|  |  |
| --- | --- |
| 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 | Дмитриев А.Р. |
| Принял ассистент кафедры | Хлебникова В.Л. |

|  |  |  |
| --- | --- | --- |
| Лабораторные работы выполнены | «25» апреля 2019 г. |  |
|  |  |  |
| «Зачтено» | «\_\_»\_\_\_\_\_\_\_201\_\_ г. |  |

Москва 2019

**Таблица успеваемости**

**Лабораторные занятия**

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

**Лабораторная работа №1**

**Создание многофайловых проектов**

**Цель практической работы**

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

**Задачи**

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

**Ход работы**

Программа 1  
Файл Complex.h  
#pragma once

#include<iostream>

class Complex

{

private:

double re;

double im;

public:

Complex();//Конструкторы

Complex(double, double);

void setComplex(double re, double im);//Задать значения re,im.

void getComplex();//Вывод комплексного числа

double getModule();//Получить значение модуля комплексного числа

Complex& operator=(const Complex&);//Перегруженный оператор присваивания

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

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

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

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

~Complex();//Деструктор

};

Файл Complex.cpp  
  
#include "Complex.h"

Complex::Complex()

{

this->re = 0;

this->im = 0;

}

Complex::Complex(double re, double im)

{

this->re = re;

this->im = im;

}

void Complex::setComplex(double re, double im) {

this->re = re;

this->im = im;

}

void Complex::getComplex() {

std::cout << re << " + (" << im << "i)\n";

}

double Complex::getModule() {

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

}

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

re = x.re;

im = x.im;

return \*this;

}

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

return Complex(x1.re + x2.re, x1.im + x2.im);

}

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

return Complex(x1.re - x2.re, x1.im - x2.im);

}

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

return Complex(x1.re\*x2.re - x1.im\*x2.im, x1.re\*x2.im + x2.re\*x1.im);

}

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

return Complex((x1.re\*x2.re + x1.im\*x2.im) / (x2.re\*x2.re + x2.im\*x2.im), (x1.re\*x2.im - x1.im\*x2.re) / (x2.re\*x2.re + x2.im\*x2.im));

}

Complex::~Complex()

{

}  
  
Файл Source.cpp  
#include"Complex.h"

#include<iostream>

using namespace std;

int main()

{

double re, im;

cout << "Enter Re and Im for Complex1:";

cin >> re >> im;

Complex c1(re, im);

cout << "Enter Re and Im for Complex2:";

cin >> re >> im;

Complex c2(re, im);

cout << "Complex1 = ";

c1.getComplex();

cout << "Module = " << c1.getModule() << endl << endl;

cout << "Complex2 = ";

c2.getComplex();

cout << "Module = " << c2.getModule() << endl << endl;

cout << "Complex1 + Complex2 = ";

Complex c3 = c1 + c2;

c3.getComplex();

cout << "Module = " << c3.getModule() << endl << endl;

cout << "Complex1 - Complex2 = ";

c3 = c1 - c2;

c3.getComplex();

cout << "Module = " << c3.getModule() << endl << endl;

cout << "Complex1 \* Complex2 = ";

c3 = c1 \* c2;

c3.getComplex();

cout << "Module = " << c3.getModule() << endl << endl;

cout << "Complex1 / Complex2 = ";

c3 = c1 / c2;

c3.getComplex();

cout << "Module = " << c3.getModule() << endl << endl;

system("pause");

}

Программа 2  
Файл Complex.h  
#pragma once

#include<iostream>

class Complex

{

private:

double re;

double im;

public:

Complex();//Конструкторы

Complex(double, double);

void setComplex(double re, double im);//Задать значения re,im.

void getComplex();//Вывод комплексного числа

double getModule();//Получить значение модуля комплексного числа

Complex& operator=(const Complex&);//Перегруженный оператор присваивания

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

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

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

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

~Complex();//Деструктор

};

Файл Complex.cpp  
  
#include "Complex.h"

Complex::Complex()

{

this->re = 0;

this->im = 0;

}

Complex::Complex(double re, double im)

{

this->re = re;

this->im = im;

}

void Complex::setComplex(double re, double im) {

this->re = re;

this->im = im;

}

void Complex::getComplex() {

std::cout << re << " + (" << im << "i)\n";

}

double Complex::getModule() {

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

}

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

re = x.re;

im = x.im;

return \*this;

}

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

return Complex(x1.re + x2.re, x1.im + x2.im);

}

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

return Complex(x1.re - x2.re, x1.im - x2.im);

}

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

return Complex(x1.re\*x2.re - x1.im\*x2.im, x1.re\*x2.im + x2.re\*x1.im);

}

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

return Complex((x1.re\*x2.re + x1.im\*x2.im) / (x2.re\*x2.re + x2.im\*x2.im), (x1.re\*x2.im - x1.im\*x2.re) / (x2.re\*x2.re + x2.im\*x2.im));

}

Complex::~Complex()

