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

|  |  |
| --- | --- |
| **ОТЧЕТ ПО ЛАБОРАТОРНЫМ РАБОТАМ** | |
| **по дисциплине** | |
| «Объектно-ориентированное программирование» | |
|  | |
| Выполнил студент группы ИНБО-04-18 | Кузин М.В. |
| Принял старший преподаватель | Хлебникова В.Л. |

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

Москва 2019

Содержание

[Содержание 2](#_Toc8582922)

[Таблица успеваемости 3](#_Toc8582923)

[Лабораторная работа №1 4](#_Toc8582924)

[Лабораторная работа №2 10](#_Toc8582925)

[Лабораторная работа №3 13](#_Toc8582926)

[Лабораторная работа №4 20](#_Toc8582927)

[Лабораторная работа №5 28](#_Toc8582928)

[Лабораторная работа №6 35](#_Toc8582929)

[Лабораторная работа №7 40](#_Toc8582930)

[Лабораторная работа №8 43](#_Toc8582931)

# Таблица успеваемости

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| №/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

**Задание:**

1. Напишите программу–калькулятор комплексных чисел. Для

реализации необходимо разработать абстрактный тип данных–

комплексное число. Программа должна реализовывать

арифметические операции над комплексными числами. Программа

должна быть представлена в виде многофайлового проекта, все

прототипы функций, объявления структур должны быть вынесены в

заголовочный файл с соответствующим названием. Всего должно

быть три файла: файл с объявлениями, файл реализации и файл с

функцией main(), демонстрирующий работу с новым типом данных.

Файл с объявлениями должен называться Complex.h, файл с

реализацией функций должен называться Complex.с, файл с функцией

main() может называться main.с Программа должна обеспечивать

удобный интерфейс пользователя для работы с ней

2. На основе разработанного в предыдущем задании типа данных

комплексное число написать программу, которая считывает

информацию из файла complex.txt — количество комплексных чисел в

переменную n, а сами комплексные числа в массив p. Затем

происходит поиск комплексного числа с максимальным модулем в

массиве p.

**Ход выполнения:**

Вариант 1

Complex.cpp

|  |  |  |
| --- | --- | --- |
|  |  | #include "Complex.h"  #include <iostream> |
|  |  | using namespace std; |
|  |  |  |
|  |  | Complex::Complex() |
|  |  | { |
|  |  | re = 0; |
|  |  | im = 0; |
|  |  | } |
|  |  |  |
|  |  | Complex::Complex(double re, double im) |
|  |  | { |
|  |  | this->re = re; |
|  |  | this->im = im; |
|  |  | } |
|  |  |  |
|  |  | Complex operator+ (Complex a, Complex b) |
|  |  | { |
|  |  | Complex c; |
|  |  | c.re =a.re+b.re; |
|  |  | c.im =a.im+b.im; |
|  |  | return c; |
|  |  | } |
|  |  |  |
|  |  | Complex operator- (Complex a, Complex b) |
|  |  | { |
|  |  | Complex c; |
|  |  | c.re = a.re - b.re; |
|  |  | c.im = a.im - b.im; |
|  |  | return c; |
|  |  | } |
|  |  |  |
|  |  | Complex operator\* (Complex a, Complex b) |
|  |  | { |
|  |  | Complex c; |
|  |  | c.re = a.re\*b.re - a.im\*b.im; |
|  |  | c.im = a.im\*b.re + a.re\*b.im; |
|  |  | return c; |
|  |  | } |
|  |  |  |
|  |  | Complex operator/ (Complex a, Complex b) |
|  |  | { |
|  |  | Complex c; |
|  |  | c.re = (a.re\*b.re + a.im\*b.im) / (b.re\*b.re + b.im\*b.im); |
|  |  | c.im = (a.im\*b.re - a.re\*b.im) / (b.re\*b.re + b.im\*b.im); |
|  |  | return c; |
|  |  | } |
|  |  |  |
|  |  | void Complex::Print() |
|  |  | { |
|  |  | cout << re << "+" << im << "i" << endl; |
|  |  | } |
|  |  |  |
|  |  | double Complex::module() |
|  |  | { |
|  |  | return sqrt(this->re\*this->re + this->im \* this->im); |
|  |  | } |
|  |  |  |
|  |  | double Complex::getRe() |
|  |  | { |
|  |  | return this->re; |
|  |  | } |
|  |  |  |
|  |  | double Complex::getIm() |
|  |  | { |
|  |  | return this->im; |
|  |  | } |
|  |  |  |
|  |  | Complex Complex::pow(int p) { |
|  |  | Complex res; |
|  |  | res = \*this; |
|  |  | while (p > 1) { |
|  |  | res = res \* \*this; |
|  |  | p--; |
|  |  | } |
|  |  | return res; |
|  |  | } |

Complex.h

|  |  |
| --- | --- |
| #pragma once | |
|  | | class Complex | | |
|  | | { | | |
|  | | private: | | |
|  | | double re; | | |
|  | | double im; | | |
|  | | public: | | |
|  | | Complex(); | | |
|  | | Complex(double re, double im); | | |
|  | | friend Complex operator+(Complex a, Complex b); | | |
|  | | friend Complex operator-(Complex a, Complex b); | | |
|  | | friend Complex operator\*(Complex a, Complex b); | | |
|  | | friend Complex operator/(Complex a, Complex b); | | |
|  | | void Print(); | | |
|  | | double module(); | | |
|  | | double getRe(); | | |
|  | | double getIm(); | | |
|  | | Complex pow(int p); | | |
|  | | }; | | |
|  | |  |
|  | |  |
|  | |  |
|  | |  |
|  | |  |
|  | |  |
|  | |  |
|  | |  |
|  | |  |
|  | |  |
|  | |  |
|  | |  |
|  | |  |
|  | |  |
|  | |  |
|  | |  |
|  | |  |
|  | |  |

