);

**friend** **const** ostream& **operator<<** (ostream& out, **const** Complex& c);

**friend** **const** istream& **operator>>** (istream& in, Complex& c);

**~Complex**() {}

};

**int** **main**() {

Complex c1;

cout << "Enter complex number in format x+yi: ";

cin >> c1;

Complex c2 = c1;

**if** (c1 == c2) {

c2++;

}

c1--;

cout << c1;

cout << **endl** << c2;

cout << **endl**;

**system**("pause");

**return** 0;

}

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

**const** Complex **operator++** (Complex& c, **int**) {

Complex oldC(c);

c.re++;

**return** oldC;

}

**const** Complex **operator--** (Complex& c, **int**) {

Complex oldC(c);

c.re--;

**return** oldC;

}

**const** **bool** **operator==** (**const** Complex& c1, **const** Complex& c2) {

**return** (c1.re == c2.re) && (c1.im == c2.im);

}

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

out << c.re;

**if** (c.im < 0) {

out << c.im << "i";

} **else** {

out << "+" << c.im << "i";

}

**return** out;

}

**const** istream& **operator>>** (istream& in, Complex& c) {

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

in.get();

**return** in;

}

1. Программа №2

**#include** <iostream>

**#include** <cmath>

**using** **namespace** std;

**class** Vector {

**double** x;

**double** y;

**double** z;

**public**:

**Vector**() : x(0), y(0), z(0) {}

**Vector**(**double** x, **double** y, **double** z) : x(x), y(y), z(z) {};

**Vector**(**const** Vector& v);

Vector **operator=** (**const** Vector& v);

**friend** Vector **operator+** (**const** Vector& v1, **const** Vector& v2);

**friend** Vector **operator-** (**const** Vector& v1, **const** Vector& v2);

**friend** **const** **bool** **operator==** (**const** Vector& v1, **const** Vector& v2);

**friend** **const** istream& **operator>>** (istream& in, Vector& v);

**friend** **const** ostream& **operator<<** (ostream& out, **const** Vector& v);

};

**int** **main**() {

cout << "Set coordinates of the vector:";

Vector v1;

cin >> v1;

Vector v2 = v1;

**if** (v1 == v2) {

cout << v1;

}

cout << **endl**;

**system**("pause");

**return** 0;

}

**Vector::Vector**(**const** Vector& v) {

x = v.x;

y = v.y;

z = v.z;

}

Vector **Vector::operator=** (**const** Vector& v){**return** Vector(v.x, v.y, v.z);}

Vector **operator+** (**const** Vector& v1, **const** Vector& v2) {

**return** Vector(v1.x + v2.x, v1.y + v2.y, v1.z + v2.z);

}

Vector **operator-** (**const** Vector& v1, **const** Vector& v2) {

**return** Vector(v1.x - v2.x, v1.y - v2.y, v1.z - v2.z);

}

**const** **bool** **operator==** (**const** Vector& v1, **const** Vector& v2) {

**return** (v1.x == v2.x) && (v1.y == v2.y) && (v1.z == v2.z);

}

**const** istream& **operator>>** (istream& in, Vector& v) {

in >> v.x >> v.y >> v.z;

**return** in;

}

**const** ostream& **operator<<** (ostream& out, **const** Vector& v) {

out << v.x << " " << v.y << " " << v.z;

**return** out;

}

**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 рисунках (рис. 39, 40). Исходный код программ также доступен по ссылкам на GitHub:

1. <https://github.com/IvLaptev/OOP/tree/master/Lab5.1>
2. <https://github.com/IvLaptev/OOP/tree/master/Lab5.2>

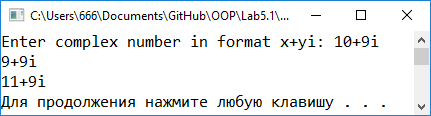


Рис. 39 Результат работы программы №1

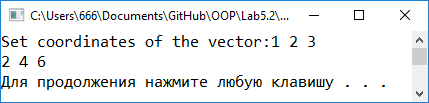


Рис. 40 Результат работы программы №2

## 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** <iostream>

**#include** <string>

**#include** <Windows.h>

**using** **namespace** std;

**class** Student { // базовый класс "Студент"

string name;

string surname;