{

}  
  
Файл Source.cpp  
#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 re, im;

filex >> re >> im;

filex.get();

c1.setComplex(re, im);

if (c1.getModule() > max)

c = c1;

max = c.getModule();

}

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

c.getComplex();

filex.close();

system("pause");

}  
  
complex.txt:

6

1 + 3i

3 + 5i

6 - 10i

10 - 20i

5 + 8i

1 + 1i

**Вывод**

Для достижения требуемого результата был создан класс, разбитый на файлы с реализацией и объявлением. Также были перегружены некоторые операторы и использована библиотека fstream. Результат работы программ показан ниже (Рис.1 и 2). Исходный код программ также доступен на GitHub по ссылке:<https://github.com/WarerBit/OOP/tree/master/Lab1>

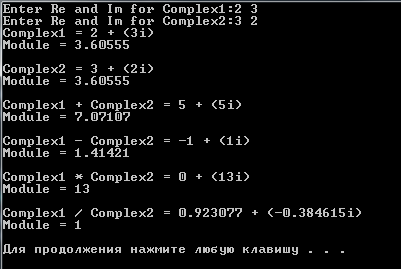


Рис. 1 Результат работы программы 1



Рис. 2 Результат работы программы 2

**Лабораторная работа №2**

**Указатели на функции**

**Цель практической работы**

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

**Задачи**

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

**Ход работы**

Программа 1  
#include<iostream>

int sum(int x1, int x2, int x3);//Складывает 3 числа

int dif(int x1, int x2, int x3);//Вычитает из первого числа второе и третье

int sumdif(int x1, int x2, int x3);//Прибавляет к первому числу второе и вычитает третье

int(\*fun)(int x1, int x2, int x3);//Выбирает как посчитать

using namespace std;

int main()

{

int a, b, c;

cout << "Enter 3 numbers:";

cin >> a >> b >> c;

if (a < b + c)

fun = sum;

else

if (a > b + c)

fun = dif;

else

fun = sumdif;

cout << "Result: " << fun(a, b, c) << endl;

system("pause");

}

int sum(int x1, int x2, int x3) {

return(x1 + x2 + x3);

}

int dif(int x1, int x2, int x3) {

return(x1 - x2 - x3);

}

int sumdif(int x1, int x2, int x3) {

return(x1 + x2 - x3);

}

Программа 2  
#include<iostream>

#include<ctime>

void invert(int \*arr, int size);//Инвертирует массив

void sortInc(int \*arr, int size);//Сортирует по возрастанию

void sortDec(int \*arr, int size);//Сортирует по убыванию

void(\*fun(int \*arr, int size))(int \*arr, int size);//Выбирает,как изменить массив

using namespace std;

int main()

{

srand(time(NULL));

int n;

cout << "Enter size:";

cin >> n;

int \*a = new int[n];

cout << "It's an array: ";

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

{

a[i] = rand() % 100;

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

}

fun(a, n)(a, n);

cout << "\nAnd it's an array after some magic: ";

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

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

cout << endl;

system("pause");

}

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

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

swap(arr[i], arr[size - i - 1]);

}

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

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

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

if (arr[j] > arr[j + 1])

swap(arr[j], arr[j + 1]);

}

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

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

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

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

swap(arr[j], arr[j + 1]);

}

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

int sum = 0;

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

sum += arr[i];

if (sum == arr[1])

return invert;

if (sum > arr[1])

return sortInc;

if (sum < arr[1])

return sortDec;

}

**Вывод**

Для достижения поставленных целей были использованы указатели на функции с различными возвращаемыми значениями. Результат работы программ показан ниже (Рис.3 и 4). Исходный код программ также доступен на GitHub по ссылке: <https://github.com/WarerBit/OOP/tree/master/Lab2>



Рис. 3 Результат работы программы 1

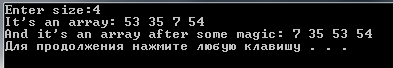


Рис. 4 Результат работы программы 2

**Лабораторная работа №3**

**Классы**

**Цель практической работы**

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

**Задачи**

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

**Ход работы**

Программа 1  
Файл Child.h  
#pragma once

#include <string>

#define AGE\_CH 1

#define NAME\_CH 2

#define SURNAME\_CH 3

using namespace std;

class Child

{

private:

int age;

string name;

string surname;

public:

Child();//Конструктор

~Child();//Деструктор

void setVal(int val, int set);//Задать значение age

void setVal(int val, string set);//Задать значение name,surname.

void getVal(int val);//Вывести значение(age,name,surname) на экран

};

Файл Child.cpp  
#include "Child.h"

#include<iostream>

Child::Child()

{

}

void Child::setVal(int val, int set) {

age = set;

}

void Child::setVal(int val, string set) {

if (val == NAME\_CH)

name = set;

else

surname = set;

}

void Child::getVal(int val) {

if (val == AGE\_CH)

cout << "Age is " << age << endl;

if (val == NAME\_CH)

cout << "Name is " << name << endl;

if (val == SURNAME\_CH)

cout << "Surname is " << surname << endl;

