|  |  |
| --- | --- |
| 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 | Ватулин Д.С. |
| Принял ассистент кафедры | Хлебникова В.Л. |

|  |  |  |
| --- | --- | --- |
| Лабораторные работы выполнены | «28» апреля 2019 г. |  |
|  |  |  |
| «Зачтено» | «\_\_»\_\_\_\_\_\_\_201\_\_ г. |  |

Москва 2019

## 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 |  |

## Лабораторная работа №1

**Создание многофайловых проектов**

**Цель практической работы**

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

**Задачи**

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

**Ход работы**

1. Программа №1

#include <iostream>

#include <cmath>

#include <complex>

using namespace std;

int main()

{

cout << "Enter the first complex number"<<endl;

complex<double> a;

cin >> a;

cout << "Enter the second complex number "<< endl;

complex<double> b;

cin >> b;

cout << "What you need" << endl;

cout << "1 - Square"<< endl;

cout << "2 - Multiply the first number by a constant "<< endl;

cout << "3 - 1\*2" << endl;

cout << "4 - Subtract the second from the first number" << endl;

cout << "5 - Add the first number to the second" << endl;

int x;

cin >> x;

complex<double> c;

const double n = 2.5;

switch (x)

{

case 1:

c = a \* a;

cout << c;

break;

case 2:

c = a \* n;

cout << c;

break;

case 3:

c = b \* a;

cout << c;

break;

{

}

case 4:

c = a - b;

cout << c;

break;

case 5:

c = a + b;

cout << c;

break;

}

return 0;

}

Файл заголовков Complex.h:

#pragma once

#include<iostream>

class Complex

{

private:

double real;

double imag;

public:

Complex();

Complex(double, double);

void setComplex(double real,double imag);

void getComplex();

double getModule();

Complex& operator=(const Complex&);

friend const Complex operator+(const Complex&,const Complex&);

friend const Complex operator-(const Complex&, const Complex&);

friend const Complex operator\*(const Complex&, const Complex&);

friend const Complex operator/(const Complex&, const Complex&);

~Complex();

};

Файл реализации класса Complex.cpp:

#include "Complex.h"

Complex::Complex()

{

this->real = 0;

this->imag = 0;

}

Complex::Complex(double real, double imag)

{

this->real = real;

this->imag = imag;

}

void Complex::setComplex(double real,double imag) {

this->real = real;

this->image = imag;

}

void Complex::getComplex() {

std::cout << real << " + (" << imag << "i)\n";

}

double Complex::getModule() {

return(sqrt(real\*real + imag \* imag));

}

Complex& Complex::operator=(const Complex& x) {

re = x.real;

im = x.imag;

return \*this;

}

const Complex operator+(const Complex& x1, const Complex& x2) {

return Complex(x1.real + x2.real, x1.imag + x2.imag);

}

const Complex operator-(const Complex& x1, const Complex& x2) {

return Complex(x1.real - x2.real, x1.imag - x2.imag);

}

const Complex operator\*(const Complex& x1, const Complex& x2) {

return Complex(x1.real\*x2.real - x1.imag\*x2.imag, x1.real\*x2.imag + x2.real\*x1.imag);

}

const Complex operator/(const Complex& x1, const Complex& x2) {

return Complex((x1.real\*x2.real + x1.imag\*x2.imag) / (x2.real\*x2.real + x2.imag\*x2.imag), (x1.real\*x2.imag - x1.imag\*x2.real) / (x2.real\*x2.real + x2.imag\*x2.imag));

}

Complex::~Complex()

{

}

1. Программа №2

#include"Complex.h"

#include<iostream>

#include<fstream>

using namespace std;

int main()

{

ifstream filex("complex.txt");

double max = -1;

Complex c;

int n;

filex >> n;

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

Complex c1;

double real, imag;

filex >> real >> imag;

filex.get();

c1.setComplex(real, imag);

if (c1.getModule() > max)

c = c1;

max = c.getModule();

}

cout << "MaxModule = " << max << endl << "It's module of complex =";

c.getComplex();

filex.close();

system("pause");

}

**Вывод**

Для достижения требуемого результата был создан класс, разбитый на файлы с реализацией и объявлением..

