|  |  |
| --- | --- |
| 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 | Чирков К.Р. |
| Принял ассистент кафедры | Хлебникова В.Л. |

|  |  |  |
| --- | --- | --- |
| Лабораторные работы выполнены | «27» апреля 2019 г. |  |
|  |  |  |
| «Зачтено» | «\_\_»\_\_\_\_\_\_\_201\_\_ г. |  |

Москва 2019

**Содержание**

[Таблица успеваемости 2](#_2et92p0)

[Практическая работа №1](#_tyjcwt) 4

[Практическая работа №2](#_1t3h5sf) 6

[Практическая работа №3](#_4d34og8) 10

[Практическая работа №4](#_2s8eyo1) 22

[Практическая работа №5](#_17dp8vu) 27

[Практическая работа №6](#_3rdcrjn) 35

[Практическая работа №7](#_26in1rg) 43

[Практическая работа №8](#_lnxbz9) 48

[Лабораторная работа №1](#_35nkun2) 51

[Лабораторная работа №2](#_1ksv4uv) 55

[Лабораторная работа №3](#_44sinio) 59

[Лабораторная работа №4](#_2jxsxqh) 66

[Лабораторная работа №5](#_z337ya) 74

[Лабораторная работа №6](#_3j2qqm3) 78

[Лабораторная работа №7](#_1y810tw) 85

[Лабораторная работа №8](#_4i7ojhp) 88

## 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**

1. Реализовать программу, которая считывает имя и здоровается
2. Реализовать программу, производящую операции над числами. (Сложение/вычитание, целые числа/числа с плавающей запятой)

**Ход работы**

1. Реализовать программу, которая считывает имя и здоровается

#include <iostream>

using namespace std;

int main()

{

char name[20];

cout << "Enter your name!" << endl;

cin.getline(name, 19);

cout << "Hello, " << name << "!" << endl;

system("pause");

return 0;

}

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

#include <iostream>

using namespace std;

int main()

{

double a, b, s, r, u, d;

cout << "Vvedite 2 chisla v poryadke ubivaniya" << endl;

cin >> a >> b;

s = a + b;

r = a - b;

u = a \* b;

if (b != 0)

d = a / b;

cout << "Summa = " << s << endl;

cout << "Raznost = " << r << endl;

cout << "Umnozhenie = " << u << endl;

cout << "Delenie = " << d << 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/kchirkov/OOP/tree/master/pr11>
2. <https://github.com/kchirkov/OOP/tree/master/pr12>

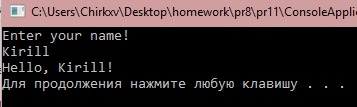


Рис. 1 Результат работы первой программы

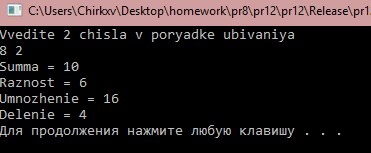


Рис. 2 Результат работы второй программы

## 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**

1. Вариант 1

#include <iostream>

#include <math.h>

using namespace std;

int main()

{

double a, b, n, m;

a = 4.8;

b = -7.9;

if (a >= b)

n = pow((a - b), (1 / 3.0));

else

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

if (n < b)

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

if (n == b)

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

if (n > b)

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

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

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

cout << "n = " << n << endl;

cout << "m = " << m << endl;

system("pause");

return 0;

}

1. Вариант 2

#include <iostream>

#include <math.h>

using namespace std;

int main()

{

double a, b, z, t;

a = -0.6;

b = 5.3;

if (a >= b)

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

else

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

if (z < b)

t = pow((z + a \* a\*b), (1 / 3.0));

if (z == b)

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

if (z > b)

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

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

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

cout << "z = " << z << endl;

cout << "t = " << t << endl;

system("pause");

return 0;

}

1. Вариант 3

#include <iostream>

#include <math.h>

using namespace std;

int main()

{

double a, b, y, t;

a = 4.6;

b = 1.5;

if (a <= b)

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

else

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

if (y == b)

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

if (y < b)

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

if (y > b)

t = -pow(-y \* sin(a), 1 / 3.0) + 1 / sqrt(y\*sin(b));

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

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

cout << "y = " << y << endl;

cout << "t = " << t << endl;

system("pause");

return 0;

}

1. Вариант 4

#include <iostream>

#include <math.h>

using namespace std;

int main()

{

double a, x, y, t;

a = 3.5;

x = -0.7;

if (a <= x)

y = a + log(x + a);

else

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

if (y < a)

t = y / (a - x);

if (y = a)

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

if (y > a)

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

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

cout << "x = " << x << endl;

cout << "y = " << y << endl;

cout << "t = " << t << 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/kchirkov/OOP/tree/master/pr21>
2. <https://github.com/kchirkov/OOP/tree/master/pr21>
3. <https://github.com/kchirkov/OOP/tree/master/pr23>
4. <https://github.com/kchirkov/OOP/tree/master/pr24>

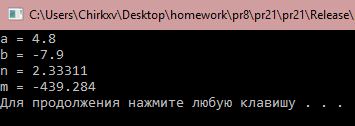


Рис. 3 Результат практической работы №2 вариант 1

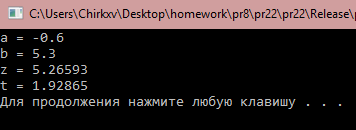


Рис. 4 Результат практической работы №2 вариант 2

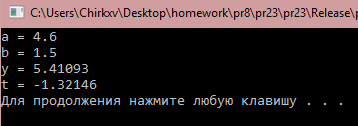


Рис. 5 Результат практической работы №2 вариант 3

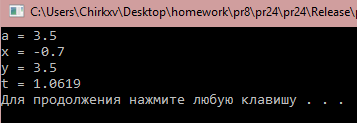


Рис. 6 Результат практической работы №2 вариант 4

## 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**

1. Реализовать класс «Книга»
2. Реализовать класс «Группа студентов»
3. Реализовать класс «Автомобиль»
4. Реализовать класс «Вектор»
5. Реализовать класс «Библиотека»

**Ход работы**

1. Реализовать класс «Книга»

#include <iostream>

#include <string>

using namespace std;

class kniga {

string janr;

string avtor;

string nazvanie;

int god\_izdania;

int cena;

public:

void set()

{

cout << "Vvedite janr:";

getline(cin, janr);

cout << "Vvedite avtora:";

getline(cin, avtor);

cout << "Vvedite nazvanie:";

getline(cin, nazvanie);

cout << "Vvedite god vipyska:";

cin >> god\_izdania;

cout << "Vvedite ceny:";

cin >> cena;

}

void show()

{

cout << "Janr:" << janr << endl;

cout << "Avtor:" << avtor << endl;

cout << "Nazvanie:" << nazvanie << endl;

cout << "God vipuska:" << god\_izdania << endl;

cout << "Price:" << cena << endl;

}

};

int main()

{

kniga n1;

n1.set();

n1.show();

system("pause");

return 0;

}

1. Реализовать класс «Группа студентов»

#include <iostream>

#include <string>

using namespace std;

class groupofStudents {

string name;

int kurs;

string napr;

int numberof;

string starosta;

public:

void set\_name(string stname)

{

name = stname;

}

string get\_name()

{

return name;

}

void set\_starosta(string ststarosta)

{

starosta = ststarosta;

}

string get\_starosta()

{

return starosta;

}

void set\_numberof(int stnumberof)

{

numberof = stnumberof;

}

int get\_numberof()

{

return numberof;

}

void set\_napr(string stnapr)

{

napr = stnapr;

}

string get\_napr()

{

return napr;

}

void set\_kurs(int stkurs)

{

kurs = stkurs;

}

int get\_kurs()

{

return kurs;

}

};

int main()

{

groupofStudents gruppa;

int kurs, numberof;

string napr, starosta, name;

cout << "Vvedite nazvanie gruppi: " << endl;

getline(cin, name);

cout << "Vvedite kurs: " << endl;

cin >> kurs;

cin.ignore();

cout << "Vvedite napravlenie " << endl;

getline(cin, napr);

cout << "Vvedite kolichestvo studentov" << endl;

cin >> numberof;

cin.ignore();

cout << "Vvedite imya starosti" << endl;

getline(cin, starosta);

gruppa.set\_name(name);

gruppa.set\_kurs(kurs);

gruppa.set\_napr(napr);

gruppa.set\_numberof(numberof);

gruppa.set\_starosta(starosta);

cout << gruppa.get\_name() << endl;

cout << gruppa.get\_kurs() << endl;

cout << gruppa.get\_napr() << endl;

cout << gruppa.get\_numberof() << endl;

cout << gruppa.get\_starosta() << endl;

system("pause");

return 0;