}

Child::~Child()

{

}

Файл Source.cpp  
#include"Child.h"

#include<iostream>

using namespace std;

int main()

{

Child son, daughter;

son.setVal(AGE\_CH, 12);

son.setVal(NAME\_CH, "Tolik");

son.setVal(SURNAME\_CH, "Ivanov");

cout << "Son:\n";

son.getVal(AGE\_CH);

son.getVal(NAME\_CH);

son.getVal(SURNAME\_CH);

daughter.setVal(AGE\_CH, 8);

daughter.setVal(NAME\_CH, "Masha");

daughter.setVal(SURNAME\_CH, "Ivanova");

cout << "\nDaughter:\n";

daughter.getVal(AGE\_CH);

daughter.getVal(NAME\_CH);

daughter.getVal(SURNAME\_CH);

system("pause");

}

Программа 2  
Файл Tiles.h  
#pragma once

#include<string>

#include<iostream>

using namespace std;

class Tiles

{

public:

int price;

int size\_h;

int size\_w;

string brand;

Tiles(int size\_h, int size\_w, int price, string brand);//Конструктор

~Tiles();//Деструктор

void getData();//Вывести данные на экран

};

Файл Tiles.cpp  
#include "Tiles.h"

Tiles::Tiles(int size\_h, int size\_w, int price, string brand)

{

this->price = price;

this->size\_h = size\_h;

this->size\_w = size\_w;

this->brand = brand;

}

void Tiles::getData() {

cout << brand << endl << "H - " << size\_h << endl << "W - " << size\_w << endl << "Price:" << price << endl << endl;

}

Tiles::~Tiles()

{

}

Файл Source.cpp  
#include"Tiles.h"

using namespace std;

int main()

{

Tiles tile1(5, 5, 100, "<<GopstopOOO>>"), tile2(3, 4, 40, "<<DinDong>>"), tile3(1, 1, 10, "<<PingPong>>");

cout << "First tiles:\n";

tile1.getData();

cout << "Second tiles:\n";

tile2.getData();

cout << "Third tiles:\n";

tile3.getData();

system("pause");

}

Программа 3  
Файл Complex.h  
#pragma once

#include<cmath>

#include<iostream>

using namespace std;

class Complex

{

private:

double re;

double im;

public:

Complex(double, double);//Конструктор

~Complex();//Деструктор

void getModule();//Получить модуль комплексного числа

void getArg();//Получить аргумент комплексного числа

};

Файл Complex.cpp  
#include "Complex.h"

Complex::Complex(double re, double im)

{

this->re = re;

this->im = im;

}

void Complex::getModule() {

cout << "Module = " << (sqrt(re\*re + im \* im)) << endl;

}

void Complex::getArg() {

if (re > 0)

cout << "Argument = " << atan(im / re) << endl;

if ((re < 0) and (im >= 0))

cout << "Argument = " << atan(im / re) << "+ pi\n";

if ((re < 0) and (im < 0))

cout << "Argument = " << atan(im / re) << "- pi\n";

}

Complex::~Complex()

{

}

Файл Source.cpp  
#include"Complex.h"

int main()

{

Complex c1(2, -6);

c1.getModule();

c1.getArg();

system("pause");

}

Программа 4  
Файл Vector.h  
#pragma once

#include<iostream>

#include<cmath>

using namespace std;

class Vector

{

private:

int x;

int y;

int z;

public:

Vector();//Конструкторы

Vector(int x, int y, int z);

~Vector();//Деструктор

void getModule();//Получить модуль вектора

void getVec();//Вывести координаты вектор на экран

Vector& operator=(const Vector&);//Перегруженный оператор присваивания

friend const Vector operator+(const Vector&, const Vector&);//Сложение

friend const Vector operator-(const Vector&, const Vector&);//Вычитание

};

Файл Vector.cpp  
#include "Vector.h"

Vector::Vector(int x, int y, int z)

{

this->x = x;

this->y = y;

this->z = z;

}

Vector::Vector()

{

}

void Vector::getModule() {

cout << "Module = " << sqrt(x\*x + y \* y + z \* z) << endl;

}

void Vector::getVec() {

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

}

Vector& Vector::operator=(const Vector& a) {

x = a.x;

y = a.y;

z = a.z;

return \*this;

}

const Vector operator+(const Vector& a, const Vector& b) {

return Vector(a.x + b.x, a.y + b.y, a.z + b.z);

}

const Vector operator-(const Vector& a, const Vector& b) {

return Vector(a.x - b.x, a.y - b.y, a.z - b.z);

}

Vector::~Vector()

{

}

Файл Source.cpp  
#include "Vector.h"

int main()

{

Vector v1(1, 2, 3), v2(4, 5, 6), v3;

cout << "Vector1:";

v1.getModule();

cout << "Vector2:";

v2.getModule();

v3 = v1 + v2;

cout << "Vector1 + Vector2 = ";

v3.getVec();

v3 = v1 - v2;

cout << "Vector1 - Vector2 = ";

v3.getVec();

system("pause");