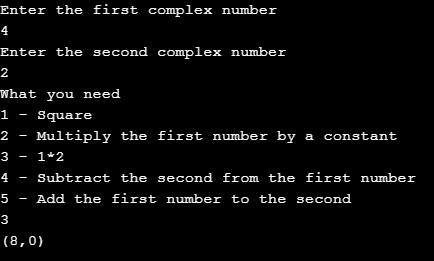
****

Рис. 1 Результат работы первой программы



Рис. 2 Результат работы второй программы

## Лабораторная работа №2

**Указатели на функции**

**Цель практической работы**

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

**Задачи**

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

**Ход работы**

1. Программа №1

#include <iostream>

using namespace std;

int summ(int a, int b) {

return a + b;

}

int multiplication(int a, int b) {

return a \* b;

}

int difference(int a, int b) {

return a - b;

}

int(\*d)(int a, int b);

int main()

{

int a, b;

cout << "enter a:" << endl;

cin >> a;

cout << "enter b:" << endl;

cin >> b;

int c;

cout << "1-difference 2-summ 3-multiplication:" << endl;

cin >> c;

switch (c)

{

case 1: d = difference;

break;

case 2: d = summ;

break;

case 3: d = multiplication;

break;

default:

break;

}

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

system("pause");

return 0;

}

1. Программа №2

#include <iostream>

#include<ctime>

#include<string>

using namespace std;

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

{

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

}

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

{

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

}

void d1(int \*a, int c) { for (int i = c - 1; i > -1; i--)cout << a[i]; cout << endl; }

void d2(int \*a, int c) { qsort(a, c, sizeof(int), compare); };

void d3(int \*a, int c) { qsort(a, c, sizeof(int), comp2); };

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

int sum=0;

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

if (sum == a[0])return d1;

else if (sum > a[0])return d2;

else return d3;

}

int main()

{

srand(time(0));

int c;

cin >> c;

int \*a= new int[c];

a[0] = 1000;

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

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

cout << a[i] << endl;

}

func(a, c)(a, c);

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

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

}

cout << endl;

delete[] a;

system("pause");

return 0;

}

**Вывод**

Для достижения поставленных целей были использованы указатели на функции с различными возвращаемыми значениями.

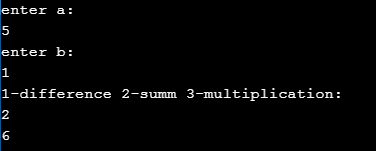


Рис. 3 Результат работы первой программы

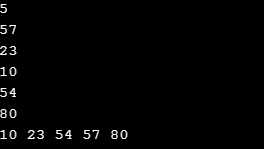


Рис. 4 Результат работы второй программы

## 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 <iostream>

#include <string>

using namespace std;

class Child {

private:

string name;

string family;

int age;

public:

void set\_name(string name) {

this->name = name;

};

void set\_family(string family) {

this->family = family;

};

void set\_age(int age) {

this->age = age;

};

void print\_value() {

cout << "Имя: " << name << endl;

cout << "Фамилия: " << family << endl;

cout << "Возраст: " << age << endl;

};

};

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

{

Child ch1,ch2;

ch1.set\_name((string)"Дмитрий");

ch1.set\_family((string)"Ватулин");

ch1.set\_age(18);

ch2.set\_name((string)"Василий");

ch2.set\_family((string)"Иванов");

ch2.set\_age(18);

ch1.print\_value();

ch2.print\_value();

return 0;

}

1. Программы №2

#include <iostream>

#include <string>

using namespace std;

class Tiles {

public:

string brand;

int size\_h;

int size\_w;

int price;

void getData() {

cout << "Brand: " << '\t' << brand << endl;

cout << "Length: " << '\t' << size\_h << endl;

cout << "Width " << '\t' << size\_w << endl;

cout << "Price: " << '\t' << price << endl;

};

};

int main() {

Tiles t1,t2;

t1.brand = "Дуб";

t1.size\_h = 12;

t1.size\_w = 10;

t1.price = 200;

t2.brand = "Береза";

t2.size\_h = 40;

t2.size\_w = 25;

t2.price = 350;

t1.getData();

t2.getData();

return 0;

}

1. Программа №3

#include <iostream>

#include <string>

#include <cmath>

using namespace std;

class Complex

{

double Re, Im;

public:

void set\_c()

{

cout << "Enter valid part: " << endl;

cin >> Re;

cout << "Enter the imaginary part: " << endl;

cin >> Im;

}

string get\_c()

{

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

}

double modul()

{

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

}

double arg()

{

return atan(Im / Re);

}

};

int main()

{

Complex a;

a.set\_c();

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

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

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

system("pause");

return 0;

}

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);

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

void printData();

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**

Для получения требуемого результата были использованы конструкции класса и перегрузка некоторых операторов.