}

1. Реализовать класс «Автомобиль»

#include <iostream>

#include <string>

using namespace std;

class autom {

string brand;

int price;

string color;

string name;

string tip;

public:

void set\_name(string stname)

{

name = stname;

}

string get\_name()

{

return name;

}

void set\_brand(string stbrand)

{

brand = stbrand;

}

string get\_brand()

{

return brand;

}

void set\_price(int stprice)

{

price = stprice;

}

int get\_price()

{

return price;

}

void set\_color(string stcolor)

{

color = stcolor;

}

string get\_color()

{

return color;

}

void set\_tip(string sttip)

{

tip = sttip;

}

string get\_tip()

{

return tip;

}

};

int main()

{

autom auto1;

int price;

string brand, color, name, tip;

cout << "Vvedite marku auto: " << endl;

getline(cin, brand);

cout << "Vvedite nazvanie auto: " << endl;

getline(cin, name);

cout << "Vvedite tsvet " << endl;

getline(cin, color);

cout << "Vvedite price: " << endl;

cin >> price;

cin.ignore();

cout << "Vvedite tip auto: " << endl;

getline(cin, tip);

auto1.set\_name(name);

auto1.set\_price(price);

auto1.set\_color(color);

auto1.set\_brand(brand);

auto1.set\_tip(tip);

cout << auto1.get\_brand() << endl;

cout << auto1.get\_name() << endl;

cout << auto1.get\_tip() << endl;

cout << auto1.get\_color() << endl;

cout << auto1.get\_price() << endl;

system("pause");

return 0;

}

1. Реализовать класс «Вектор»

#include <iostream>

#include <string>

#include <math.h>

using namespace std;

class vektor {

int x;

int y;

int z;

double dlin;

public:

void set\_x(int stx)

{

x = stx;

}

int get\_x()

{

return x;

}

void set\_y(int sty)

{

y = sty;

}

int get\_y()

{

return y;

}

void set\_z(int stz)

{

z = stz;

}

int get\_z()

{

return z;

}

void count\_dlin(int x, int y, int z)

{

dlin = sqrt(x\*x + y \* y + z \* z);

}

double get\_dlin()

{

return dlin;

}

};

int main()

{

vektor vektor1;

int x, y, z;

cout << "Vvedite x" << endl;

cin >> x;

cout << "Vvedite y" << endl;

cin >> y;

cout << "Vvedite z" << endl;

cin >> z;

vektor1.set\_x(x);

vektor1.set\_y(y);

vektor1.set\_z(z);

vektor1.count\_dlin(x, y, z);

cout << vektor1.get\_x() << endl;

cout << vektor1.get\_y() << endl;

cout << vektor1.get\_z() << endl;

cout << vektor1.get\_dlin() << endl;

system("pause");

return 0;

}

1. Реализовать класс «Библиотека»

Файл library.h

pragma once

#include "pch.h"

#include <vector>

#include <string>

#include <iostream>

using namespace std;

class library

{

vector<library> lib;

int year;

string author, title;

public:

library();

void addBook(library);

void getBooks();

void setyear(int);

int getyear();

void setauthor(string);

string getauthor();

void settitle(string);

string gettitle();

~library();

};

Файл library.cpp

#include "pch.h"

#include "library.h"

library::library() {}

void library::addBook(library book) {

lib.push\_back(book);

}

void library::getBooks() {

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

cout << i + 1 <<

": Title: " << lib[i].title << endl <<

"Author: " << lib[i].author << endl <<

"Year: " << lib[i].year << endl;

}

}

void library::setyear(int year) {

this->year = year;

}

int library::getyear() {

return year;

}

void library::setauthor(string author) {

this->author = author;

}

string library::getauthor() {

return author;

}

void library::settitle(string title) {

this->title = title;

}

string library::gettitle() {

return title;

}

library::~library(){}

Файл pr35.cpp

#include "pch.h"

#include "library.h"

int main()

{

library book1, book2,lib;

int year;

string title, author;

cout << "First book:"<<endl;

cout << "Enter author: ";

getline(cin, author);

book1.setauthor(author);

cout << "Enter year: ";

cin >> year;

cin.ignore();

book1.setyear(year);

cout << "Enter title: ";

getline(cin, title);

book1.settitle(title);

lib.addBook(book1);

cout << "Second book:"<<endl;

cout << "Enter author: ";

getline(cin, author);

book2.setauthor(author);

cout << "Enter year: ";

cin >> year;

cin.clear();

cin.ignore();

book2.setyear(year);

cout << "Enter title: ";

getline(cin, title);

book2.settitle(title);

cout << endl;

lib.addBook(book2);

lib.getBooks();

system("pause");

return 0;}

**Вывод**

В ходе работы были получены знания по созданию классов. Результаты работы программ приведены в рисунках (рис. 7-11). Код программ также доступен на GitHub по ссылкам:

1. <https://github.com/kchirkov/OOP/tree/master/pr31>
2. <https://github.com/kchirkov/OOP/tree/master/pr32>
3. <https://github.com/kchirkov/OOP/tree/master/pr33>
4. <https://github.com/kchirkov/OOP/tree/master/pr34>
5. <https://github.com/kchirkov/OOP/tree/master/pr35>

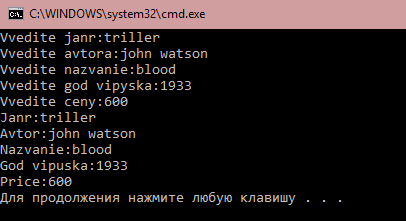


Рис. 7 Результат работы программы №1

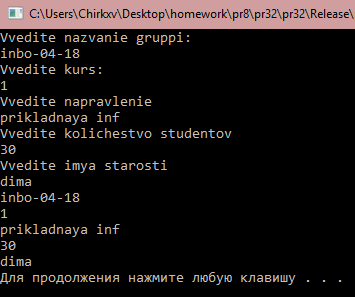


Рис. 8 Результат работы программы №2

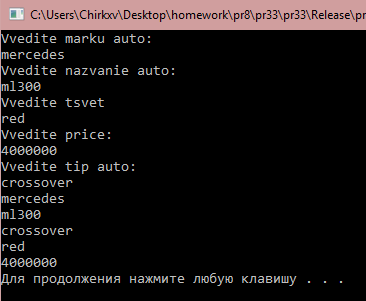


Рис. 9 Результат работы программы №3

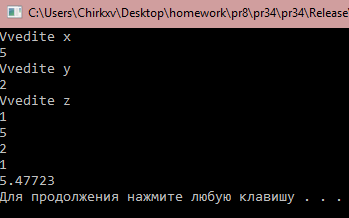


Рис 10 Результат работы программы №4

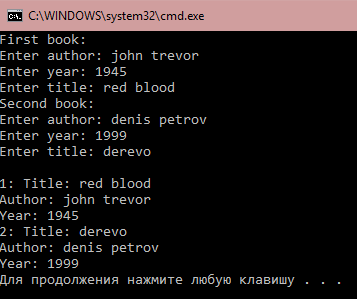


Рис. 11 Результат работы программы №5

## 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 также работа с указателями на языке С++.

**Задачи**

1. 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.
2. Объявите указатель на массив типа int и выделите память для 12-ти элементов. Необходимо написать функцию, которая поменяет значения четных и нечетных ячеек массива.
3. 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**

1. Программа №1

#include "pch.h"

#include <iostream>

#include <ctime>

using namespace std;

double\* func1(int a)

{

return new double[a];

}

void func2(int a, double \*b)

{

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

b[i] = rand() % 20;

}

void func3(int a, double \*b)

{

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

cout << b[i] << " ";

cout << endl;

}

void func4(double \*b)

{

delete[] b;

}

int main()

{

start:

srand(time(0));

int size, ind;

double \*massiv;

cout << "Vvedite razmer massiva" << endl;

cin >> size;

massiv = func1(size);

func2(size, massiv);

func3(size, massiv);

func4(massiv);

cout << "stop or contunie? (0/1)" << endl;

cin >> ind;

if (ind == 1)

goto start;

else

return 0;