main.cpp

|  |
| --- |
|  |
|  | #include "Complex.h"  #include <iostream> |
|  | #include <fstream> |
|  | #include <vector> |
|  | using namespace std; |
|  | int main() |
|  | { |
|  | setlocale(0, "rus"); |
|  | cout << "1 - 1,1 2 - 1,2"<<endl; |
|  | int k; |
|  | cin>> k; |
|  | switch (k) |
|  | { |
|  | case 1: |
|  | { |
|  | double re1, re2, im1, im2; |
|  | cout << "Введите первое комплексное число(сначала действительную часть, а через пробел мнимую)" << endl; |
|  | cin >> re1 >> im1; |
|  | cout << " Введите второе комплексное число(сначала действительную часть, а через пробел мнимую)" << endl; |
|  | cin >> re2 >> im2; |
|  | Complex a(re1, im1), b(re2, im2), c; |
|  | c = a + b; |
|  | cout << "Сумма введенных чисел = "; |
|  | c.Print(); |
|  | cout << endl; |
|  |  |
|  | c = a - b; |
|  | cout << "Разность введенных чисел = "; |
|  | c.Print(); |
|  | cout << endl; |
|  |  |
|  | c = a \* b; |
|  | cout << "Произведение введенных чисел = "; |
|  | c.Print(); |
|  | cout << endl; |
|  |  |
|  | c = a / b; |
|  | cout << "Частное введенных чисел = "; |
|  | c.Print(); |
|  | cout << endl; |
|  | } |
|  | case 2: |
|  | { |
|  | ifstream file("complex.txt"); |
|  | int n; |
|  | double r, m; |
|  | vector<Complex> p; |
|  | file >> n; |
|  | for (int i = 0; i < n; i++) { |
|  | file >> r >> m; |
|  | Complex el(r, m); |
|  | p.push\_back(el); |
|  | } |
|  | file.close(); |
|  | Complex max; |
|  | double md = p[0].module(); |
|  | for (int i = 0; i < n; i++) |
|  | if (p[i].module() > md) { |
|  | md = p[i].module(); |
|  | max = p[i]; |
|  | } |
|  | cout << max.getRe() << "+" << max.getIm() << "i" << endl; |
|  | } |
|  | default: |
|  | break; |
|  | } |
|  | system("pause"); |
|  | return 0; |
|  | } |

|  |  |  |
| --- | --- | --- |
|  | | |
|  | |  |
|  | | |

**Выводы:**

В ходе выполнения Лабораторной работы №1 были использованы заголовочный файл Complex.h и файлы исходные Complex.cpp, main.cpp. Были перегружены операторы: =,+,-,\*,/,<<,>>. Были использованы библиотеки iostream, fstream. Реализован метод модуля комлексного числа. Были достигнуты результаты:

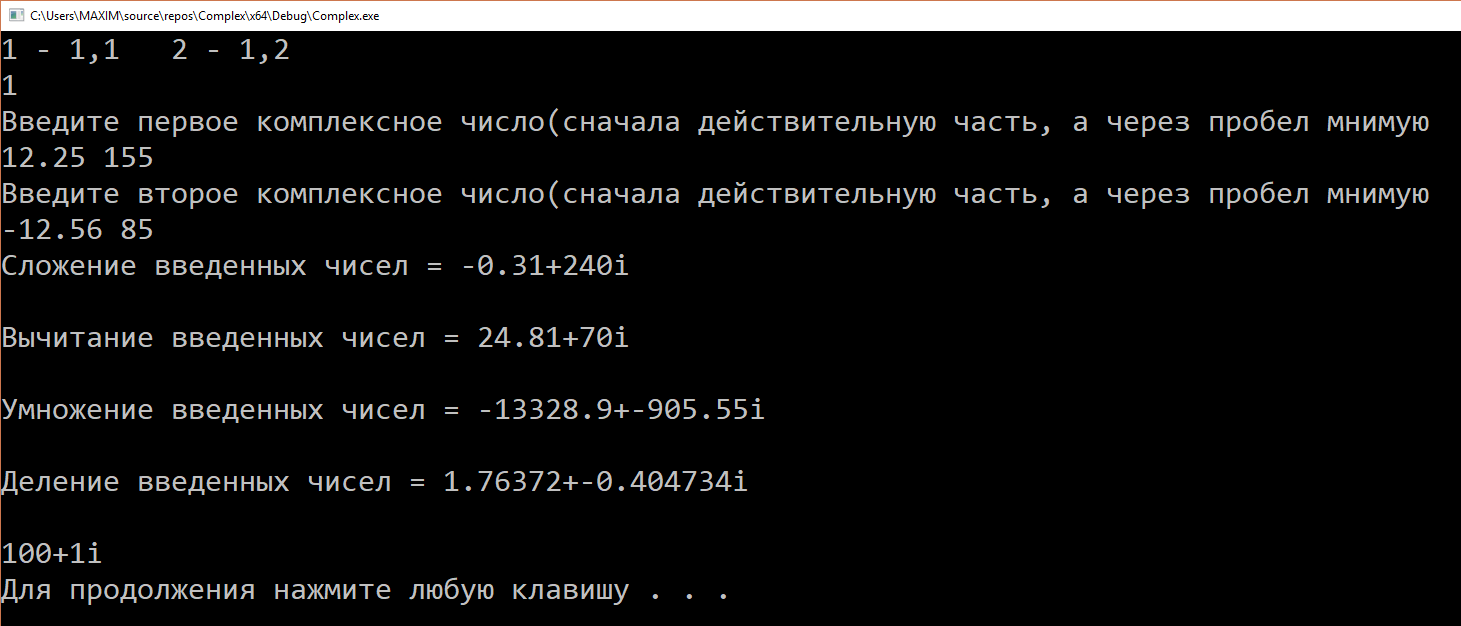


Рис. 1

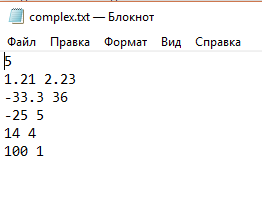


Рис. 2

<https://github.com/FunFuhrer/OOP/commit/8bc24aa9f13604f28eb385a4851c3bf682445501>

# Лабораторная работа №2

**Задание:**

1. Напишите программу, которая вызывает различные виды

функции в зависимости от заданного условия. Можно

использовать примеры выше.

2. Напишите программу с использованием некой универсальной

функции, которая в качестве возвращаемого значения

возвращает указатель на функцию. Эта функция в зависимости

от исходного массива должна соответствующим образом

преобразовать массив. Также в функции main() должен быть

объявлен указатель. В теле программы указателю на функцию

должно присваиваться значение. Исходные данные -

сгенерировать целочисленный массив из случайных чисел.