}

**Вывод**

Для получения требуемого результата были использованы конструкции класса и перегрузка некоторых операторов. Результат работы программ показан ниже (Рис.5-8). Исходный код программ также доступен на GitHub по ссылке:<https://github.com/WarerBit/OOP/tree/master/Lab3>

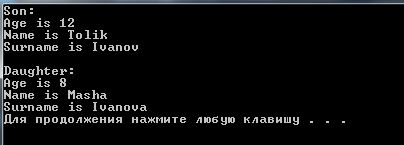


Рис. 5 Результат работы программы 1

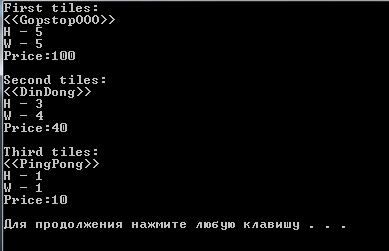


Рис. 6 Результат работы программы 2



Рис. 7 Результат работы программы 3

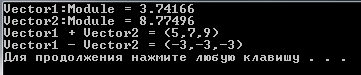


Рис. 8 Результат работы программы 4

**Лабораторная работа №4**

**Конструкторы и деструкторы**

**Цель практической работы**

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

**Задачи**

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

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

**Ход работы**

Программа 1  
Файл Complex.h  
#pragma once

#include<cmath>

#include<iostream>

using namespace std;

class Complex

{

private:

double re;

double im;

public:

Complex();//Контрукторы

Complex(const Complex& compl\_obj);

Complex(double, double);

~Complex();//Деструктор

void getModule();//Получить модуль комплексного числа

void getArg();//Получить аргумент комплексного числа

};

Файл Complex.cpp  
#include "Complex.h"

Complex::Complex(double re, double im)

{

this->re = re;

this->im = im;

}

Complex::Complex()

{

}

Complex::Complex(const Complex& compl\_obj)

{

}

void Complex::getModule() {

cout << "Module = " << (sqrt(re\*re + im \* im)) << endl;

}

void Complex::getArg() {

if (re > 0)

cout << "Argument = " << atan(im / re) << endl;

if ((re < 0) and (im >= 0))

cout << "Argument = " << atan(im / re) << "+ pi\n";

if ((re < 0) and (im < 0))

cout << "Argument = " << atan(im / re) << "- pi\n";

}

Complex::~Complex()

{

}

Файл Source.cpp  
#include"Complex.h"

int main()

{

Complex c1(2, -6);

c1.getModule();

c1.getArg();

system("pause");

}  
  
  
  
Программа 2  
Файл Vector.h  
#pragma once

#include<iostream>

#include<cmath>

using namespace std;

class Vector

{

private:

int x;

int y;

int z;

public:

Vector();//Конструкторы

Vector(const Vector& vec\_obj);

Vector(int, int, int);

~Vector();//Деструктор

void getModule();//Получить модуль вектора

void getVec();//Вывести координаты вектора на экран

Vector& operator=(const Vector&);//Присваивание

friend const Vector operator+(const Vector&, const Vector&);//Сложение

friend const Vector operator-(const Vector&, const Vector&);//Вычитание

};  
  
Файл Vector.cpp  
#include "Vector.h"

Vector::Vector(int x, int y, int z)

{

this->x = x;

this->y = y;

this->z = z;

}

Vector::Vector()

{

}

Vector::Vector(const Vector& vec\_obj)

{

}

void Vector::getModule() {

cout << "Module = " << sqrt(x\*x + y \* y + z \* z) << endl;

}

void Vector::getVec() {

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

}

Vector& Vector::operator=(const Vector& a) {

x = a.x;

y = a.y;

z = a.z;

return \*this;

}

const Vector operator+(const Vector& a, const Vector& b) {

return Vector(a.x + b.x, a.y + b.y, a.z + b.z);

}

const Vector operator-(const Vector& a, const Vector& b) {

return Vector(a.x - b.x, a.y - b.y, a.z - b.z);

}

Vector::~Vector()

{

}  
  
Файл Source.cpp  
#include "Vector.h"

int main()

{

Vector v1(1, 2, 3), v2(4, 5, 6), v3;

cout << "Vector1:";

v1.getModule();

cout << "Vector2:";

v2.getModule();

v3 = v1 + v2;

cout << "Vector1 + Vector2 = ";

v3.getVec();

v3 = v1 - v2;

cout << "Vector1 - Vector2 = ";

v3.getVec();

system("pause");

}  
  
  
  
Программа 3  
Файл Tiles.h  
#pragma once

#include<string>

#include<iostream>

using namespace std;

class Tiles

{

public:

int price;

int size\_h;

int size\_w;

string brand;

Tiles();//Конструкторы

Tiles(const Tiles& tile\_obj);

Tiles(int size\_h, int size\_w, int price, string brand);