}

1. Программа №2

#include "pch.h"

#include <iostream>

#include <ctime>;

using namespace std;

int main()

{

int \*a = new int[12];

for (int i = 0; i < 12; i++)

{

a[i] = rand() % 20;

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

}

cout << endl;

for (int i = 0; i < 11; i++)

{

int c = a[i];

a[i] = a[i + 1];

a[i + 1] = c;

i++;

}

for (int i = 0; i < 12; i++)

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

system("pause");

return 0;

}

1. Программа №3

#include "pch.h"

#include <iostream>

#include <ctime>

using namespace std;

void func1(int a, int b, int \*\*c)

{

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

for (int j = 0; j < b; j++)

c[i][j] = rand() % 40 + 10;

}

void func2(int a, int b, int \*\*c)

{

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

{

for (int j = 0; j < b; j++)

cout << c[i][j] << " ";

cout << endl;

}

}

int main()

{

srand(time(0));

int c1, c2;

cout << "vvedite kolichesvo strok" << endl;

cin >> c1;

cout << "vvedite kolichesvo stolbcov" << endl;

cin >> c2;

int \*\*a = new int\*[c1];

for (int i = 0; i < c1; i++)

a[i] = new int[c2];

func1(c1, c2, a);

func2(c1, c2, a);

for (int i = 0; i < c1; i++)

delete[] a[i];

delete[] a;

system("pause");

return 0;

}

**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/kchirkov/OOP/tree/master/pr41>
2. <https://github.com/kchirkov/OOP/tree/master/pr42>
3. <https://github.com/kchirkov/OOP/tree/master/pr43>

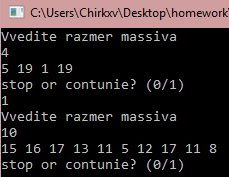


Рис. 12 Результат работы программы №1

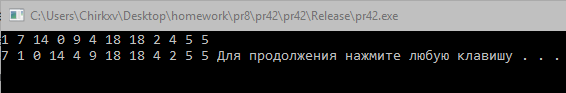


Рис. 13 Результат работы программы №2

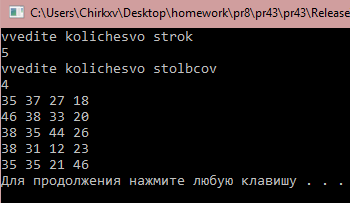


Рис. 14 Результат работы программы №3

## 

## Практическая работа №5

**Перегрузка функций**

**Цель практической работы**

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

**Задачи**

1. Реализовать сортировку пузырьком для целых чисел, а затем перегрузить её для дробных.
2. Реализовать сортировку выбором для целых чисел, а затем перегрузить её для дробных.
3. Реализовать сортировку вставками для целых чисел, а затем перегрузить её для дробных.
4. Реализовать программу-калькулятор, работающую с разными типами данных.

**Ход работы**

1. Программа №1

#include "pch.h"

#include <iostream>

using namespace std;

void bubblesort(int n, int \*mass)

{

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

{

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

{

if (mass[r] < mass[r + 1])

{

int temp = mass[r];

mass[r] = mass[r + 1];

mass[r + 1] = temp;

}

}

}

}

void bubblesort(int n, double \*mass)

{

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

{

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

{

if (mass[r] < mass[r + 1])

{

double temp = mass[r];

mass[r] = mass[r + 1];

mass[r + 1] = temp;

}

}

}

}

int main()

{

int n;

cout << "Enter count: ";

cin >> n;

double\* mass;

mass = new double[n];

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

{

cout << i + 1 << " element: ";

cin >> mass[i];

}

cout << "Array: ";

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

{

cout << mass[i] << " ";

}

cout << endl;

bubblesort(n, mass);

cout << "Sorted array: ";

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

{

cout << mass[i] << " ";

}

cout << endl;

delete[] mass;

system("pause");

return 0;

}

1. Программа №2

#include "pch.h"

#include <iostream>

using namespace std;

void selectionSort(int \*mass, int size)

{

int min, temp;

for (int i = 0; i < size - 1; i++)

{

min = i;

for (int j = i + 1; j < size; j++)

{

if (mass[j] < mass[min])

min = j;

}

temp = mass[i];

mass[i] = mass[min];

mass[min] = temp;

}

}

void selectionSort(double \*mass, int size)

{

int min;

double temp;

for (int i = 0; i < size - 1; i++)

{

min = i;

for (int j = i + 1; j < size; j++)

{

if (mass[j] < mass[min])

min = j;

}

temp = mass[i];

mass[i] = mass[min];

mass[min] = temp;

}

}

int main()

{

int n;

cout << "Count of elements: ";

cin >> n;

double\* mass;

mass = new double[n];

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

{

cout << i + 1 << " element ";

cin >> mass[i];

}

cout << "Array: ";

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

{

cout << mass[i] << " ";

}

cout << endl;

selectionSort(mass, n);

cout << "Sorted array: ";

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

{

cout << mass[i] << " ";

}

cout << endl;

delete[] mass;

system("pause");

return 0;

}

1. Программа №3

#include "pch.h"

#include <iostream>

using namespace std;

void insertSort(int \*mass, int n)

{

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

for (int j = i; j > 0 && mass[j - 1] > mass[j]; j--)

swap(mass[j - 1], mass[j]);

}

void insertSort(double \*mass, int n)

{

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

for (int j = i; j > 0 && mass[j - 1] > mass[j]; j--)

swap(mass[j - 1], mass[j]);

}

int main()

{

int n;

cout << "Count of elements: ";

cin >> n;

double\* mass;

mass = new double[n];

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

{

cout << i + 1 << " element ";

cin >> mass[i];

}

cout << "Array: ";

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

{

cout << mass[i] << " ";

}

cout << endl;

insertSort(mass, n);

cout << "Sorted array: ";

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

{

cout << mass[i] << " ";

}

cout << endl;

delete[] mass;

system("pause");

return 0;

}

1. Программа №4

#include "pch.h"

#include <iostream>

using namespace std;

int sum(int a, int b)

{

return a + b;

}

int raz(int a, int b)

{

return a - b;

}

double sum(double a, double b)

{

return a + b;

}

double raz(double a, double b)

{

return a - b;

}

int main()

{

int s1 = sum(1, 2);

int r1 = raz(2, 1);

double s2 = sum(1.1, 2.2);

double r2 = raz(2.2, 1.1);

cout <<"sum and raz int chisel"<<endl<< s1 << endl << r1 << endl<<"sum and raz double chisel "<<endl << s2 << endl << r2 << endl;

system("pause");

return 0;

}

**Вывод**

Все задачи были выполнены. В ходе выполнения была изучена техника перегрузки функций. Также 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/kchirkov/OOP/tree/master/pr51>
2. <https://github.com/kchirkov/OOP/tree/master/pr52>
3. <https://github.com/kchirkov/OOP/tree/master/pr53>
4. <https://github.com/kchirkov/OOP/tree/master/pr54>

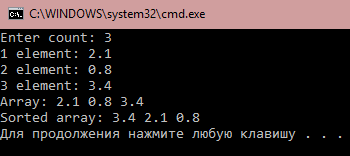


Рис. 15 Результат работы программы №1

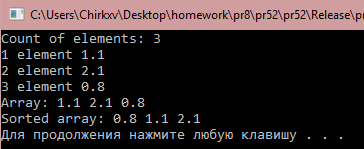


Рис. 16 Результат работы программы №2

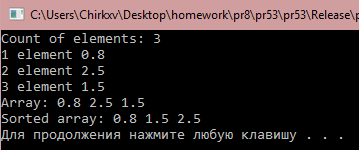


Рис. 17 Результат работы программы №3

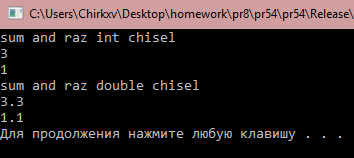


Рис. 18 Результат работы программы №4

## 

## 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**

1. 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.
2. 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**

1. Программа №1

#include "pch.h"

#include <iostream>

using namespace std;

class Property {

public:

double worth;

Property(double w)

{

worth = w;

}

virtual double tax() = 0;

~Property() {}

};

class Appartment : public Property {

public:

Appartment(double s1) : Property(s1) {}

double tax()

{

return worth / 1000;

}

};

class Car : public Property {

public:

Car(double s2) : Property(s2) {}

double tax()

{

return worth / 200;

}

};

class CountryHouse : public Property {

public:

CountryHouse(double s3) : Property(s3) {};

double tax()

{

return worth / 500;

}

};

int main()

{

Property \*M[7];

M[0] = new Appartment(10000000);

M[1] = new Appartment(20000000);

M[2] = new Appartment(30000000);

M[3] = new Car(1000000);

M[4] = new Car(2000000);

M[5] = new CountryHouse(5000000);

M[6] = new CountryHouse(6000000);

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

{

cout << "Velichina naloga dlya " << i + 1 << " objecta = " << M[i]->tax() << endl;

delete M[i];

}

system("pause");

return 0;

}

1. Программа №2

#include "pch.h"

#include <iostream>

#include <string>

#include <cmath>

using namespace std;

class vir {

public:

virtual void pechat() = 0;

virtual double vich() = 0;

virtual string proizvod() = 0;

};

class konst : public vir {

public:

double x;

konst(double ch) : x(ch) {}

void pechat()

{

cout << x << endl;

}

double vich()

{

return x;

}

string proizvod()

{

return "0";

}

};

class per : public vir {

public:

double x;

per(double ch) : x(ch) {}

void pechat()

{

cout << x << endl;

}

double vich()

{

return x;

}

string proizvod()

{

return "1";

}

};

class sum : public vir {

public:

double x, y;

sum(double ch1, double ch2) : x(ch1), y(ch2) {}

void pechat()

{

cout << x << "+" << y << endl;

}

double vich()

{

return x + y;

}

string proizvod()

{

return "1+y";

}

};

class raz : public vir {

public:

double x, y;

raz(double ch1, double ch2) : x(ch1), y(ch2) {}

void pechat()

{

cout << x << "-" << y << endl;

}

double vich()

{

return x - y;

}

string proizvod()

{

return "1-y";

}

};

class umn : public vir {

public:

double x, y;

umn(double ch1, double ch2) : x(ch1), y(ch2) {}

void pechat()

{

cout << x << "\*" << y << endl;

}

double vich()

{

return x \* y;

}

string proizvod()

{

return "y";

}

};

class del : public vir {

public:

double x, y;

del(double ch1, double ch2) : x(ch1), y(ch2) {}

void pechat()

{

cout << x << "/" << y << endl;

}

double vich()

{

return x / y;

}

string proizvod()

{

return "1/y";

}

};

class sinus : public vir {

public:

double x;

sinus(double ch1) : x(ch1) {}

void pechat()

{

cout << "sinx" << endl;

}

double vich()

{

return sin(x);

}

string proizvod()

{

return "cosx";

}

};

class cosinus : public vir {

public:

double x;

cosinus(double ch1) : x(ch1) {}

void pechat()

{

cout << "cosx" << endl;

}

double vich()

{

return cos(x);

}

string proizvod()

{

return "-sinx";

}

};

class expo : public vir {

public:

double x;

expo(double ch1) : x(ch1) {}

void pechat()

{

cout << "e^x" << endl;

}

double vich()

{

return exp(x);

}

string proizvod()

{

return "e^x";

}

};

class natlog : public vir {

public:

double x;

natlog(double ch1) : x(ch1) {}

void pechat()

{

cout << "lnx" << endl;

}

double vich()

{

return log(x);

}

string proizvod()

{

return "1/x";

}

};

int main()

{

sum a(3, 4);

a.pechat();

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

cout << "proizvodnaya po x = " << a.proizvod() << endl;

sinus b(2);

b.pechat();

cout << b.vich() << endl;

cout << "proizvodnaya po x = " << b.proizvod() << endl;

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/kchirkov/OOP/tree/master/pr61>
2. <https://github.com/kchirkov/OOP/tree/master/pr62>

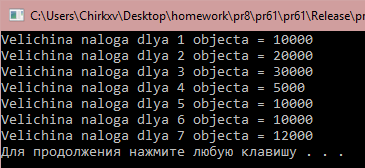


Рис. 19 Результат работы программы №1

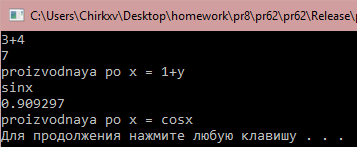


Рис. 20 Результат работы программы №2

## Практическая работа №7

**Абстрактные классы. Множественное наследование**

**Цель практической работы**

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

**Задачи**

1. Реализовать абстрактный класс «Животное» и путём наследования от него получить классы «Кошка», «Собака», «Попугай».
2. Реализовать абстрактный класс «Фигура» и путём наследования от него получить абстрактный класс «Четырёхугольник», и затем путём наследования получить классы «Ромб», «Прямоугольник».
3. Реализовать абстрактный класс «Транспортное средство» и путём наследования от него получить классы «Автомобиль», «Автобус», «Велосипед».
4. Реализовать абстрактные классы «Экран» и «Клавиатура», путём наследования от них получить классы «Ноутбук», «Телефон», «Стационарный компьютер».

**Ход работы**

1. Программа №1

#include "pch.h"

#include <iostream>

#include <string>

using namespace std;

class animal {

string name;

int legs;

int age;

public:

animal(string sname, int slegs, int sage) : name(sname), legs(slegs), age(sage) {}

string getname() { return name; }

int getlegs() { return legs; }

int getage() { return age; }

virtual void say() = 0;

};

class cat : public animal {

public:

cat(string sname, int slegs, int sage) : animal(sname, slegs, sage) {}

void say() { cout << "koshechka govorit nyaaaaa" << endl; }

};

class dog : public animal {

public:

dog(string sname, int slegs, int sage) : animal(sname, slegs, sage) {}

void say() { cout << "sobachka govorit gav" << endl; }

};

class parrot : public animal {

public:

parrot(string sname, int slegs, int sage) : animal(sname, slegs, sage) {}

void say() { cout << "popugaychik govorit durak" << endl; }

};

int main()

{

cat Neko("Neko", 4, 5);

cout << Neko.getname() << endl;

Neko.say();

system("pause");

return 0;

}

1. Программа №2

#include "pch.h"

#include <iostream>

#include <string>

using namespace std;

class figure {

int corners;

public:

figure(int scorners) : corners(scorners) {}

virtual void getsquareformula() = 0;

};

class chug : public figure {

public:

chug() : figure(4) {}

void getsquareformula() = 0;

};

class romb : public chug {

public:

void getsquareformula() { cout << "d1\*d2/2" << endl; }

};

class pryamoug : public chug {

public:

void getsquareformula() { cout << "a\*b" << endl; }

};

int main()

{

romb a;

pryamoug b;

a.getsquareformula();

b.getsquareformula();

system("pause");

return 0;

}

1. Программа №3

#include "pch.h"

#include <iostream>

#include <string>

using namespace std;

class transport {

string mark;

string name;

int age;

public:

int wheels;

transport(string smark, string sname, int swheels, int sage) : mark(smark), name(sname), wheels(swheels), age(sage) {}

string getmark() { return mark; }

string getname() { return name; }

int getwheels() { return wheels; }

int getage() { return age; }

virtual double maxspeedfakeformula() = 0;

};

class automobile : public transport {

public:

automobile(string smark, string sname, int swheels, int sage) : transport(smark, sname, swheels, sage) {}

double maxspeedfakeformula() { return wheels \* 60; }

};

class bus : public transport {

public:

bus(string smark, string sname, int swheels, int sage) : transport(smark, sname, swheels, sage) {}

double maxspeedfakeformula() { return wheels \* 30; }

};

class velo : public transport {

public:

velo(string smark, string sname, int swheels, int sage) : transport(smark, sname, swheels, sage) {}

double maxspeedfakeformula() { return wheels \* 20; }

};

int main()

{

automobile n1("mersedes", "ML", 4, 2);

bus n2("peugeot", "big", 4, 3);

cout <<"auto speed:"<< n1.maxspeedfakeformula() << endl;

cout <<"bus speed:"<< n2.maxspeedfakeformula() << endl;

system("pause");

return 0;

}

1. Программа №4

#include "pch.h"

#include <iostream>

using namespace std;

class screen {

int screensize;

public:

screen(double sscreensize) : screensize(sscreensize) {}

virtual void mediumsize() = 0;