Запрограммировать следующие действия в виде функций:

* если сумма элементов в массиве равна его первому

элементу, то необходимо инвертировать массив

* если сумма элементов в массиве больше его первого

элемента, то необходимо расположить его элементы в

неубывающем порядке

* если сумма элементов массива меньше его первого

элемента, то необходимо расположить его элементы в

невозрастающем порядке

**Ход выполнения:**

#include <iostream>

#include <ctime>

using namespace std;

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

int t;

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

t = arr[i];

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

arr[size - 1 - i] = t;

}

}

void sort1(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])

swap(arr[j], arr[i]);

}

void sort2(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])

swap(arr[j], arr[i]);

}

void(\*foopointer(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])

return invert;

if (sum > arr[0])

return sort1;

if (sum < arr[0])

return sort2;

}

int main()

{

setlocale(0, "rus");

srand(time(NULL));

int n;

cout << "Размер массива : ";

cin >> n;

int \*arr = new int[n];

cout << "\nНаш массив : ";

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

{

arr[i] = rand() % 51 - 25;

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

}

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

foopointer(arr, n)(arr, n);

cout << "\nМассив после преобразований : ";

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

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

cout << endl;

system("pause");

return 0;

}

**Выводы:**

В ходе выполнения Лабораторной работы №2 были использованы библиотеки iostream, ctime, algorithm. Реализованы функции суммы, инвертации, сортировок элементов массива и сложения, вычитания, умножения, деления чисел. Были достигнуты результаты:

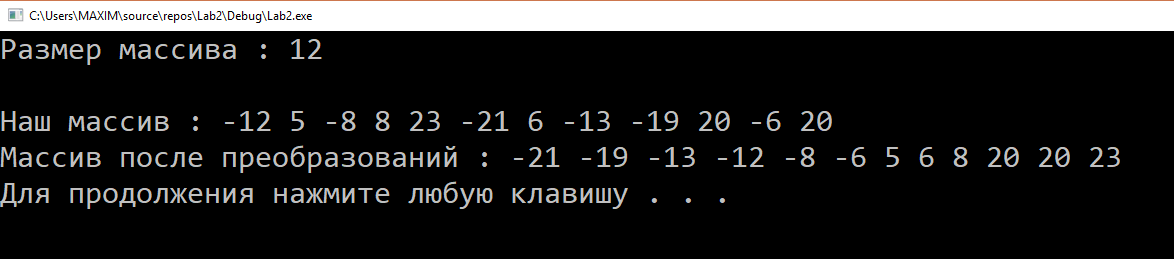


Рис. 3

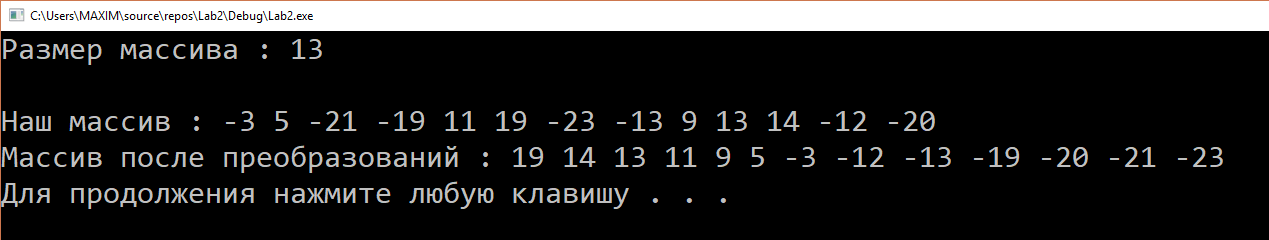


Рис. 4

<https://github.com/FunFuhrer/OOP/blob/master/Lab2.2.cpp>

# Лабораторная работа №3

**Задание:**

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 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 выполнения:**

Вариант 1

|  |
| --- |
|  |
|  | #include <iostream>  #include <string> |
|  | using namespace std; |
|  |  |
|  | class Children { |
|  | string firstName; |
|  | string lastName; |
|  | int years; |
|  | public: |
|  | void set\_data() { |
|  | cout << "Введите имя: "; |
|  | cin >> firstName; |
|  | cout << "Введите фамилию: "; |
|  | cin >> lastName; |
|  | cout << "Введите возраст: "; |
|  | cin >> years; |
|  | } |
|  | void print\_data() { |
|  | cout << "Возраст: " << firstName << endl; |
|  | cout << "Фамилия: " << lastName << endl; |
|  | cout << "Возраст: " << years << endl; |
|  | } |
|  | }; |
|  | int main() { |
|  | setlocale(0, "rus"); |
|  | Children child1, child2; |
|  | child1.set\_data(); |
|  | child2.set\_data(); |
|  | child1.print\_data(); |
|  | child2.print\_data(); |
|  | system("pause"); |
|  | return 0; |
|  | } |

Вариант 2

|  |
| --- |
|  |
| |  | | --- | |  | |  | #include<iostream>  #include <string> | |  | using namespace std; | |  |  | |  | class Tiles { | |  | public: | |  | string brand; | |  | int size\_h, size\_w, price; | |  | void getData() { | |  | cout << "Бренд: " << brand << endl; | |  | cout << "Высота: " << size\_h << endl; | |  | cout << "Ширина: " << size\_w << endl; | |  | cout << "Цена: " << price << endl; | |  | } | |  | }; | |  | int main() { | |  | setlocale(0, "rus"); | |  | Tiles tile1, tile2; | |  | tile1.brand = "nike"; | |  | tile2.brand = "puma"; | |  | tile1.price = 14; | |  | tile2.price = 4; | |  | tile1.size\_h = 14; | |  | tile1.size\_w = 88; | |  | tile2.size\_h = 13; | |  | tile2.size\_w = 37; | |  | tile1.getData(); | |  | tile2.getData(); | |  | system("pause"); | |  | return 0; | |  | } | |

Вариант 3

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| |  | | --- | |  | |  | #include <iostream>  #include <string> | |  | using namespace std; | |  |  | |  | class Complex { | |  | double re, im; | |  | public: | |  | Complex(double re, double im) | |  | { | |  | this->re = re; | |  | this->im = im; | |  | } | |  | void Print() | |  | { | |  | cout << re << "+" << im << "i" << endl; | |  | } | |  |  | |  | double module() | |  | { | |  | return sqrt(this->re\*this->re + this->im \* this->im); | |  | } | |  |  | |  | double getRe() | |  | { | |  | return this->re; | |  | } | |  |  | |  | double getIm() | |  | { | |  | return this->im; | |  | } | |  | }; | |  | int main() { | |  | Complex comp(14.88, 13.37); | |  | cout << comp.module() << endl; | |  | system("pause"); | |  | return 0; | |  | } | |
|  |