~Tiles();//Деструктор

void getData();//Вывести данные на экран

};  
  
Файл Tiles.cpp  
#include "Tiles.h"

Tiles::Tiles(int size\_h, int size\_w, int price, string brand)

{

this->price = price;

this->size\_h = size\_h;

this->size\_w = size\_w;

this->brand = brand;

}

Tiles::Tiles()

{

}

Tiles::Tiles(const Tiles& tile\_obj)

{

}

void Tiles::getData() {

cout << brand << endl << "H - " << size\_h << endl << "W - " << size\_w << endl << "Price:" << price << endl << endl;

}

Tiles::~Tiles()

{

}  
  
Файл Source.cpp

#include"Tiles.h"

using namespace std;

int main()

{

Tiles tile1(5, 5, 100, "<<GopstopOOO>>"), tile2(3, 4, 40, "<<DinDong>>"), tile3(1, 1, 10, "<<PingPong>>");

cout << "First tiles:\n";

tile1.getData();

cout << "Second tiles:\n";

tile2.getData();

cout << "Third tiles:\n";

tile3.getData();

system("pause");

}

Программа 4  
Файл Child.h

#pragma once

#include <string>

#define AGE\_CH 1

#define NAME\_CH 2

#define SURNAME\_CH 3

using namespace std;

class Child

{

private:

int age;

string name;

string surname;

public:

Child();//Конструкторы

Child(const Child& ch\_obj);

Child(int age, string name, string surname);

~Child();//Деструктор

void setVal(int val, int set);//Задать значение age

void setVal(int val, string set);//Задать значение name,surname

void getVal(int val);//Вывести значение(age,name,surname) на экран

};

Файл Child.cpp

#include "Child.h"

#include<iostream>

Child::Child()

{

}

Child::Child(int age, string name, string surname)

{

this->age = age;

this->name = name;

this->surname = surname;

}

Child::Child(const Child& ch\_obj)

{

}

void Child::setVal(int val, int set) {

age = set;

}

void Child::setVal(int val, string set) {

if (val == NAME\_CH)

name = set;

else

surname = set;

}

void Child::getVal(int val) {

if (val == AGE\_CH)

cout << "Age is " << age << endl;

if (val == NAME\_CH)

cout << "Name is " << name << endl;

if (val == SURNAME\_CH)

cout << "Surname is " << surname << endl;

}

Child::~Child()

{

}

Файл Source.cpp

#include"Child.h"

#include<iostream>

using namespace std;

int main()

{

Child son, daughter;

son.setVal(AGE\_CH, 12);

son.setVal(NAME\_CH, "Tolik");

son.setVal(SURNAME\_CH, "Ivanov");

cout << "Son:\n";

son.getVal(AGE\_CH);

son.getVal(NAME\_CH);

son.getVal(SURNAME\_CH);

daughter.setVal(AGE\_CH, 8);

daughter.setVal(NAME\_CH, "Masha");

daughter.setVal(SURNAME\_CH, "Ivanova");

cout << "\nDaughter:\n";

daughter.getVal(AGE\_CH);

daughter.getVal(NAME\_CH);

daughter.getVal(SURNAME\_CH);

system("pause");

}

**Вывод**

В процессе работы были более подробно изучены разные виды конструкторов и деструкторы. Для выполнения работы были использованы простые конструкторы, конструкторы копирования. Результат работы программ показан ниже (Рис.9-12). Исходный код программ также доступен на GitHub по ссылке:<https://github.com/WarerBit/OOP/tree/master/Lab4>



Рис. 9 Результат работы программы 1

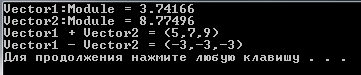


Рис. 10 Результат работы программы 2

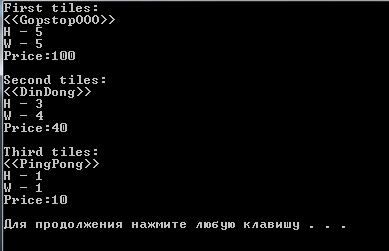


Рис. 11 Результат работы программы 3

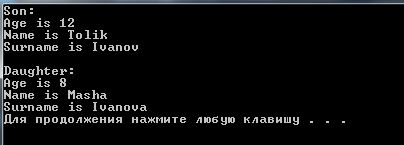


Рис. 12 Результат работы программы 4

**Лабораторная работа №5**

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

**Цель практической работы**

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

**Задачи**

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

**Ход работы**

Программа 1

Файл Complex.h  
#pragma once

#include<cmath>

#include<iostream>

using namespace std;

class Complex

{

protected:

double re;

double im;

public:

Complex(double, double);//Конструкторы

Complex();

~Complex();//Деструктор

Complex operator=(const Complex&);//Присваивание

friend const Complex operator++(Complex&, int);//Инкремент

friend const Complex operator--(Complex&, int);//Декремент

friend const ostream& operator<<(ostream&, const Complex&);//Вывод

friend const istream& operator>>(istream&, Complex&);//Ввод

friend const bool operator==(const Complex&, const Complex&);//Сравнение

};