};

class keyboard {

int keycaps;

public:

keyboard(double skeycaps) : keycaps(skeycaps) {}

virtual void height() = 0;

};

class smartphone : public screen, keyboard {

public:

smartphone(double sscreensize, double skeycaps) : screen(sscreensize), keyboard(skeycaps) {}

void mediumsize() { cout << "average screensize: 5 inch" << endl; }

void height() { cout << "virtual keyboard height=0" << endl; }

};

class pc : public screen, keyboard {

public:

pc(double sscreensize, double skeycaps) : screen(sscreensize), keyboard(skeycaps) {}

void mediumsize() { cout << "average screensize: 21 inch" << endl; }

void height() { cout << "average keyboard height = 2 cm" << endl; }

};

class laptop : public screen, keyboard {

public:

laptop(double sscreensize, double skeycaps) : screen(sscreensize), keyboard(skeycaps) {}

void mediumsize() { cout << "average screensize: 16 inch" << endl; }

void height() { cout << "average keyboard height= 0,5 cm" << endl; }

};

int main()

{

laptop a(6, 0);

pc b(24, 60);

a.mediumsize();

a.height();

b.mediumsize();

b.height();

system("pause");

return 0;

}

**Вывод**

Все задачи были выполнены. В ходе выполнения работы были более плотно изучены множественное наследование и абстрактные классы. Результаты работы программ представлены на изображениях (рис. 21-24). Исходный код выполненных программ также доступе но ссылка на GitHub:

1. <https://github.com/kchirkov/OOP/tree/master/pr71>
2. <https://github.com/kchirkov/OOP/tree/master/pr72>
3. <https://github.com/kchirkov/OOP/tree/master/pr73>
4. <https://github.com/kchirkov/OOP/tree/master/pr74>

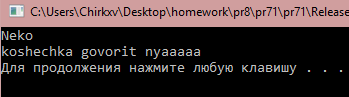


Рис. 21 Результат работы программы №1

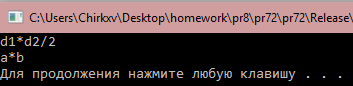


Рис. 22 Результат работы программы №2

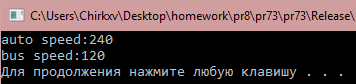


Рис. 23 Результат работы программы №3

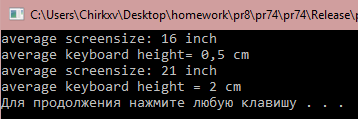


Рис. 24 Результат работы программы №4

## 

## 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**

1. Реализуйте программу, считывающую текст из файла и выводящую каждое слово с новой строки.
2. Реализуйте программу, считывающую текст с клавиатуры и записывающую его в файл.

**Ход работы**

1. Программа №1

#include "pch.h"

#include <iostream>

#include <fstream>

#include <string>

using namespace std;

int main()

{

ifstream fin("text.txt");

while (!fin.eof()) {

string s;

fin >> s;

cout << s << endl;

}

fin.close();

system("pause");

return 0;

}

2. Программа №2

#include "pch.h"

#include <iostream>

#include <fstream>

#include <string>

using namespace std;

int main()

{

setlocale(LC\_ALL, "rus");

string s;

cout << "Enter the text: " << endl;

getline(cin, s);

ofstream writetext("writetext.txt");

writetext << s;

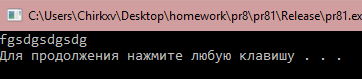
writetext.close();

}

**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 программ доступен по ссылкам на GitHub:

1. <https://github.com/kchirkov/OOP/tree/master/pr81>
2. <https://github.com/kchirkov/OOP/tree/master/pr82>



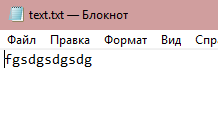
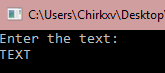


Рис. 25 Результат работы программы №1



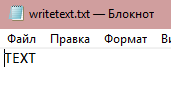


Рис. 26 Результат работы программы №2

## Лабораторная работа №1

**Создание многофайловых проектов**

**Цель практической работы**

Целью данной лабораторной работы освоить на практике создание многофайловых проектов на 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 находящую в них число с наибольшим модулем.

**Ход работы**

1. Программа №1

Файл заголовков complex.h:

#ifndef COMPLEX\_H

#define COMPLEX\_H

struct Complex {

double re, im;

void init(double r, double i) {

re = r;

im = i;

}

void add(Complex c);

void min(Complex c);

void umn(Complex c);

void del(Complex c);

};

#endif

Файл реализации класса Complex.cpp:

#include "pch.h"

#include "complex.h"

#include <cmath>

void Complex::add(Complex c)

{

re += c.re;

im += c.im;

}

void Complex::min(Complex c)

{

re -= c.re;

im -= c.im;

}

void Complex::umn(Complex c)

{

double t = re;

re = re \* c.re - im \* c.im;

im = t \* c.im + im \* c.re;

}

void Complex::del(Complex c)

{

double t = re;

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

im = (im\*c.re - t \* c.im) / (c.re\*c.re + c.im\*c.im);

}

Файл lab111.cpp

#include "pch.h"

#include <iostream>

#include "complex.h"

using namespace std;

int main()

{

Complex c1, c2;

char dei;

c1.init(1.0, 1.0);

c2.init(1.0, 1.0);

cout << "your numbers: " << endl << "c1 = 1 + 1i " << "c2 = 1 + 1i" << endl;

cout << "vvedite deystvie s/v/u/d" << endl;

cin >> dei;

if (dei == 's')

c1.add(c2);

if (dei == 'v')

c1.min(c2);

if (dei == 'u')

c1.umn(c2);

if (dei == 'd')

c1.del(c2);

cout << "real = " << c1.re << " "<< "im = " << c1.im << endl;

system("pause");

return 0;

}

2. Программа №2

#include "pch.h"

#include <iostream>

#include <cmath>

#include <fstream>

using namespace std;

struct Complex {

double re, im;

void init(double r, double i) {

re = r;

im = i;

}

double mod()

{

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

}

};

int main()

{

setlocale(LC\_ALL, "Russian");

int n;

double x;

double y;

ifstream amountfile("amount.txt");

amountfile >> n;

Complex \*p = new Complex[n];

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

{

amountfile >> x >> y;

p[i].init(x, y);

}

cout << n << endl;

amountfile.close();

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

{

cout << "p[" << i + 1 << "] = " << p[i].re << " + " << p[i].im << "\*i" << endl;

}

int mindex = 0;

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

{

if (p[i].mod() > p[mindex].mod())

mindex = i;

}

cout << "Индекс числа с максимальным модулем: " << mindex+1 << endl;

delete[] p;

system("pause");

return 0;

}

**Вывод**

Все поставленные задачи были выполнены. Для достижения требуемого результата был создан класс, разбитый на файлы с реализацией и объявлением. Результаты работы программ приведены на рисунках (рис. 27, 28). Исходный код также доступен по ссылкам на GitHub:

1. <https://github.com/kchirkov/OOP/tree/master/lab11>
2. <https://github.com/kchirkov/OOP/tree/master/lab12>

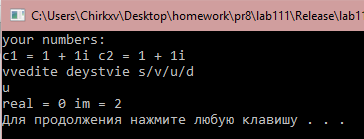
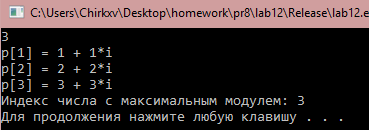


Рис. 27 Результат работы программы №1



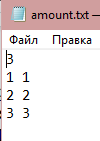


Рис. 28 Результат работы программы №2

## Лабораторная работа №2

**Указатели на функции**

**Цель практической работы**

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

**Задачи**

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 "pch.h"

#include <iostream>

using namespace std;

double oilspending1(double speed)

{

return speed \* 0.1;

}

double oilspending2(double speed)

{

return speed \* 0.15;

}

double(\*oilspending) (double speed);

int main()

{

double speed;

cout << "enter speed: " << endl;

cin >> speed;

if (speed <= 100)

{

oilspending = oilspending1;

}

else

{

oilspending = oilspending2;

}

cout << "oil spending per 100 km = " << oilspending(speed)<<endl;

}

1. Программа №2

#include "pch.h"