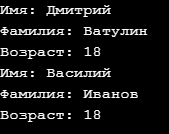


Рис. 5 Результат работы первой программы



Рис. 6 Результат работы второй программы

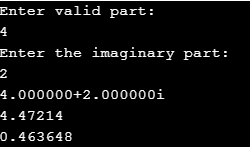


Рис. 7 Результат работы третьей программы

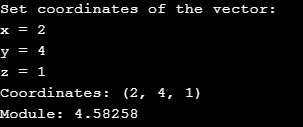


Рис. 8 Результат работы четвертой программы

## 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 <cmath>

using namespace std;

class Complex {

private:

float real;

float imag;

public:

Complex(float a, float b) {

real = a;

imag = b;

};

float get\_abs() {

return sqrt(pow(real,2)+pow(imag,2));

};

float getData() {

cout << "Complex number: "<< endl;

cout << "Real part: " << real << endl;

cout << "Imaginary part: " << imag << endl;

cout << "The absolute value of a number: " << this->get\_abs() << endl;

};

};

int main() {

Complex c1(5,2.2),c2(2,4.5);

c1.getData();

c2.getData();

return 0;

}

1. Программа №2

#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);

}

1. Программа №3

#include <iostream>

#include <string>

using namespace std;

class Tiles {

public:

string brand;

int size\_h;

int size\_w;

int price;

Tiles(string b, int h, int w, int p) {

brand = b;

size\_h = h;

size\_w = w;

price = p;

};

void getData() {

cout << "Brand: " << '\t' << brand << endl;

cout << "Length: " << '\t' << size\_h << endl;

cout << "Width " << '\t' << size\_w << endl;

cout << "Price: " << '\t' << price << endl;

};

};

int main() {

Tiles t1((string)"Дуб", 12,10,200),t2((string)"Береза",40,25,350);

t1.getData();

t2.getData();

return 0;

}

1. Программа №4

#include <iostream>

#include <string>

using namespace std;

class Child {

private:

string name;

string family;

int age;

public:

int some;

Child(string name, string family, int age) {

this->age = age;

this->name = name;

this->family = family;

};

void print\_value() {

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

cout << "Surname: " << family << endl;

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

};

};

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

{

Child ch1((string)"Дмитрий", (string)"Ватулин", 18),ch2((string)"Василий", (string)"Иванов", 18);

ch1.print\_value();

ch2.print\_value();

return 0;

}

**Вывод**

В процессе работы были более подробно изучены разные виды конструкторов и деструкторы. Для выполнения работы были использованы простые конструкторы и конструкторы копирования.

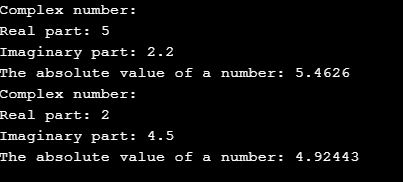


Рис. 9 Результат работы первой программы

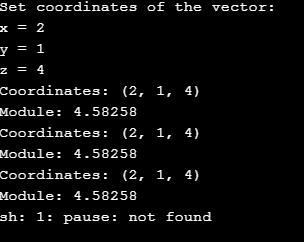


Рис. 10 Результат работы второй программы

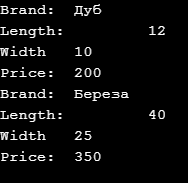


Рис. 11 Результат работы третьей программы

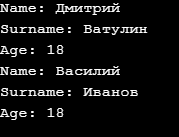


Рис. 12 Результат работы четвертой программы

## 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 операторыприсваивания, инкремента, декремента, сравнения, ввода ивывода.
2. Для класса Vector перегрузить операторы присваивания,сравнения, ввода и вывода.

**Ход работы**

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;

}

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 + v2;

}

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**

В процессе выполнения работы был более плотно изучен механизм перегрузки. Для выполнения поставленных задач была использована перегрузка некоторых бинарных и унарных операторов.

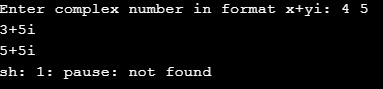


Рис. 13 Результат работы первой программы

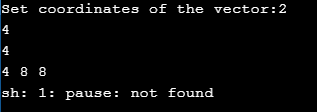


Рис. 14 Результат работы второй программы

## Лабораторная работа №6

**Наследование**

**Цель практической работы**

Целью данной лабораторной работы является изучение наследованния 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>

using namespace std;

class Student {

string name;

string surname;

int age;

string group;

public:

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

string getName() {

return name; }

string getSurname() {

return surname; }

int getAge() {

return age; }

string getGroup() {

return group; }

};

class Headman : public Student {

string phone;

string mail;

public:

Headman(string name, string surname, int age, string group, string phone, string mail) :