Файл Complex.cpp  
#include "Complex.h"

Complex::Complex(double re, double im)

{

this->re = re;

this->im = im;

}

Complex::Complex()

{

}

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

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

}

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

Complex complex(c);

c.re++;

return complex;

}

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

Complex complex(c);

c.re--;

return complex;

}

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;

}

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

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

}

Complex::~Complex()

{

}

Файл Source.cpp  
#include"Complex.h"

int main()

{

Complex c;

cout << "Enter complex(x+yi): ";

cin >> c;

Complex c2 = c;

c++;

cout << "\nFirst complex after ++: ";

cout << c;

c2--;

cout << "\nSecond complex after --: ";

cout << c2;

if (c == c2)

cout << "\nFirst complex = second complex\n";

else

cout << "\nFirst complex != second complex\n";

system("pause");

}

Программа 2  
Файл Vector.h

#pragma once

#include<iostream>

#include<cmath>

using namespace std;

class Vector

{

int x;

int y;

int z;

public:

Vector(int, int, int);//Конструкторы

Vector();

~Vector();//Деструктор

Vector operator=(const Vector&);//Присваивание

friend const ostream& operator<<(ostream&, const Vector&);//Вывод

friend const istream& operator>>(istream&, Vector&);//Ввод

friend const bool operator==(const Vector&, const Vector&);//Сравнение

};

Файл Vector.cpp

#include "Vector.h"

Vector::Vector()

{

}

Vector::Vector(int x, int y, int z)

{

this->x = x;

this->y = y;

this->z = z;

}

Vector Vector::operator=(const Vector& v) {

return Vector(v.x, v.y, v.z);

}

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

out << "(" << v.x << "," << v.y << "," << v.z << ")";

return out;

}

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

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

in.get();

return in;

}

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

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

}

Vector::~Vector()

{

}

Файл Source.cpp

#include"Vector.h"

int main()

{

Vector v1;

cout << "Enter vector(x,y,z):";

cin >> v1;

Vector v2 = v1;

cout << "\nThis is the second vector - ";

cout << v2;

cout << "\nCheck if v1 = v2:";

if (v1 == v2)

cout << "\nThere are 2 similar vectors.\n";

system("pause");

}

**Вывод**

В процессе выполнения работы был более плотно изучен механизм перегрузки. Для выполнения поставленных задач была использована перегрузка некоторых бинарных и унарных операторов. Результат работы программ показан ниже (Рис.13-14). Исходный код программ также доступен на GitHub по ссылке:<https://github.com/WarerBit/OOP/tree/master/Lab5>

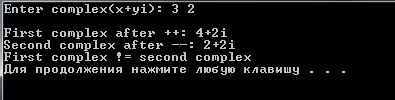


Рис. 13 Результат работы программы 1

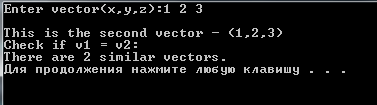


Рис. 14 Результат работы программы 2

**Лабораторная работа №6**

**Наследование**

**Цель практической работы**

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

**Задачи**

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

**Ход работы**

Программа 1  
Файл Student.h  
#pragma once

#include <iostream>

#include<string>

#include<ctime>

using namespace std;

class Student//Базовый класс

{

protected:

int age;

int number;

string name;

public:

Student(int age, int number, string name) {

this->age = age;

this->number = number;

this->name = name;

}

~Student()

{

}

int getAge() {

return age;

}

int getNumber() {

return number;

}

string getName() {

return name;

}

};

class Starosta : public Student {//Производный класс

private:

string group;

int missed;

public:

Starosta(int age, int number, string name, string group, int missed) : Student(age, number, name) {

this->group = group;

this->missed = missed;

}

string getGroup() {

return group;

}

int getMiss() {

return missed;

}

void miss() {

srand(time(NULL));

int r = rand() % 4;

cout << endl << name << " today have decided to miss " << r << " classes.";

missed += r;

}

void check() {

srand(time(NULL));

int r = rand() % 30;

cout << endl << r << " students have missed classes today.\n";

}

};  
  
Файл Source.cpp

#include"Student.h"

int main()

{

srand(time(NULL));

Starosta st(18, 7, "Ivan Ivanov", "INBO-04-18", rand() % 10);

cout << "Starosta: " << st.getName() << " " << "\nAge - " << st.getAge() << "\nNumber - " << st.getNumber() << "\nGroup - " << st.getGroup() << "\nAlready missed " << st.getMiss() << " classes\n";

st.miss();

cout << endl;

st.check();

system("pause");

}

Программа 2  
Файл Alive.h  
#pragma once

#include <iostream>

#include <string>

using namespace std;

class Alive//Базовый класс

{

protected:

int age;

string spicies;

public:

Alive(int age, string spicies) {

this->age = age;

this->spicies = spicies;