**int** age;

**public**:

**Student**(string &name, string &surname, **int** age) : name(name), surname(surname), age(age) {};

string &**getName**() { **return** name; }

string &**getSurname**() { **return** surname; }

**int** **getAge**() { **return** age; }

};

**class** Headman : **public** Student { // Класс "Староста", производный от "Студент"

string phone;

string email;

**public**:

**Headman**(string &name, string &surname, **int** age, string &phone, string &email) :

Student(name, surname, age), phone(phone), email(email) {};

string &**getEmail**() { **return** email; }

string &**getPhone**() { **return** phone; }

**friend** **const** ostream& **operator<<** (ostream& out, **const** Headman& h);

};

**int** **main**() {

**SetConsoleCP**(1251);

**SetConsoleOutputCP**(1251);

string name, surname, phone, email;

**int** age;

cout << "Enter name: ";

cin >> name;

cout << "Enter surname: ";

cin >> surname;

cout << "Enter age: ";

cin >> age;

cout << "Enter phone: ";

cin >> phone;

cout << "Enter email: ";

cin >> email;

Headman h(name, surname, age, phone, email);

cout << **endl** << "Information about headman:" << **endl**;

cout << "Name: " << h.getName() << **endl**;

cout << "Surname: " << h.getSurname() << **endl**;

cout << "Age: " << h.getAge() << **endl**;

cout << "Phone: " << h.getPhone() << **endl**;

cout << "Email: " << h.getEmail() << **endl**;

cout << "Crown phrase: " << h;

cout << **endl**;

**system**("pause");

**return** 0;

}

**const** ostream& **operator<<** (ostream& out, **const** Headman& h) {

out << "I am your headman!";

**return** out;

}

1. Программа №2

**#include** <iostream>

**#include** <string>

**#include** <Windows.h>

**using** **namespace** std;

**class** Alive { // Базовый класс

**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 { // Производные классы

**public**:

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

**void** **move**() { cout << name << " is running" << **endl**; }

};

**class** Fish : **public** Alive {

**public**:

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

**void** **move**() { cout << name << " is swimming" << **endl**; }

};

**class** Bird : **public** Alive {

**public**:

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

**void** **move**() { cout << name << " is flying"; }

};

**int** **main**() {

**SetConsoleCP**(1251);

**SetConsoleOutputCP**(1251);

string name;

**int** age;

cout << "Fish" << **endl**;

cout << "Enter name: ";

cin >> name;

cout << "Enter age: ";

cin >> age;

Fish f(name, age);

cout << **endl** << "Animal" << **endl**;

cout << "Enter name: ";

cin >> name;

cout << "Enter age: ";

cin >> age;

Animal a(name, age);

cout << **endl** << "Bird" << **endl**;

cout << "Enter name: ";

cin >> name;

cout << "Enter age: ";

cin >> age;

Bird b(name, age);

cout << **endl** << "Your bird" << **endl** << "Name: " << b.getName() << **endl** << "Age: " <<

b.getAge() << **endl**;

b.move();

cout << **endl** << "Your fish" << **endl** << "Name: " << f.getName() << **endl** << "Age: " <<

f.getAge() << **endl**;

f.move();

cout << **endl** << "Your animal" << **endl** << "Name: " << a.getName() << **endl** << "Age: " <<

a.getAge() << **endl**;

a.move();

**system**("pause");

**return** 0;

}

1. Программа №3

**#include** <iostream>

**#include** <string>

**#include** <Windows.h>

**using** **namespace** std;

**class** Animal { // Базовый класс

**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** **talk**() = 0;

};

**class** Dog : **public** Animal { // Производные классы

**public**:

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

**void** **talk**() { cout << "woof" << **endl**; }

};

**class** Cat : **public** Animal {

**public**:

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

**void** **talk**() { cout << "meow" << **endl**; }

};