Student(name, surname, age, group), phone(phone), mail(mail) {};

string getMail() {

return mail; }

string getPhone() {

return phone; }

};

int main() {

string name, surname, group, phone, mail;

int age;

cout << "Enter name: ";

cin >> name;

cout << "Enter surname: ";

cin >> surname;

cout << "Enter age: ";

cin >> age;

cout << "Enter group";

cin >> group;

cout << "Enter phone: ";

cin >> phone;

cout << "Enter mail: ";

cin >> mail;

Headman h(name, surname, age, group, phone, mail);

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

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

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

cout << "Group: " << h.getGroup() << endl;

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

cout << "Mail: " << h.getMail() << endl;

cout << endl;

system("pause");

return 0;

}

1. Программа №2

#include <iostream>

#include <string>

using namespace std;

class Alive {

protected:

string habitat;

int age;

string name;

public:

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

string getName() {

return name;

}

int getAge() {

return age;

}

string getHabitat() {

return habitat;

}

virtual void move() = 0;

};

class Fish : public Alive {

public :

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

void move()

{

cout << "fish " << "swim" << endl;

}

};

class Animal : public Alive {

public :

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

void move()

{

cout << "animal " << "run" << endl;

}

};

class Bird : public Alive {

public :

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

void move()

{

cout << "bird " << "fly" << endl;

}

};

int main() {

int age;

string name;

string habitat;

cout << "Animal" << endl;

cout << "enter name: ";

cin >> name;

cout << "enter age: " ;

cin >> age;

cout << "enter habitat: ";

cin >> habitat;

Animal a(age, name, habitat);

cout << "Bird" << endl;

cout << "enter name: ";

cin >> name;

cout << "enter age: " ;

cin >> age;

cout << "enter habitat: ";

cin >> habitat;

Bird b(age, name, habitat);

cout << "Fish" << endl;

cout << "enter name: ";

cin >> name;

cout << "enter age: " ;

cin >> age;

cout << "enter habitat: ";

cin >> habitat;

Fish c(age, name, habitat);

cout << "Animal" << endl << "name: " << a.getName() << endl << "age: " << a.getAge() << endl << "habitat: " << a.getHabitat() <<endl;

a.move();

cout << "Bird" << endl << "name: " << b.getName() << endl << "age: " << b.getAge() << endl << "habitat: " << a.getHabitat() <<endl;

b.move();

cout << "Fish" << endl << "name: " << c.getName() << endl << "age: " << c.getAge() << endl << "habitat: " << a.getHabitat() <<endl;

c.move();

return 0;

};

1. Программа №3

#include <iostream>

#include <string>

using namespace std;

class Animal {

protected:

int size;

string name;

int weight;

virtual void make\_sound() = 0;

public:

Animal(int size, int weight, string name): size(size), weight(weight), name(name) {

};

int getSize() {

return size;

}

int getWeight() {

return weight;

}

string getName() {

return name;

}

};

class Cat : public Animal {

public:

Cat( int size, int weight, string name) : Animal(size, weight, name) {

};

void make\_sound() {

cout << "Meow" << endl;

}

};

class Dog : public Animal {

public:

Dog( int size, int weight, string name) : Animal(size, weight, name) {

};

void make\_sound() {

cout << "Woof" << endl;

}

};

int main (){

int size;

int weight;

string name;

cout << "Cat" << endl;

cout << "Enter name: ";

cin >> name;

cout << "Enter size: ";

cin >> size;

cout << "Enter weight: ";

cin >> weight;

Cat a(size, weight, name);

cout << "Dog" << endl;

cout << "Enter name: ";

cin >> name;

cout << "Enter size: ";

cin >> size;

cout << "Enter weight: ";

cin >> weight;

Dog b(size, weight, name);

cout << "Cat" << endl << "name: " << a.getName() << endl << "size: " << a.getSize() << endl << "weight: " << a.getWeight() <<endl;

a.make\_sound();

cout << "Dog" << endl << "name: " << b.getName() << endl << "size: " << b.getSize() << endl << "weight: " << b.getWeight() <<endl;

b.make\_sound();

return 0;

}

**Вывод**

В процессе выполнения работы было изучено и использовано наследование.