Вариант 4

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| |  |  | | --- | --- | |  | #include <iostream>  #include <string> | |  | using namespace std; | |  |  | |  | class Vector { | |  | double x,y,z; | |  | public: | |  | Vector (double x, double y, double z) | |  | { | |  | this->x = x; | |  | this->y = y; | |  | this->z = z; | |  | } | |  | Vector() | |  | { | |  | this->x = 0; | |  | this->y = 0; | |  | this->z = 0; | |  | } | |  | void Print() | |  | { | |  | cout << x << "," << y << "," << z << endl; | |  | } | |  |  | |  | double module() | |  | { | |  | return sqrt(x\*x+y\*y+z\*z); | |  | } | |  |  | |  | friend Vector sum(Vector a, Vector b) | |  | { | |  | Vector c; | |  | c.x=a.x+b.x; | |  | c.y = a.y + b.y; | |  | c.z = a.z + b.z; | |  | return c; | |  | } | |  | friend Vector dif(Vector a, Vector b) | |  | { | |  | Vector c; | |  | c.x = a.x - b.x; | |  | c.y = a.y - b.y; | |  | c.z = a.z - b.z; | |  | return c; | |  | } | |  | }; | |  | int main() { | |  | setlocale(0, "rus"); | |  | Vector a(1, 4, 8); | |  | Vector b(1, 3, 7); | |  | Vector c; | |  | Vector d; | |  | c = sum(a, b); | |  | d = dif(a, b); | |  | cout << "Длинны :" << a.module() << " " << b.module() << endl; | |  | cout << "Сумма :"; | |  | c.Print(); | |  | cout << "Разность :"; | |  | d.Print(); | |  | system("pause"); | |  | return 0; | |  | } | |
|  |

**Выводы:**

В ходе выполнения Лабораторной работы №3 были использованы библиотеки iostream, string, сtime, cmath. В классах реализованы методы для присвоения значений полям и вывода этих значений, методы нахождения аргумента и модуля комплексного числа, методы суммы, разности, модуля векторов. Были достигнуты результаты:

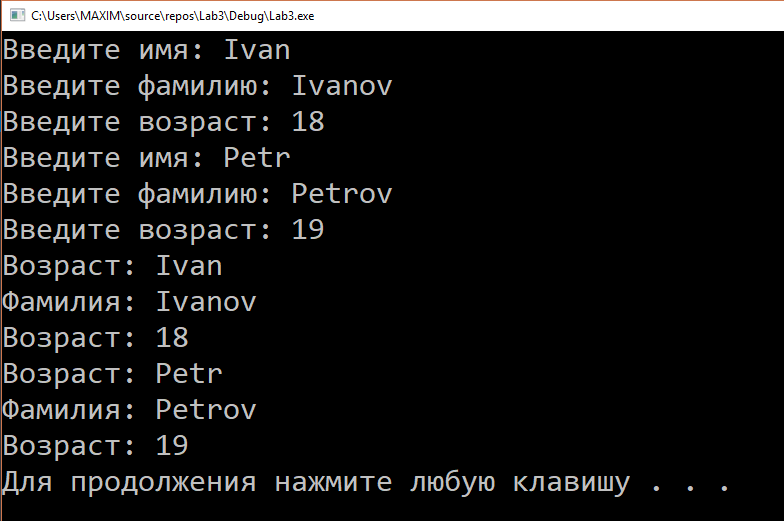


Рис. 5

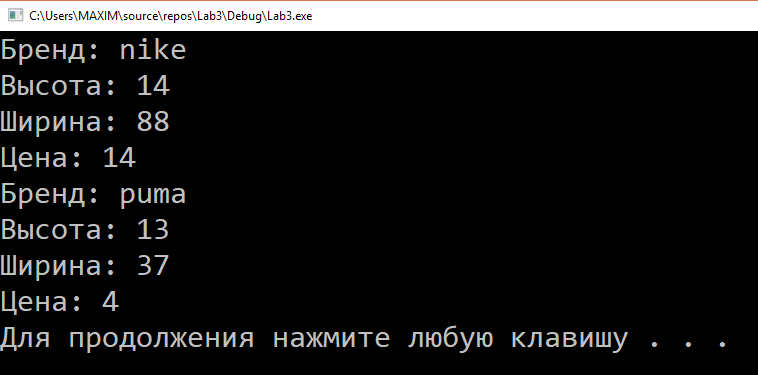


Рис. 6

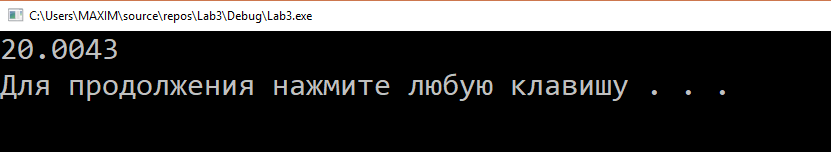


Рис. 7

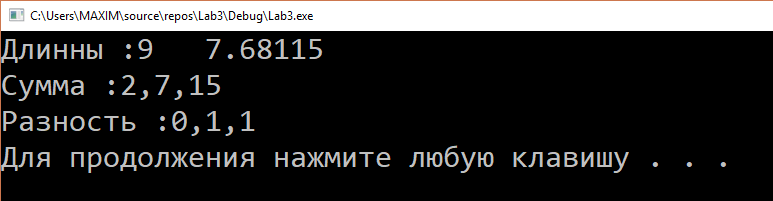


Рис. 8

<https://github.com/FunFuhrer/OOP/blob/master/Lab3.1.cpp>

<https://github.com/FunFuhrer/OOP/blob/master/Lab3.2.cpp>

<https://github.com/FunFuhrer/OOP/blob/master/Lab3.3.cpp>

<https://github.com/FunFuhrer/OOP/blob/master/Lab3.4.cpp>

# Лабораторная работа №4

**Задание:**

Из прошлой лабораторной работы дополнить всем видами

конструкторов и деструкторами классы:

1. Complex

2. Vector

3. Tiles

4. Child

**Ход выполнения:**

Вариант 1

|  |  |
| --- | --- |
|  | #include <iostream>  #include <string> |
|  | using namespace std; |
|  |  |
|  | class Children { |
|  | string firstName; |
|  | string lastName; |
|  | int years; |
|  | public: |
|  | Children(string firstName, string lastName, int years) { |
|  | cout << "Made " << this << endl; |
|  | this->firstName = firstName; |
|  | this->lastName = lastName; |
|  | this->years = years; |
|  | } |
|  | void set\_data() { |
|  | cout << "Введите имя: "; |
|  | cin >> firstName; |
|  | cout << "Введите фамилию: "; |
|  | cin >> lastName; |
|  | cout << "Введите возраст: "; |
|  | cin >> years; |
|  | } |
|  | void print\_data() { |
|  | cout << "имя: " << firstName << endl; |
|  | cout << "фамилия: " << lastName << endl; |
|  | cout << "возраст: " << years << endl; |
|  | } |
|  | Children(const Children &other) { |
|  | cout << "Copy " << this << endl; |
|  | this->firstName = other.firstName; |
|  | this->lastName = other.lastName; |
|  | this->years = other.years; |
|  | } |
|  |  |
|  | ~Children() { |
|  | cout << "Dekonstruktor " << this << endl; |
|  | } |
|  | }; |
|  | int main() { |
|  | setlocale(0, "rus"); |
|  | Children child1("a","b",0); |
|  | child1.set\_data(); |
|  | Children child2(child1); |
|  | child2.print\_data(); |
|  | system("pause"); |
|  | return 0; |
|  | } |