}

~Alive()

{

}

int getAge() {

return age;

}

string getSpicies() {

return spicies;

}

};

class Bird : public Alive {//Производный класс

private:

bool flying;

public:

Bird(int age, string spicies, bool flying) : Alive(age, spicies) {

this->flying = flying;

}

string getFlyable() {

if (flying)

return "Can fly";

else

return "Can't fly";

}

};

class Fish : public Alive {//Производный класс

private:

string depth;

public:

Fish(int age, string spicies, string depth) : Alive(age, spicies) {

this->depth = depth;

}

string getDepth() {

return depth;

}

};

class Animal : public Alive {//Производный класс

private:

int legs;

public:

Animal(int age, string spicies, int legs) : Alive(age, spicies) {

this->legs = legs;

}

int getLegs() {

return legs;

}

};  
Файл Source.cpp  
#include"Alive.h"

#include<ctime>

int main()

{

srand(time(NULL));

Bird bi(rand() % 10, "penguine", false);

Fish fi(rand() % 15, "shark", "average");

Animal an(rand() % 10, "cat", 4);

cout << "Bird:\n" << "Age - " << bi.getAge() << "\nSpicies - " << bi.getSpicies() << "\nFlyable - " << bi.getFlyable();

cout << "\nFish:\n" << "Age - " << fi.getAge() << "\nSpicies - " << fi.getSpicies() << "\nDepth - " << fi.getDepth();

cout << "\nAnimal:\n" << "Age - " << an.getAge() << "\nSpicies - " << an.getSpicies() << "\nLegs - " << an.getLegs();

cout << endl;

system("pause");

}

Программа 3  
Файл Animal.h  
#pragma once

#include <string>

#include <iostream>

using namespace std;

class Animal//Базовый класс

{

protected:

int age;

int legs;

string color;

public:

Animal(int age, int legs, string color) {

this->age = age;

this->legs = legs;

this->color = color;

}

virtual void sound() = 0;

void setLegs(int legs) {

this->legs = legs;

}

void setAge(int age) {

this->age = age;

}

void setColor(string color) {

this->color = color;

}

int getLegs() {

return legs;

}

int getAge() {

return age;

}

string getColor() {

return color;

}

};

class Cat : public Animal {//Производный класс

public:

Cat(int age, int legs, string color) : Animal(age, legs, color) {}

void sound() {

cout << "Nya~" << endl;

}

};

class Dog : public Animal {//Производный класс

public:

Dog(int age, int legs, string color) : Animal(age, legs, color) {}

void sound() {

cout << "Woof" << endl;

}

};  
Файл Source.cpp

#include "Animal.h"

#include <time.h>

int main()

{

srand(time(NULL));

string sc = "black", sd = "white";

int ch;

cout << "Enter:\n 1 - Cat\n 2 - Dog" << endl;

cin >> ch;

switch (ch)

{

case 1: {

Cat cat(rand() % 9 + 1, 4, sc);

cout << "Cat's age is " << cat.getAge() << ", it has " << cat.getLegs() << " legs,and it's color is " << sc << endl;

cout << "Cat says - ";

cat.sound();

break;

}

case 2: {

Dog dog(rand() % 9 + 1, 4, sd);

cout << "Dog's age is " << dog.getAge() << ", it has " << dog.getLegs() << " legs,and it's color is " << sd << endl;

cout << "Dog says - ";

dog.sound();

break;

}

default:

break;

}

system("pause");

}

**Вывод**

В процессе выполнения работы было изучено и использовано наследование. Результат работы программ показан ниже (Рис.15-17). Исходный код программ также доступен на GitHub по ссылке:

<https://github.com/WarerBit/OOP/tree/master/Lab6>

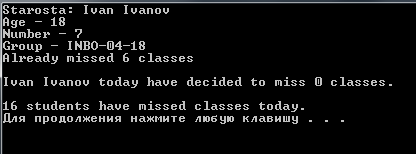


Рис. 15 Результат работы программы 1

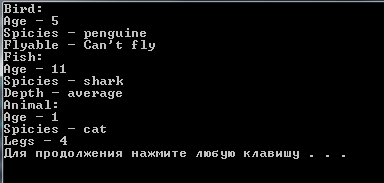


Рис. 16 Результат работы программы 2

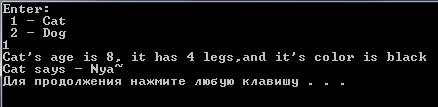


Рис. 17 Результат работы программы 3

**Лабораторная работа №7**

**Создание абстрактных классов**

**Цель практической работы**

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

**Задачи**

Реализовать систему из классов, изображённую на UML диаграмме.