**int** **main**() {

**SetConsoleCP**(1251);

**SetConsoleOutputCP**(1251);

string name;

**int** age;

cout << "Cat" << **endl**;

cout << "Enter name: ";

cin >> name;

cout << "Enter age: ";

cin >> age;

Cat c(name, age);

cout << **endl** << "Dog" << **endl**;

cout << "Enter name: ";

cin >> name;

cout << "Enter age: ";

cin >> age;

Dog d(name, age);

cout << **endl** << "Your dog" << **endl** << "Name: " << d.getName() << **endl** << "Age: " <<

d.getAge() << **endl**;

d.talk();

cout << **endl** << "Your cat" << **endl** << "Name: " << c.getName() << **endl** << "Age: " <<

c.getAge() << **endl**;

c.talk();

**system**("pause");

**return** 0;

}

**Вывод**

Все поставленные задачи были выполнены. В процессе выполнения работы было изучено и использовано наследование. Результат работы программ изображены на рисунках (рис. 41-43). Исходный код программ также можно просмотреть на GitHub:

1. <https://github.com/IvLaptev/OOP/tree/master/Lab6.1>
2. <https://github.com/IvLaptev/OOP/tree/master/Lab6.2>
3. <https://github.com/IvLaptev/OOP/tree/master/Lab6.3>