#include <iostream>

#include <ctime>

using namespace std;

void invert(int \*arr, int size) {

for (int i = 0; i < size / 2; i++) {

int c = arr[i];

arr[i] = arr[size - 1 - i];

arr[size - 1 - i] = c;

}

}

void sortvozr(int \*arr, int size) {

for (int i = 0; i < size - 1; i++) {

for (int j = i + 1; j < size; j++) {

if (arr[j] < arr[i]) {

int c = arr[i];

arr[i] = arr[j];

arr[j] = c;

}

}

}

}

void sortub(int \*arr, int size) {

for (int i = 0; i < size - 1; i++) {

for (int j = i + 1; j < size; j++) {

if (arr[j] > arr[i]) {

int c = arr[i];

arr[i] = arr[j];

arr[j] = c;

}

}

}

}

void(\*f(int \*arr, int size))(int \*arr, int size) {

int sum = 0;

for (int i = 1; i < size; i++) {

sum += arr[i];

}

if (sum == arr[0]) {

cout << "sum = pervomu elementu" << endl;

return invert;

}

else {

if (sum > arr[0]) {

cout << "sum > pervogo elementa" << endl;

return sortvozr;

}

else {

cout << "sum < pervogo elementa" << endl;

return sortub;

}

}

}

int main() {

srand(time(0));

int n = 10;

int \*arr = new int[n];

cout << "Array: ";

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

arr[i] = rand() % 100;

cout << arr[i] << " ";

}

cout << endl << endl;

f(arr, n)(arr, n);

cout << endl << "Changed array: ";

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

cout << arr[i] << " ";

}

cout << endl;

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

1. <https://github.com/kchirkov/OOP/tree/master/lab21>
2. <https://github.com/kchirkov/OOP/tree/master/lab22>

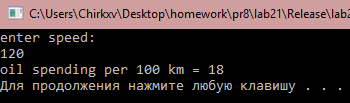


Рис. 29 Результат работы программы №1

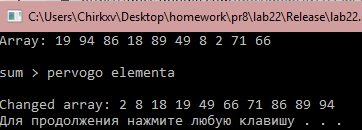


Рис. 30 Результат работы программы №2

## 

## Лабораторная работа №3

**Классы**

**Цель практической работы**

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

**Задачи**

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 рассчитать и вывести 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 Child {

string name;

string surname;

int age;

public:

void setName(string sname) { name = sname; }

void setSurname(string ssurname) { surname = ssurname; }

void setAge(int sage) { age = sage; }

void display() {

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

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

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

}

};

int main() {

string name, surname;

int age;

Child c1, c2;

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

cin >> name;

c1.setName(name);

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

cin >> surname;

c1.setSurname(surname);

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

cin >> age;

c1.setAge(age);

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

cin >> name;

c2.setName(name);

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

cin >> surname;

c2.setSurname(surname);

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

cin >> age;

c2.setAge(age);

c1.display();

c2.display();

system("pause");

return 0;

}

1. Программы №2

#include "pch.h"

#include <iostream>

#include <string>

using namespace std;

class Tiles {

public:

string brand;

double w;

double h;

double price;

void getData() {

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

cout << "Width: " << w << endl;

cout << "Height: " << h << endl;

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

}

};

int main() {

Tiles t1;

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

cin >> t1.brand;

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

cin >> t1.w;

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

cin >> t1.h;

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

cin >> t1.price;

Tiles t2;

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

cin >> t2.brand;

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

cin >> t2.w;

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

cin >> t2.h;

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

cin >> t2.price;

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

t1.getData();

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

t2.getData();

system("pause");

return 0;

}

1. Программа №3

#include "pch.h"

#include <iostream>

#include <cmath>

using namespace std;

class Complex {

double re;

double im;

public:

void setcomplex(double real, double im2) {

re = real;

im = im2;

}

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

void display() {

cout << "Number: " << re;

if (im < 0) {

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

}

else {

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

}

cout << "Module: " << getmodule() << endl;

}

};

int main() {

double re, im;

Complex c1, c2;

cout << "Enter re(1): ";

cin >> re;

cout << "Enter im(1): ";

cin >> im;

c1.setcomplex(re, im);

cout << "Enter re(2): ";

cin >> re;

cout << "Enter im(2): ";

cin >> im;

c2.setcomplex(re, im);

c1.display();

c2.display();

system("pause");

return 0;

}

1. Программа №4

#include "pch.h"

#include <iostream>

#include <cmath>

using namespace std;

class Vector {

double x;

double y;

double z;

public:

Vector(double sx, double sy, double sz) : x(sx), y(sy), z(sz) {};

double getmodule() { return sqrt(x \* x + y \* y + z \* z); }

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);

}

void display()

{

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

cout << "Module: " << getmodule() << endl;

}

};

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.display();

system("pause");

return 0;

}

**Вывод**

Все поставленные задачи были достигнуты. Для получения требуемого результата были использованы 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/kchirkov/OOP/tree/master/lab31>
2. <https://github.com/kchirkov/OOP/tree/master/lab32>
3. <https://github.com/kchirkov/OOP/tree/master/lab33>
4. <https://github.com/kchirkov/OOP/tree/master/lab34>

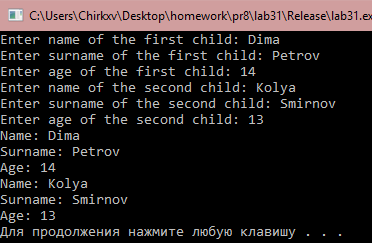


Рис. 31 Результат работы программы №1

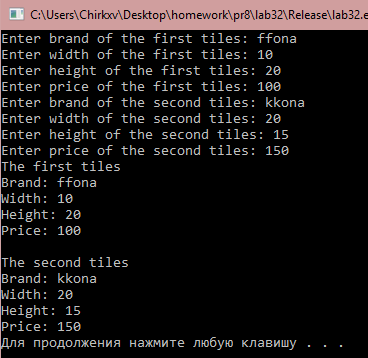


Рис. 32 Результат работы программы №2

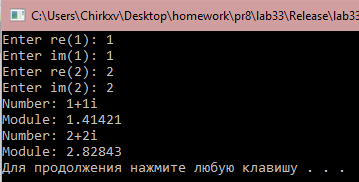


Рис. 33 Результат работы программы №3

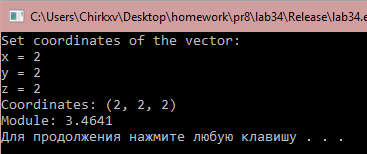


Рис. 34 Результат работы программы №4

## 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 "pch.h"

#include <iostream>

#include <string>

using namespace std;

class Child {

string name;

string surname;

int age;

public:

Child(string sname, string ssurname, int sage) : name(sname), surname(ssurname), age(sage) {}

Child(const Child& c) {

name = c.name;

surname = c.surname;

age = c.age;

}

Child operator= (const Child& c) {

return Child(c.name, c.surname, c.age);

}

void setname(string sname) { name = sname; }

void setsurname(string ssurname) { surname = ssurname; }

void setage(int sage) { age = sage; }

void display() {

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

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

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

}

~Child() {}

};

int main() {

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;

}

2. Программа №2

#include "pch.h"

#include <iostream>

#include <string>

using namespace std;

class Tiles {

string brand;

double wght;

double hght;

double price;

public:

Tiles(string sbrand, double swght, double shght, double sprice): brand(sbrand), wght(swght), hght(shght), price(sprice) {}

Tiles(const Tiles& t) {

brand = t.brand;

wght = t.wght;

hght = t.hght;

price = t.price;

}

Tiles operator=(const Tiles& t)

{

return Tiles(t.brand, t.wght, t.hght, t.price);

}

void setbrand(string sbrand) {

brand = sbrand;

}

void setwght(double swght) {

wght = swght;

}

void sethght(double shght) {

hght = shght;

}

void setprice(double sprice) {

price = sprice;

}

void getData() {

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

cout << "Width: " << wght << endl;

cout << "Height: " << hght << endl;

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

}

~Tiles() {}

};

int main() {

string brand;

double wght, hght, price;

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

cin >> brand;

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

cin >> wght;

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

cin >> hght;

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

cin >> price;

Tiles t1(brand, wght, hght, price);

Tiles t2=t1;

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

t1.getData();

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

t2.getData();

system("pause");

return 0;}

2. Программа №3

#include "pch.h"