Рис. 15 Результат работы первой программы

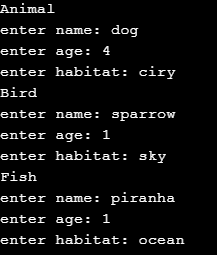
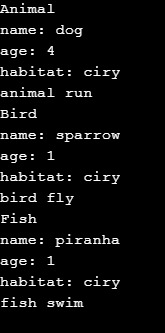
 

Рис. 16 Результат работы второй программы

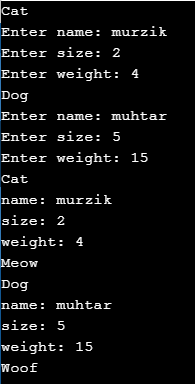


Рис. 17 Результат работы третьей программы

## 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 UMLдиаграмме.

**Ход работы**

#include <iostream>

#include <string>

using namespace std;

class Human {

protected:

string name;

string surname;

string midname;

int age;

public:

Human() {

name = " ";

surname = " ";

midname = " ";

age = 0;

};

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

};

virtual void print() = 0;

virtual ~Human() {

};

};

class Student : public Human {

private:

bool on\_lesson;

public:

Student() : Human() {

on\_lesson = true;

};

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

};

void print() {

cout << "Student" << endl;

cout << "Enter surname: ";

cin >> surname;

cout << "Enter name: ";

cin >> name;

cout << "Enter midname: ";

cin >> midname;

cout << "Enter age: ";

cin >> age;

cout << ((on\_lesson) ? "Yes" : "No") << endl;

cin >> on\_lesson;

};

~Student() {};

};

class Boss: public Human {

private:

int number\_of\_workers;

protected:

Boss() : Human() {

number\_of\_workers = 0;

}

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

};

void print() {

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

cout << "Enter surname: ";

cin >> surname;

cout << "Enter name: ";

cin >> name;

cout << "Enter midname: ";

cin >> midname;

cout << "Enter age: ";

cin >> age;

cout << "Enter number of workers: ";

cin >> number\_of\_workers;

};

~Boss() { };

};

**Вывод**

Для создания программ были использованы механизмы наследования, переопределения функций и абстрактные классы.

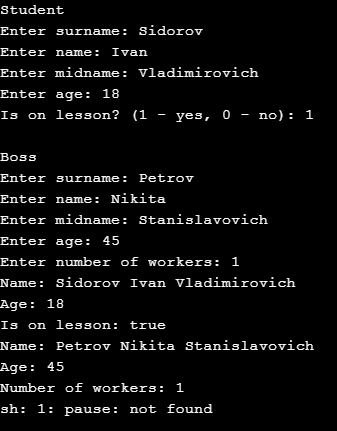


Рис. 18 Результат работы программы

## Лабораторная работа №8

**Бибилиотека STL**

**Цель практической работы**

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

**Задачи**

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

**Ход работы**

1. Программа №1

#include <iostream>

#include <string>

#include <vector>

using namespace std;

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> vec;

string name;

string surname;

int age;

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

cout << "Student " << endl;

cout << "Enter name: ";

cin >> name;

cout << "Enter surname: ";

cin >> surname;

cout << "Enter age: ";

cin >> age;

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

Student a(name, surname, age);

cout << "Name: " << a.getName() << endl << "Surname: " << a.getSurname() << endl << "Age: " << a.getAge() << endl;

}

return 0;

}

1. Программа №2

#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) {}

Complex(const Complex& c) {

re = c.re;

im = c.re;

}

Complex operator= (const Complex& c) {

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

}

void setNumber(double re, double im) {

this->re = re;

this->im = im;

}

double getModule() {

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

}

void displayData() {

cout << "Number: " << re;

if (im < 0) {

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

} else {

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

}

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

}

~Complex() {}

};

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

}

system("pause");

return 0;

}

void Client::getData() {

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

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

}

**Вывод**

Для создания программы были использованы коллекции queue, string, list и vector.

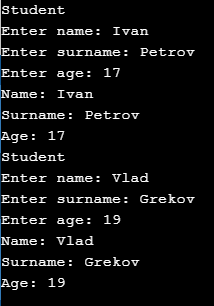


Рис. 19 Результат работы первой программы

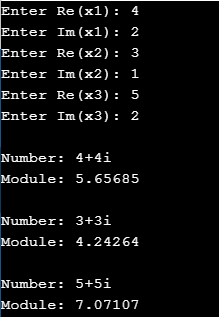


Рис. 20 Результат работы второй программы

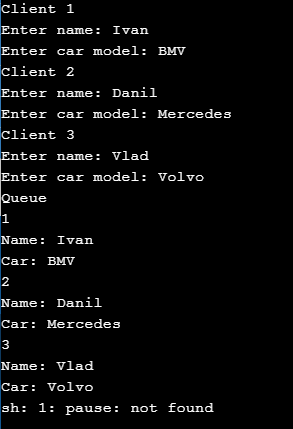


Рис. 21 Результат работы третьей программы