Вариант 2

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| |  | | --- | |  | |  | #include <iostream>  #include <string> | |  | using namespace std; | |  |  | |  | class Tiles { | |  | public: | |  | string brand; | |  | int size\_h, size\_w, price; | |  | Tiles(string brand, int size\_h, int size\_w, int price) { | |  | cout << "Made " << this << endl; | |  | this->brand = brand; | |  | this->size\_h = size\_h; | |  | this->size\_w = size\_w; | |  | this->price = price; | |  | } | |  | void getData() { | |  | cout << "Brend: " << brand << endl; | |  | cout << "H: " << size\_h << endl; | |  | cout << "W: " << size\_w << endl; | |  | cout << "price: " << price << endl; | |  | } | |  | Tiles(const Tiles &other) { | |  | cout << "Copy " << this << endl; | |  | this->brand = other.brand; | |  | this->size\_h = other.size\_h; | |  | this->size\_w = other.size\_w; | |  | this->price = other.price; | |  | } | |  | ~Tiles() { | |  | cout << "destruction " << this<<endl; | |  | } | |  | }; | |  | int main() { | |  | setlocale(0, "rus"); | |  | Tiles tile1("0",0,0,0); | |  | tile1.brand = "nike"; | |  | tile1.price = 14; | |  | tile1.size\_h = 14; | |  | tile1.size\_w = 88; | |  | Tiles tile2(tile1); | |  | tile2.getData(); | |  | system("pause"); | |  | return 0; | |  | } | |
|  |

Вариант 3

|  |  |
| --- | --- |
|  | #include <iostream>  #include <string> |
|  | using namespace std; |
|  |  |
|  | class Complex { |
|  | double re, im; |
|  | public: |
|  | Complex(double re, double im) |
|  | { |
|  | cout << "Made " << this << endl; |
|  | this->re = re; |
|  | this->im = im; |
|  | } |
|  | Complex(const Complex &other) { |
|  | cout << "Copy " << this << endl; |
|  | this->re = other.re; |
|  | this->im = other.im; |
|  | } |
|  | ~Complex() { |
|  | cout << "destruction " << this << endl; |
|  | } |
|  | void Print() |
|  | { |
|  | cout << re << "+" << im << "i" << endl; |
|  | } |
|  |  |
|  | double module() |
|  | { |
|  | return sqrt(this->re\*this->re + this->im \* this->im); |
|  | } |
|  |  |
|  | double getRe() |
|  | { |
|  | return this->re; |
|  | } |
|  |  |
|  | double getIm() |
|  | { |
|  | return this->im; |
|  | } |
|  | }; |
|  | int main() { |
|  | Complex comp(14.88, 13.37); |
|  | cout << comp.module() << endl; |
|  | Complex comp2(comp); |
|  | comp2.Print(); |
|  | system("pause"); |
|  | return 0; |
|  | } |

Вариант 4

|  |  |
| --- | --- |
|  | #include <iostream>  #include <string> |
|  | using namespace std; |
|  |  |
|  | class Vector { |
|  | double x, y, z; |
|  | public: |
|  | Vector(double x, double y, double z) |
|  | { |
|  | cout << "Made " << this << endl; |
|  | this->x = x; |
|  | this->y = y; |
|  | this->z = z; |
|  | } |
|  | Vector() |
|  | { |
|  | cout << "Made " << this << endl; |
|  | this->x = 0; |
|  | this->y = 0; |
|  | this->z = 0; |
|  | } |
|  | Vector(const Vector &other) |
|  | { |
|  | cout << "Copy " << this << endl; |
|  | this->x = other.x; |
|  | this->y = other.y; |
|  | this->z = other.z; |
|  | } |
|  | ~Vector() { |
|  | cout << "destruction " << this << endl; |
|  | } |
|  | void Print() |
|  | { |
|  | cout << x << "," << y << "," << z << endl; |
|  | } |
|  |  |
|  | double module() |
|  | { |
|  | return sqrt(x\*x + y \* y + z \* z); |
|  | } |
|  |  |
|  | friend Vector sum(Vector a, Vector b) |
|  | { |
|  | Vector c; |
|  | c.x = a.x + b.x; |
|  | c.y = a.y + b.y; |
|  | c.z = a.z + b.z; |
|  | return c; |
|  | } |
|  | friend Vector dif(Vector a, Vector b) |
|  | { |
|  | Vector c; |
|  | c.x = a.x - b.x; |
|  | c.y = a.y - b.y; |
|  | c.z = a.z - b.z; |
|  | return c; |
|  | } |
|  | }; |
|  | int main() { |
|  | setlocale(0, "rus"); |
|  | Vector a(1, 4, 8); |
|  | Vector b(1, 3, 7); |
|  | Vector c; |
|  | Vector d; |
|  | c = sum(a, b); |
|  | d = dif(a, b); |
|  | cout << "Modul' :" << a.module() << " " << b.module() << endl; |
|  | cout << "Sum :"; |
|  | c.Print(); |
|  | cout << "Dif :"; |
|  | d.Print(); |
|  | Vector m(a); |
|  | system("pause"); |
|  | return 0; |
|  | } |

**Выводы:**

В ходе выполнения Лабораторной работы №3 были использованы библиотеки iostream, string, сtime, cmath. В классах реализованы методы для присвоения значений полям и вывода этих значений, методы нахождения аргумента и модуля комплексного числа, методы суммы, разности, модуля векторов. Добавлены конструкторы и деструкторы. Были достигнуты результаты:

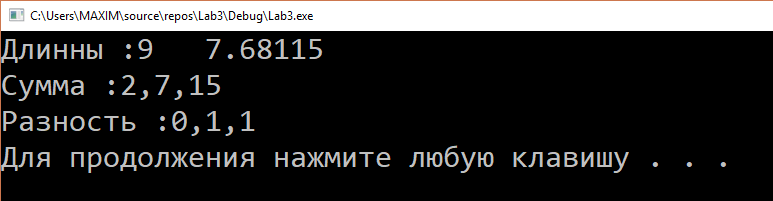


Рис. 9

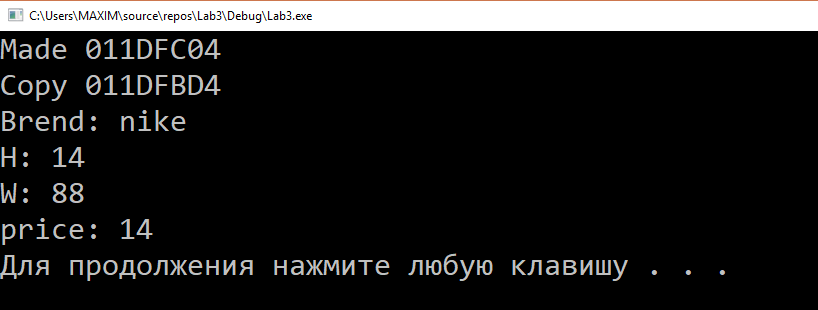


Рис. 10

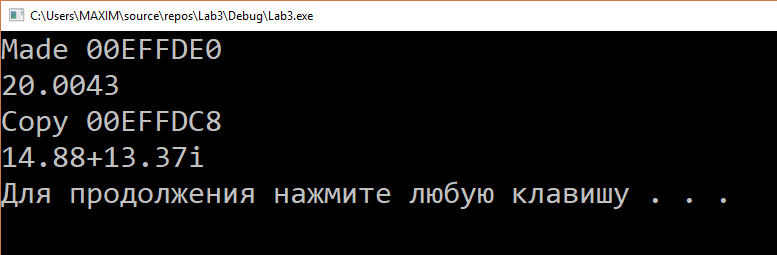


Рис. 11

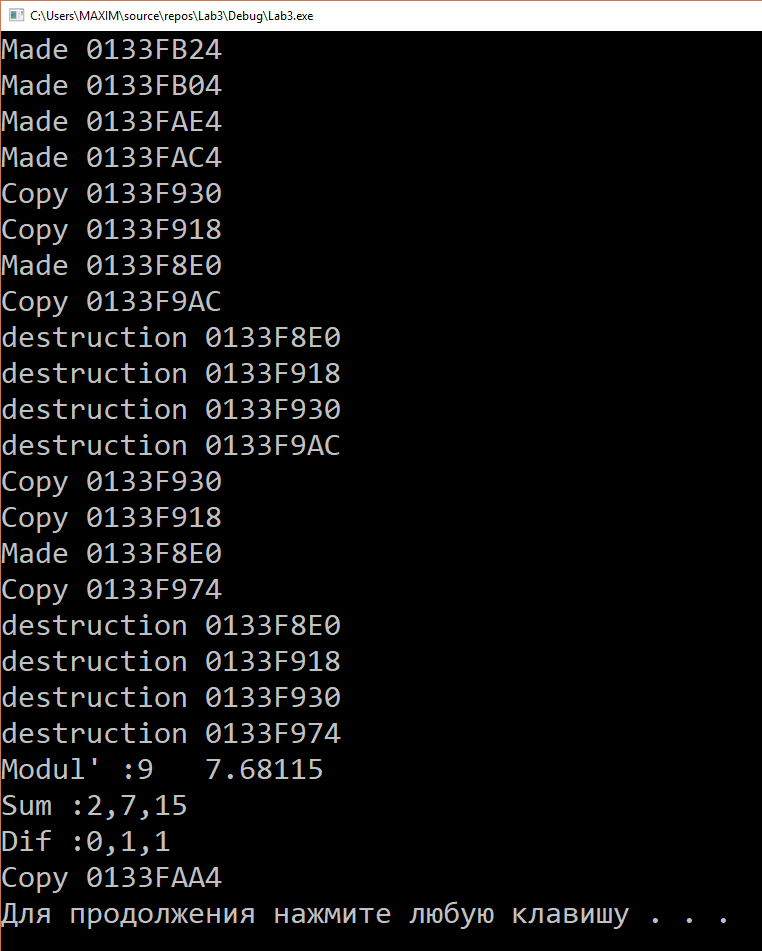


Рис. 12

<https://github.com/FunFuhrer/OOP/blob/master/Lab4.1.cpp>

<https://github.com/FunFuhrer/OOP/blob/master/Lab4.2.cpp>

<https://github.com/FunFuhrer/OOP/blob/master/Lab4.3.cpp>

<https://github.com/FunFuhrer/OOP/blob/master/Lab4.4.cpp>

# Лабораторная работа №5

**Задание:**

1. Для класса Complex перегрузить операторыприсваивания, инкремента, декремента, сравнения, ввода ивывода.

2. Для класса Vector перегрузить операторы присваивания, сравнения, ввода и вывода.

**Ход выполнения:**

Вариант 1

|  |
| --- |
| #include <iostream> |
|  | #include <string> |
|  | using namespace std; |
|  |  |
|  | class Complex { |
|  | double re, im; |
|  | public: |
|  | Complex(double re, double im) |
|  | { |
|  | cout << "Made " << this << endl; |
|  | this->re = re; |
|  | this->im = im; |
|  | } |
|  | Complex(const Complex &other) { |
|  | cout << "Copy " << this << endl; |
|  | this->re = other.re; |
|  | this->im = other.im; |
|  | } |
|  | ~Complex() { |
|  | cout << "destruction " << this << endl; |
|  | } |
|  | void Print() |
|  | { |
|  | cout << re << "+" << im << "i" << endl; |
|  | } |
|  |  |
|  | double module() |
|  | { |
|  | return sqrt(this->re\*this->re + this->im \* this->im); |
|  | } |
|  |  |
|  | double getRe() |
|  | { |
|  | return this->re; |
|  | } |
|  |  |
|  | double getIm() |
|  | { |
|  | return this->im; |
|  | } |
|  | void operator = (const Complex &other) { |
|  | this->re = other.re; |
|  | this->im = other.im; |
|  | } |
|  | void operator++() { |
|  | this->re++; |
|  | } |
|  | void operator--() { |
|  | this->re++; |
|  | } |
|  | bool operator == (const Complex &other) { |
|  | return ((this->re == other.re) && (this->im == other.im)); |
|  | } |
|  | bool operator > (const Complex &other) { |
|  | return ((this->re\*this->re + this->im\*this->im) >(other.re\*other.re + other.im\*other.im) ); |
|  | } |
|  | bool operator < (const Complex &other) { |
|  | return ((this->re\*this->re + this->im\*this->im) < (other.re\*other.re + other.im\*other.im)); |
|  | } |
|  | bool operator != (const Complex &other) { |
|  | return ((this->re != other.re) && (this->im != other.im)); |
|  | } |
|  |  |
|  | friend istream& operator>>(istream &in, Complex &other); |
|  | friend ostream& operator<<(ostream &out, Complex &other); |
|  | }; |
|  |  |
|  | istream& operator>>(istream &in, Complex &other) { |
|  | in >> other.re; |
|  | in >> other.im; |
|  | return in; |
|  | } |
|  | ostream& operator<<(ostream &out, Complex &other) { |
|  | out << other.re << "+" << other.im <<"i"<< endl; |
|  | return out; |
|  | } |
|  |  |
|  |  |
|  | int main() { |
|  | Complex comp(14.88, 13.37); |
|  | cout << comp.module() << endl; |
|  | Complex comp2(comp); |
|  | cout <<endl<< comp << endl; |
|  | system("pause"); |
|  | return 0; |
|  | } |