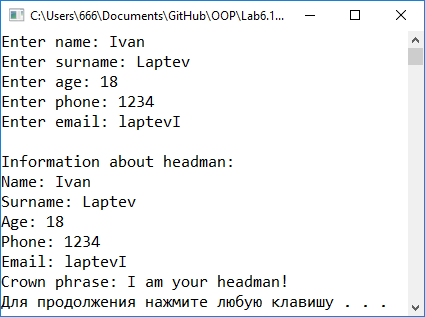


Рис. 41 Результат работы программы №1

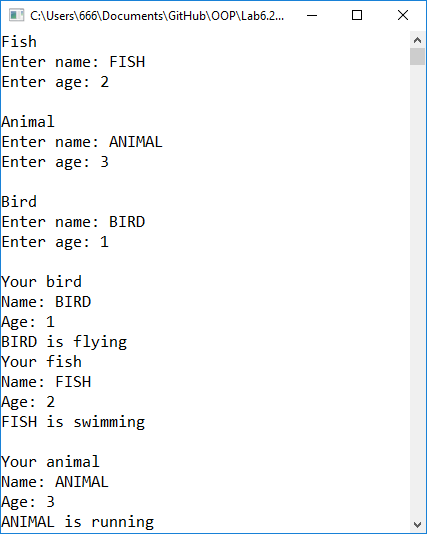


Рис. 42 Результат работы программы №2

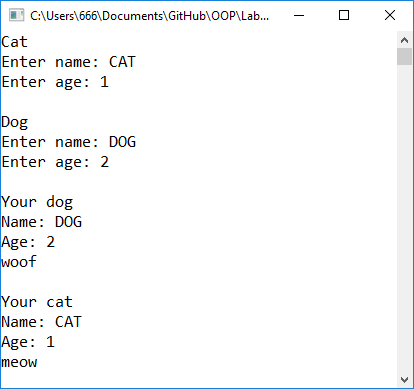


Рис. 43 Результат работы программы №3

## 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 <pch.h> |
|  |  |
|  | using namespace std; |
|  |  |
|  | class Animal { |
|  | protected: |
|  | string name; |
|  | int year; |
|  | public: |
|  | Animal(string name, int year) { |
|  | this->name = name; |
|  | this->year = year; |
|  | } |
|  | Animal() { |
|  | cout << "\_ \_ \_ \_ \_ \_ \_ \_ \_ \_ \_ \_ \_ \_ \_ \_ \_" << endl; |
|  | cout<<"Введите кол-во лет: "<<endl; |
|  | cout << "\_ \_ \_ \_ \_ \_ \_ \_ \_ \_ \_ \_ \_ \_ \_ \_ \_" << endl; |
|  | cin >> name; |
|  | cin >> year; |
|  | } |
|  | virtual void golos() = 0; |
|  | void get\_Data() { |
|  | cout << "\_ \_ \_ \_ \_ \_ \_ \_ \_ \_ \_ \_ \_ \_ \_ \_ \_" << endl; |
|  | cout << "Ya" << name << " " << year << endl; |
|  | cout << "\_ \_ \_ \_ \_ \_ \_ \_ \_ \_ \_ \_ \_ \_ \_ \_ \_" << endl; |
|  | } |
|  | }; |
|  |  |
|  | class Cat: public Animal { |
|  | public: |
|  | Cat(string name, int year):Animal(name,year){} |
|  | Cat():Animal(){} |
|  | void golos() { |
|  | cout << "\_ \_ \_ \_ \_ \_ \_ \_ \_ \_ \_ \_ \_ \_ \_ \_ \_" << endl; |
|  | cout << "Мяу."<<endl; |
|  | cout << "\_ \_ \_ \_ \_ \_ \_ \_ \_ \_ \_ \_ \_ \_ \_ \_ \_" << endl; |
|  | } |
|  |  |
|  | }; |
|  |  |
|  | class Dog : public Animal { |
|  | public: |
|  | Dog(string name, int year) :Animal(name, year) {} |
|  | Dog() :Animal() {} |
|  | void golos() { |
|  | cout << "\_ \_ \_ \_ \_ \_ \_ \_ \_ \_ \_ \_ \_ \_ \_ \_ \_" << endl; |
|  | cout << "Гав, Гав." << endl; |
|  | cout << "\_ \_ \_ \_ \_ \_ \_ \_ \_ \_ \_ \_ \_ \_ \_ \_ \_" << endl; |
|  | } |
|  |  |
|  | }; |
|  |  |
|  | class Parrot : public Animal { |
|  | public: |
|  | Parrot(string name, int year) :Animal(name, year) {} |
|  | Parrot() :Animal() {} |
|  | void golos() { |
|  | cout << "\_ \_ \_ \_ \_ \_ \_ \_ \_ \_ \_ \_ \_ \_ \_ \_ \_" << endl; |
|  | cout << "Карррр." << endl; |
|  | cout << "\_ \_ \_ \_ \_ \_ \_ \_ \_ \_ \_ \_ \_ \_ \_ \_ \_" << endl; |
|  | } |
|  |  |
|  | }; |
|  | int main(){ |
|  | Cat a; |
|  | a.get\_Data(); |
|  | a.golos(); |
|  |  |
|  | Dog b; |
|  | b.get\_Data(); |
|  | b.golos(); |
|  |  |
|  | Parrot c; |
|  | c.get\_Data(); |
|  | c.golos(); |
|  | system("pause"); |
|  | return 0; |
|  | } |

**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 рисунке (рис. 44). Код программы также доступен на GitHub по ссылке:

<https://github.com/IvLaptev/OOP/tree/master/Lab7>

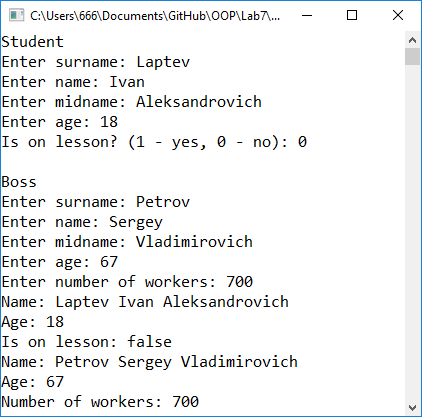


Рис. 44 Результат работы программы

## 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>

**#include** <Windows.h>

**using** **namespace** std;

**class** Student {

string name;

string surname;

**int** age;

**public**:

**Student**(string &name, string &surname, **int** age) : name(name), surname(surname), age(age) {};

string &**getName**() { **return** name; }

string &**getSurname**() { **return** surname; }

**int** **getAge**() { **return** age; }

};