**Ход работы**

Программа 1  
Файл Human.h  
#pragma once

#include<string>

#include<iostream>

using namespace std;

class Human

{

protected:

string name;

string surname;

string midname;

int age;

public:

Human() {}

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

{

this->name = name;

this->surname = surname;

this->midname = midname;

this->age = age;

}

~Human() {}

virtual void print() = 0;

};

class Student : public Human {

private:

bool on\_lesson;

public:

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

this->on\_lesson = on\_lesson;

}

void print() {

cout << "Name: " << surname << " " << name << " " << midname << "\nAge - " << age;

if (on\_lesson)

cout << "\nThe student is on the lesson.\n";

else

cout << "\nThe student isn't on the lesson.\n";

}

};

class Boss : public Human {

private:

int number\_of\_workers;

public:

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

this->number\_of\_workers = number\_of\_workers;

}

void print() {

cout << "Name: " << surname << " " << name << " " << midname << "\nAge - " << age << "\nNumber of workers - " << number\_of\_workers << endl;

}

};

Файл Source.cpp  
#include "Human.h"

#include<ctime>

int main()

{

srand(time(NULL));

Student st("Ivan", "Ivanov", "Ivanovich", rand() % 4 + 17, rand() % 2);

Boss bs("Petr", "Petrov", "Petrovich", rand() % 10 + 30, rand() % 10 + 20);

cout << "Student:\n";

st.print();

cout << "\nBoss:\n";

bs.print();

system("pause");

}

**Вывод**

При выполнении работы были изучены основные элементы UML диаграмм. Для достижения поставленной цели были использованы механизмы наследования, переопределения функций и абстрактные классы. Результат работы программ показан ниже (Рис.18). Исходный код программ также доступен на GitHub по ссылке: <https://github.com/WarerBit/OOP/tree/master/Lab7>

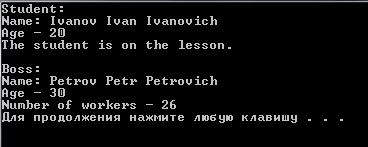


Рис. 18 Результат работы программы

**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**

Программа 1  
Файл Student.h  
#pragma once

#include <iostream>

#include<string>

using namespace std;

class Student

{

protected:

int age;

string name;

public:

Student(int age, string name) {

this->age = age;

this->name = name;

}

~Student()

{

}

int getAge() {

return age;

}

string getName() {

return name;

}

};

Файл Source.cpp  
#include"Student.h"

#include<vector>

int main()

{

vector<Student> stud;

int age; string name;

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

{

cout << "Enter age and name for " << i + 1 << " student:";

cin >> age >> name;

stud.push\_back(Student(age, name));

}

for (auto s : stud)

cout << "\nStudent " << ": Age - " << s.getAge() << ", Name -" << s.getName();

system("pause");

}

Программа 2  
Файл Complex.h  
#pragma once

#include<iostream>

#include<cmath>

using namespace std;

class Complex

{

private:

double re;

double im;

public:

Complex()

{

}

Complex(double re, double im)

{

this->re = re;

this->im = im;

}

void getComplex() {

cout << re;

if (im > 0)

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

else

cout << im << "i";

cout << endl;

}

~Complex()

{

}

};  
  
Файл Source.cpp

#include"Complex.h"

#include<list>

int main()

{

double re, im;

list<Complex> complex;

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

{

cout << "Enter complex " << i + 1 << " re and im:";

cin >> re >> im;

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

}

cout << endl;

for (auto c : complex)

c.getComplex();

system("pause");

}

Программа 3  
Файл Car.h  
#pragma once

#include<string>

#include<iostream>

using namespace std;

class Car

{

private:

string color;

string model;

public:

Car(string color, string model)

{

this->color = color;

this->model = model;

}

~Car()

{}

void getCar() {

cout << "Color - " << color << "\nModel - " << model << endl;

}

};  
  
Файл Source.cpp

#include"Car.h"

#include<queue>

int main()

{

queue<Car> cars;

string color, model;

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

{

cout << "Enter " << i + 1 << " car(color,model):";

cin >> color >> model;

cars.push(Car(color, model));

}

for (int i = 0; !cars.empty(); i++)

{

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

cars.front().getCar();

cars.pop();

}

system("pause");

}

**Вывод**

Была изучена библиотека STL. Для выполнения работы были использованы коллекции queue, string, list и vector. Результат работы программ показан ниже (Рис.19-20). Исходный код программ также доступен на GitHub по ссылке:  
<https://github.com/WarerBit/OOP/tree/master/Lab8>

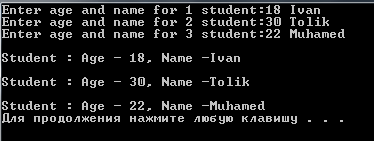


Рис. 18 Результат работы программы 1

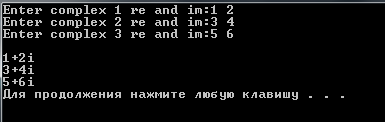


Рис. 19 Результат работы программы 2

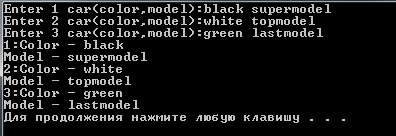


Рис. 20 Результат работы программы 3