#include <iostream>

#include <cmath>

using namespace std;

class Complex {

double re;

double im;

public:

Complex(double real, double im2) : re(real), im(im2) {}

Complex(const Complex& c) {

re = c.re;

im = c.im;

}

Complex operator=(const Complex& c)

{

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

}

void setcomplex(double real, double im2) {

re = real;

im = im2;

}

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

void display() {

cout << "Number: " << re;

if (im < 0) {

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

}

else {

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

}

cout << "Module: " << getmodule() << endl;

}

~Complex() {};

};

int main() {

double re, im;

cout << "Enter re(1): ";

cin >> re;

cout << "Enter im(1): ";

cin >> im;

Complex c1(re, im);

Complex c2 = c1;

c1.display();

c2.display();

system("pause");

return 0;

}

4. Программа №4

#include "pch.h"

#include <iostream>

#include <cmath>

using namespace std;

class Vector {

double x;

double y;

double z;

public:

Vector(double sx, double sy, double sz): x(sx), y(sy), z(sz) {}

Vector(const Vector& v) {

x = v.x;

y = v.y;

z = v.z;

}

void setvector(double sx, double sy, double sz) { x = sx; y = sy; z = sz; }

double getmodule() { return sqrt(x \* x + y \* y + z \* z); }

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);

}

void display()

{

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

cout << "Module: " << getmodule() << endl;

}

~Vector() {}

};

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;

Vector v3(v2);

v1.display();

v2.display();

v3.display();

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

1. <https://github.com/kchirkov/OOP/tree/master/lab41>
2. <https://github.com/kchirkov/OOP/tree/master/lab42>
3. <https://github.com/kchirkov/OOP/tree/master/lab43>
4. <https://github.com/kchirkov/OOP/tree/master/lab44>

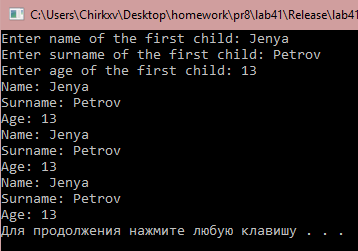


Рис. 35 Результат работы программы №1

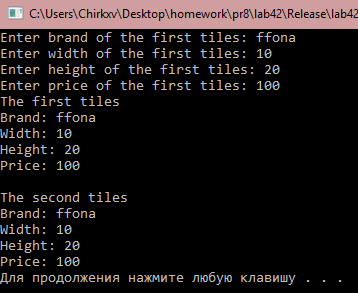


Рис. 36 Результат работы программы №2

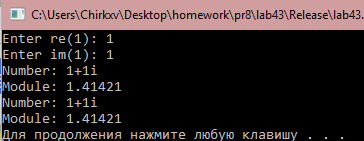


Рис. 37 Результат работы программы №3

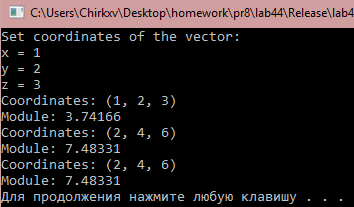


Рис 38. Результат работы программы 4

## Лабораторная работа №5

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

**Цель практической работы**

Целью данной лабораторной работы является изучение перегрузки операторов в языке С++ и использование перегруженных операторов 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 "pch.h"

#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)

{

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

}

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

Complex old(c);

c.re++;

return old;

}

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

Complex old(c);

c.re--;

return old;

}

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

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

}

friend 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;

}

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

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

return in;

}

~Complex() {}

};

int main() {

Complex c1;

cout << "Enter complex number re(space)im: ";

cin >> c1;

Complex c2 = c1;

if (c1 == c2) {

c2--;

c1++;

}

cout << c1;

cout << endl << c2;

cout << endl;

system("pause");

return 0;

}

1. Программа №2

#include "pch.h"

#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 sx, double sy, double sz) : x(sx), y(sy), z(sz) {};

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);

}

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

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

}

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

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

return in;

}

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

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

return out;

}

};

int main() {

Vector v1;

cout << "Set coordinates: ";

cin >> v1;

Vector v2 = v1;

if (v1 == v2) {

cout << v1 + v2;

}

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

1. <https://github.com/kchirkov/OOP/tree/master/lab51>
2. <https://github.com/kchirkov/OOP/tree/master/lab52>

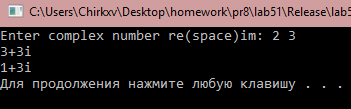


Рис. 39 Результат работы программы №1

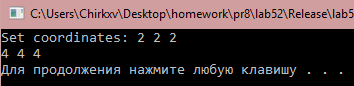


Рис. 40 Результат работы программы №2

## 

## Лабораторная работа №6

**Наследование**

**Цель практической работы**

Целью данной лабораторной работы является изучение наследования 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 "pch.h"

#include <iostream>

#include <string>

using namespace std;

class Student {

string name;

string surname;

int age;

public:

Student(string sname, string ssurname, int sage) : name(sname), surname(ssurname), age(sage) {};

string getname() { return name; }

string getsurname() { return surname; }

int getage() { return age; }

};

class Starosta : public Student {

string phone;

string vkpage;

public:

Starosta(string sname, string ssurname, int sage, string sphone, string svkpage) :

Student(sname, ssurname, sage), phone(sphone), vkpage(svkpage) {};

string getvkpage() { return vkpage; }

string getphone() { return phone; }

};

int main() {

string name, surname, phone, vkpage;

int age;

cout << "name: ";

cin >> name;

cout << "surname: ";

cin >> surname;

cout << "age: ";

cin >> age;

cout << "phone: ";

cin >> phone;

cout << "vkpage: ";

cin >> vkpage;

Starosta st(name, surname, age, phone, vkpage);

cout << endl << "Starosta info:" << endl;

cout << "Name: " << st.getname() << endl;

cout << "Surname: " << st.getsurname() << endl;

cout << "Age: " << st.getage() << endl;

cout << "Phone: " << st.getphone() << endl;

cout << "Vkpage: " << st.getvkpage() << endl;

system("pause");

return 0;

}

1. Программа №2

#include "pch.h"

#include <iostream>

#include <string>

using namespace std;

class Alive {

protected:

string name;

int age;

public:

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

string getname() { return name; }

int getage() { return age; }

virtual void habitat() = 0;

};

class Animal : public Alive {

public:

Animal(string sname, int sage) : Alive(sname, sage) {};

void habitat() { cout << "my habitat is surface of the Earth" << endl; }

};

class Fish : public Alive {

public:

Fish(string sname, int sage) : Alive(sname, sage) {};

void habitat() { cout << "my habitat is ocean" << endl; }

};

class Bird : public Alive {

public:

Bird(string sname, int sage) : Alive(sname, sage) {};

void habitat() { cout << "haha i'm flying lOOsers" << endl; }

};

int main() {

string name;

int age;

cout << endl << "Animal data" << endl;

cout << "name: ";

cin >> name;

cout << "age: ";

cin >> age;

Animal a(name, age);

cout << "Fish data" << endl;

cout << "name: ";

cin >> name;

cout << "age: ";

cin >> age;

Fish f(name, age);

cout << endl << "Bird data" << endl;

cout << "Enter name: ";

cin >> name;

cout << "Enter age: ";

cin >> age;

Bird b(name, age);

cout << endl << "Animal" << endl << "Name: " << a.getname() << endl << "Age: " <<

a.getage() << endl;

a.habitat();

cout << endl << "Bird" << endl << "Name: " << b.getname() << endl << "Age: " <<

b.getage() << endl;

b.habitat();

cout << endl << "Fish" << endl << "Name: " << f.getname() << endl << "Age: " <<

f.getage() << endl;

f.habitat();

system("pause");

return 0;

}

1. Программа №3

#include "pch.h"

#include <iostream>

#include <string>

using namespace std;

class Animal {

protected:

string name;

int age;

public:

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

string getname() { return name; }

int getage() { return age; }

virtual void saying() = 0;

};

class Dog : public Animal {

public:

Dog(string sname, int sage) : Animal(sname, sage) {};

void saying() { cout << "wooof! woof!" << endl; }

};

class Cat : public Animal {

public:

Cat(string sname, int sage) : Animal(sname, sage) {};

void saying() { cout << "Nyaaaaaaa" << endl; }

};

int main() {

string name;

int age;

cout << "Cat name" << endl;

cin >> name;

cout << "Cat age: "<<endl;

cin >> age;