**int** **main**() {

**SetConsoleCP**(1251);

**SetConsoleOutputCP**(1251);

vector<Student> students;

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

string name, surname;

**int** age;

cout << "Student " << i+1 << **endl**;

cout << "Enter name: ";

cin >> name;

cout << "Enter surname: ";

cin >> surname;

cout << "Enter age: ";

cin >> age;

students.push\_back(Student(name, surname, age));

}

**for** (**auto** a : students) {

cout << **endl** << "Name: " << a.getName() << **endl**;

cout << "Surname: " << a.getSurname() << **endl**;

cout << "Age: " << a.getAge() << **endl**;

}

**system**("pause");

**return** 0;

}

1. Программа №2

Класс Complex был взят из лабораторной работы №1.

**#include** <iostream>

**#include** <cmath>

**#include** <list>

**using** **namespace** std;

**int** **main**() {

list<Complex> l;

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

**double** re, im;

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

cin >> re;

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

cin >> im;

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

}

**for** (**auto** a : l) {

cout << **endl**;

a.displayData();

}

**system**("pause");

**return** 0;

}

1. Программа №3

**#include** <iostream>

**#include** <string>

**#include** <queue>

**#include** <Windows.h>

**using** **namespace** std;

**class** Client {

string name;

string car;

**public**:

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

**void** **getData**(); // Печатает информацию о клиенте

};

**int** **main**() {

queue<Client> q; // Создаётся очередь из объектов класса Client

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

string name, car;

cout << "Client " << i+1 << **endl**;

cout << "Enter name: ";

cin >> name;

cout << "Enter car model: ";

cin >> car;

q.push(Client(name, car)); // В конец очереди ставится новый объект

}

cout << "Queue" << **endl**;

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

cout << i;

q.front().getData(); // Берётся первый объект из очереди

q.pop(); // Удаляется первый объект очереди

}

**return** 0;

}

**void** **Client::getData**() {

cout << **endl** << "Name: " << name << **endl**;

cout << "Car: " << car << **endl**;

}

**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 рисунках (рис. 45-47). Исходный код программ так же доступен на GitHub по ссылкам:

1. <https://github.com/IvLaptev/OOP/tree/master/Lab8.1>
2. <https://github.com/IvLaptev/OOP/tree/master/Lab8.2>
3. <https://github.com/IvLaptev/OOP/tree/master/Lab8.3>

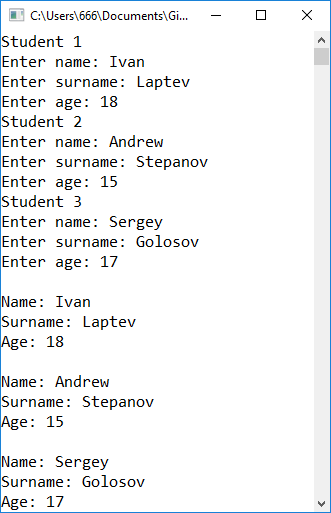


Рис. 45 Результат работы программы №1

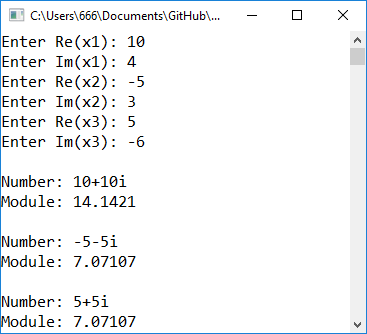


Рис. 46 Результат работы программы №2

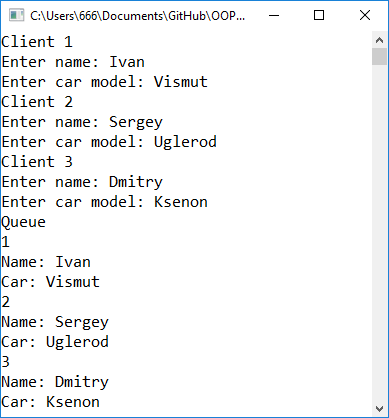


Рис. 47 Результат работы программы №3