Cat c(name, age);

cout << "Dog name" << endl;

cin >> name;

cout << "Dog age: ";

cin >> age;

Dog d(name, age);

cout << "Dog Name: " << d.getname() << endl << "Dog Age: " <<

d.getage() << endl;

d.saying();

cout << "Cat Name: " << c.getname() << endl << "Cat Age: " <<

c.getage() << endl;

c.saying();

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/kchirkov/OOP/tree/master/lab61>
2. <https://github.com/kchirkov/OOP/tree/master/lab62>
3. <https://github.com/kchirkov/OOP/tree/master/lab63>

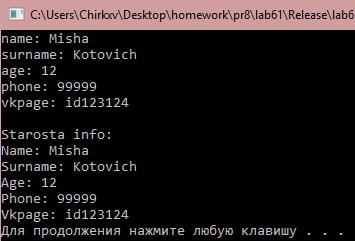


Рис. 41 Результат работы программы №1

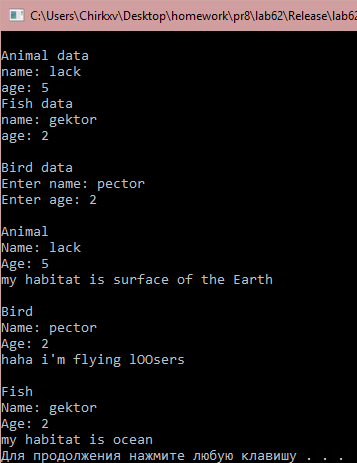


Рис. 42 Результат работы программы №2

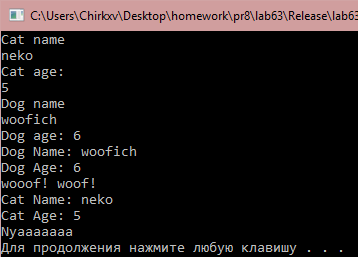


Рис. 43 Результат работы программы №3

## 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 "pch.h"

#include <iostream>

#include <string>

using namespace std;

class Human {

protected:

string name;

string surname;

int age;

public:

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

Human(string name, string surname, int age) : name(name), surname(surname), age(age) {}

virtual void print() = 0;

~Human() {}

};

class Student: public Human {

bool on\_lesson;

public:

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

Student(string name, string surname, int age, bool on\_lesson) : Human(name, surname, age), on\_lesson(on\_lesson) {}

void print() {

cout << "Name of student: " << name << surname << endl << "Age of student: " << age << endl << "On lesson or not: " << on\_lesson << endl;

}

~Student() {}

};

class Boss : public Human{

int number\_of\_workers;

public:

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

Boss(string name, string surname, int age, int number\_of\_workers) : Human(name, surname, age), number\_of\_workers(number\_of\_workers) {}

void print() {

cout << "Name of boss: " << name << surname << endl << "Age of boss: " << age << endl << "Number of workers of boss: " << number\_of\_workers << endl;

}

~Boss() {}

};

int main()

{

string name,surname;

int age, nof;

bool onlesson;

cout << "Enter Name of Student: " << endl;

cin >> name;

cout << "Enter Surname of Student: " << endl;

cin >> surname;

cout << "Enter Age of Student: " << endl;

cin >> age;

cout << "is student on lesson or not?(1/0)" << endl;

cin >> onlesson;

Student s(name, surname, age, onlesson);

cout << "Enter Name of Boss: " << endl;

cin >> name;

cout << "Enter Surname of Boss: " << endl;

cin >> surname;

cout << "Enter Age of Boss: " << endl;

cin >> age;

cout << "Enter number of workers: " << endl;

cin >> nof;

Boss b(name, surname, age, nof);

s.print();

b.print();

system("pause");

return 0;

}

**Вывод**

Все поставленные задачи были выполнены. При выполнении 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/kchirkov/OOP/tree/master/lab7>

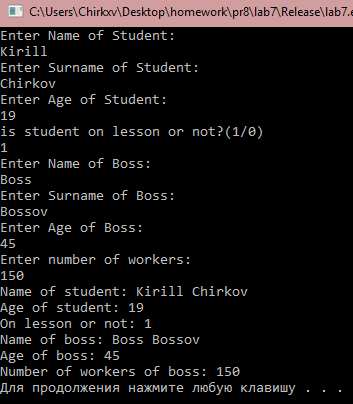


Рис. 44 Результат работы программы

## 

## 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 "pch.h"

#include <iostream>

#include <string>

#include <vector>

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() {

vector<Student> students;

int n;

cout << "Enter number of students:" << endl;

cin >> n;

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

string name, surname;

int age;

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

cout << "name: ";

cin >> name;

cout << "surname: ";

cin >> surname;

cout << "age: ";

cin >> age;

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

}

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

cout << endl << "name: " << students[i].getname() << endl;

cout << "surname: " << students[i].getsurname() << endl;

cout << "age: " << students[i].getage() << endl;

}

system("pause");

return 0;

}

1. Программа №2

#include "pch.h"

#include <iostream>

#include <cmath>

#include <list>

using namespace std;

class Complex {

double re;

double im;

public:

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

void setnumber(double sre, double sim) {

re = sre;

im = sim;

}

void display() {

cout << "Complex number: " << re;

if (im < 0) {

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

}

else {

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

}

}

~Complex() {}

};

int main() {

list<Complex> c;

int n;

cout << "Enter number of complex numbers:" << endl;

cin >> n;

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

double re, im;

cout << "Complex number " << i + 1 << endl;

cout << "Enter Re: "<<endl;

cin >> re;

cout << "Enter Im: "<<endl;

cin >> im;

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

}

for (auto a:c) {

a.display();

}

system("pause");

return 0;

}

1. Программа №3

#include "pch.h"

#include <iostream>

#include <string>

#include <queue>

using namespace std;

class automobile {

string model;

string number;

public:

automobile(string model, string number) : model(model), number(number) {};

void display() {

cout << "Model of car: " << model << endl << "Number: " << number << endl;

}

};

int main() {

queue<automobile> a;

int n;

cout << "Enter number of cars: " << endl;

cin >> n;

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

string model, number;

cout << "Auto " << i + 1 << endl;

cout << "Enter model of car: ";

cin >> model;

cout << "Enter number: ";

cin >> number;

a.push(automobile(model, number));

}

cout << "Queue" << endl;

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

cout << i<<" car in the queue: "<<endl;

a.front().display();

a.pop();

}

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

1. <https://github.com/kchirkov/OOP/tree/master/lab81>
2. <https://github.com/kchirkov/OOP/tree/master/lab82>
3. <https://github.com/kchirkov/OOP/tree/master/lab83>

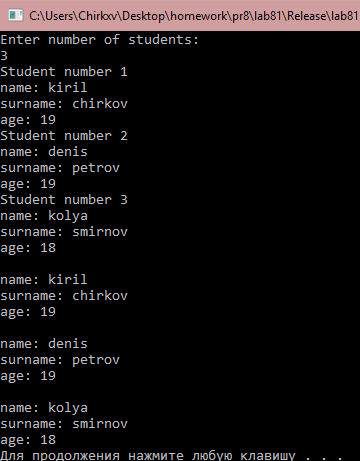


Рис. 45 Результат работы программы №1

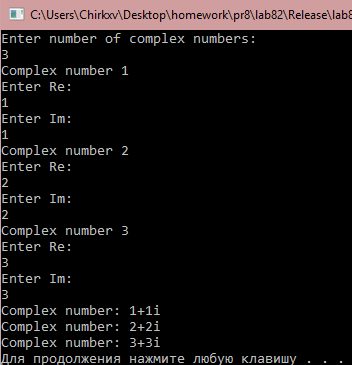


Рис. 46 Результат работы программы №2

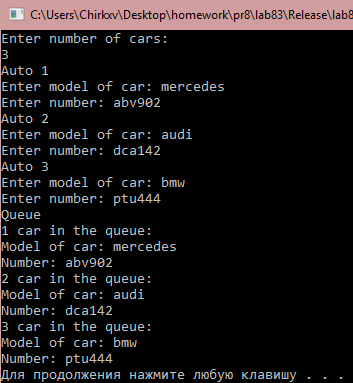


Рис. 47 Результат работы программы №3