Вариант 2

|  |  |  |
| --- | --- | --- |
|  | |  |
|  | |  |
|  | |  |
|  | |  |
|  | |  |
|  | |  |
|  | |  |
|  | |  |
|  | |  |
|  | |  |
|  | |  |
|  | |  |
|  | |  |
|  | |  |
|  | |  |
|  | |  |
|  | |  |
|  | |  |
|  | |  |
|  | |  |
|  | |  |
|  | |  |
|  | |  |
|  | |  |
|  | |  |
|  | |  |
|  | |  |
|  | |  |
|  | |  |
|  | |  |
| #include <iostream> | |
|  | | #include <string> |
|  | | using namespace std; |
|  | |  |
|  | | class Vector { |
|  | | double x, y, z; |
|  | | public: |
|  | | Vector(double x, double y, double z) |
|  | | { |
|  | | cout << "Made " << this << endl; |
|  | | this->x = x; |
|  | | this->y = y; |
|  | | this->z = z; |
|  | | } |
|  | | Vector() |
|  | | { |
|  | | cout << "Made " << this << endl; |
|  | | this->x = 0; |
|  | | this->y = 0; |
|  | | this->z = 0; |
|  | | } |
|  | | Vector(const Vector &other) |
|  | | { |
|  | | cout << "Copy " << this << endl; |
|  | | this->x = other.x; |
|  | | this->y = other.y; |
|  | | this->z = other.z; |
|  | | } |
|  | | ~Vector() { |
|  | | cout << "destruction " << this << endl; |
|  | | } |
|  | | void Print() |
|  | | { |
|  | | cout << x << "," << y << "," << z << endl; |
|  | | } |
|  | |  |
|  | | double module() |
|  | | { |
|  | | return sqrt(x\*x + y \* y + z \* z); |
|  | | } |
|  | |  |
|  | | friend Vector sum(Vector a, Vector b) |
|  | | { |
|  | | Vector c; |
|  | | c.x = a.x + b.x; |
|  | | c.y = a.y + b.y; |
|  | | c.z = a.z + b.z; |
|  | | return c; |
|  | | } |
|  | | friend Vector dif(Vector a, Vector b) |
|  | | { |
|  | | Vector c; |
|  | | c.x = a.x - b.x; |
|  | | c.y = a.y - b.y; |
|  | | c.z = a.z - b.z; |
|  | | return c; |
|  | | } |
|  | | void operator =(const Vector &other) { |
|  | | this->x = other.x; |
|  | | this->y = other.y; |
|  | | this->z = other.z; |
|  | | } |
|  | | bool operator == (const Vector &other) { |
|  | | return ((this->x == other.x) && (this->y == other.y) && (this->y == other.y)); |
|  | | } |
|  | | bool operator > (const Vector &other) { |
|  | | return ((this->x\*this->x + this->y\*this->y+this->z\*this->z) > (other.x\*other.x + other.y\*other.y+ other.z\*other.z)); |
|  | | } |
|  | | bool operator < (const Vector &other) { |
|  | | return ((this->x\*this->x + this->y\*this->y + this->z\*this->z) < (other.x\*other.x + other.y\*other.y + other.z\*other.z)); |
|  | | } |
|  | | bool operator != (const Vector &other) { |
|  | | return ((this->x != other.x) && (this->y != other.y) && (this->z != other.z)); |
|  | | } |
|  | | friend istream& operator>>(istream &in, Vector &other); |
|  | | friend ostream& operator<<(ostream &out, Vector &other); |
|  | | }; |
|  | |  |
|  | | istream& operator>>(istream &in, Vector &other) { |
|  | | in >> other.x; |
|  | | in >> other.y; |
|  | | in >> other.z; |
|  | | return in; |
|  | | } |
|  | | ostream& operator<<(ostream &out, Vector &other) { |
|  | | out << other.x << " " << other.y << " " << other.z << endl; |
|  | | return out; |
|  | | } |
|  | | int main() { |
|  | | setlocale(0, "rus"); |
|  | | Vector a(1, 4, 8); |
|  | | Vector b(1, 3, 7); |
|  | | Vector c; |
|  | | Vector d; |
|  | | c = sum(a, b); |
|  | | d = dif(a, b); |
|  | | cout << "Modul' :" << a.module() << " " << b.module() << endl; |
|  | | cout << "Sum :"; |
|  | | c.Print(); |
|  | | cout << "Dif :"; |
|  | | d.Print(); |
|  | | Vector m(a); |
|  | | system("pause"); |
|  | | return 0; |
|  | | } |

**Выводы:**

В ходе выполнения Лабораторной работы №5 были использованы библиотеки iostream, cmath. Перегружены требуемые операторы. Были достигнуты результаты:

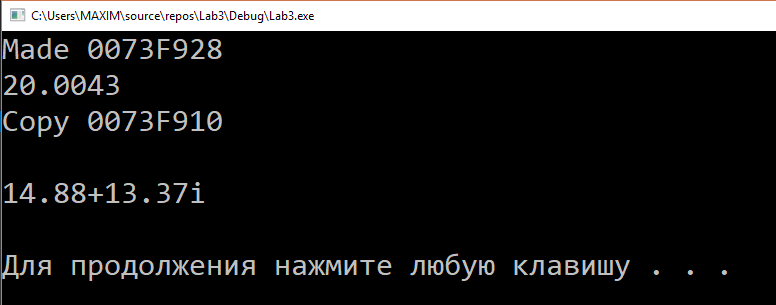


Рис. 13

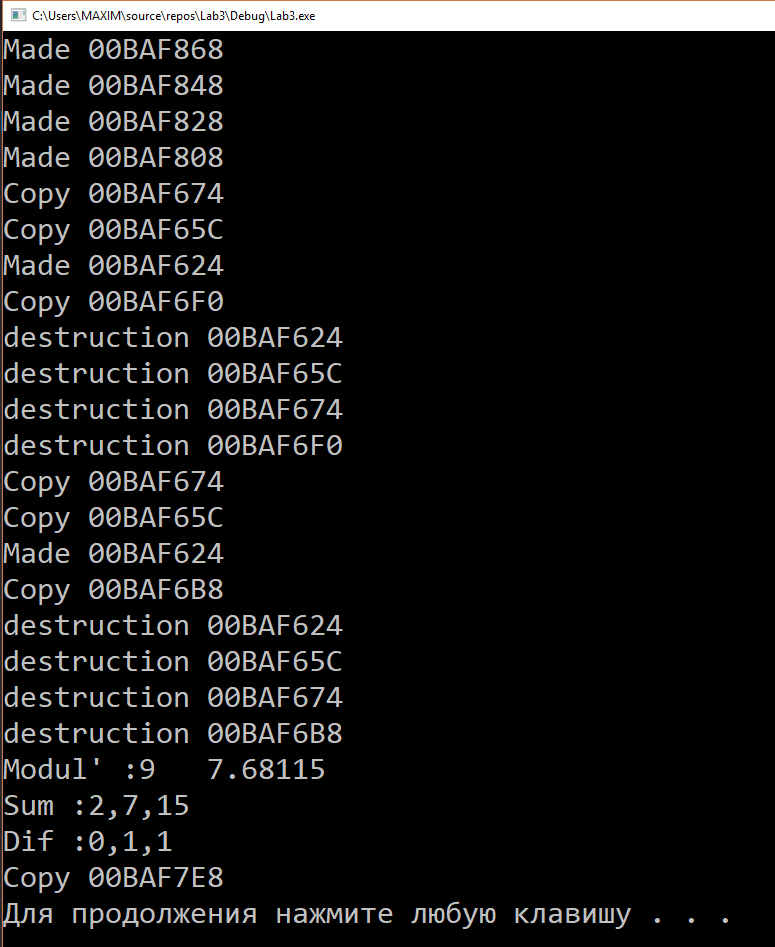


Рис. 14

<https://github.com/FunFuhrer/OOP/blob/master/Lab5.1.cpp>

<https://github.com/FunFuhrer/OOP/blob/master/Lab5.2.cpp>

# Лабораторная работа №6

**Задание:**

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 выполнения:**

Вариант 1

|  |
| --- |
| #include <iostream> |
|  | #include <string> |
|  | using namespace std; |
|  |  |
|  | class Student |
|  | { |
|  | public: |
|  | void setname(string name) { |
|  | this->name = name; |
|  | } |
|  | void setgroup(string group) { |
|  | this->group = group; |
|  | } |
|  | void setscores(int scores[5]) { |
|  | for (int i = 0; i < 5; i++) |
|  | { |
|  | this->scores[i] = scores[i]; |
|  | } |
|  | } |
|  | string getname() { |
|  | return name; |
|  | } |
|  | string getgroup() { |
|  | return group; |
|  | } |
|  | protected: |
|  | string name, group; |
|  | float averagescore; |
|  | int scores[5]; |
|  | }; |
|  |  |
|  | class Staryj:public Student |
|  | { |
|  | public: |
|  | void a\_pochemu\_ty\_ne\_na\_pare(Staryj b ,Student a) { |
|  | cout << b.getname() << ": Ty gde, " <<a.getname() <<"???"<<endl; |
|  | } |
|  |  |
|  | private: |
|  | string gmail; |
|  | }; |
|  |  |
|  | int main() |
|  | { |
|  | Student first; |
|  | Staryj second; |
|  | first.setname("Ivan"); |
|  | second.setname("Dima"); |
|  | second.a\_pochemu\_ty\_ne\_na\_pare(second,first); |
|  | system("pause"); |
|  | return 0; |
|  | } |

Вариант 2

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| |  | | --- | | #include <iostream> | |  | #include <string> | |  | using namespace std; | |  |  | |  | class Alive { | |  | protected: | |  | string name; | |  | int year; | |  | public: | |  | Alive() { | |  | name = ""; | |  | year = 0; | |  | } | |  | Alive(string name, int year){ | |  | this->name = name; | |  | this->year = year; | |  | } | |  | }; | |  |  | |  | class Bird :public Alive { | |  | int countWings; | |  | public: | |  | Bird(int countWings){ | |  | this->countWings = countWings; | |  | } | |  | }; | |  |  | |  | class Fish :public Alive { | |  | string type; | |  | public: | |  | Fish(string type) { | |  | this->type = type; | |  | } | |  | }; | |  |  | |  | class Animal :public Alive { | |  | string typeAnimal; | |  | public: | |  | Animal() { | |  | typeAnimal = ""; | |  | } | |  | Animal(string typeAnimal){ | |  | this->typeAnimal = typeAnimal; | |  | } | |  | }; | |  |  | |  | int main() { | |  | Animal a("cat"); | |  | system("pause"); | |  | return 0; | |  | } | |
|  |

Вариант 3

|  |
| --- |
| #include <iostream> |
|  | #include <string> |
|  | using namespace std; |
|  |  |
|  | class Animal { |
|  | protected: |
|  | string name; |
|  | int year; |
|  | public: |
|  | Animal() { |
|  | name = ""; |
|  | year = 0; |
|  | } |
|  | Animal(string name,int year) { |
|  | this->name = name; |
|  | this->year = year; |
|  | } |
|  | Animal getdata() { |
|  | cout << name << year; |
|  | } |
|  | }; |
|  |  |
|  | class Dog :public Animal { |
|  | string breed; |
|  | public: |
|  | Dog(string breed) { |
|  | this->breed = breed; |
|  | } |
|  | }; |
|  |  |
|  | class Cat :public Animal { |
|  | string color; |
|  | public: |
|  | Cat(string color) { |
|  | this->color = color; |
|  | } |
|  |  |
|  | }; |
|  |  |
|  | int main() { |
|  | system("pause"); |
|  | return 0; |
|  | } |

**Выводы:**

В ходе выполнения Лабораторной работы №6 были использованы библиотека iostream, vector, string. Были реализованы методы, демонстрирующие работоспоссобность классов, перегружен оператор вывода. Были достигнуты результаты:

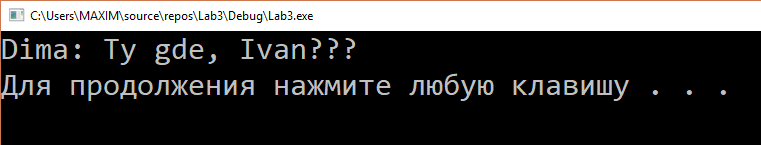


Рис. 15

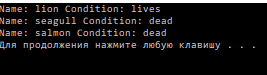


Рис. 16

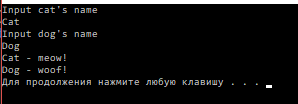


Рис. 17

<https://github.com/FunFuhrer/OOP/blob/master/Lab6.1.cpp>

<https://github.com/FunFuhrer/OOP/blob/master/Lab6.2.cpp>

<https://github.com/FunFuhrer/OOP/blob/master/Lab6.3.cpp>

# Лабораторная работа №7

**Задание:**

Создать базовый абстрактных класс «Человек», имеющий нереализованную виртуальную функцию вывода информации на экран. Затем создать классы «Ученик» и «Босс», унаследованные от него.

**Ход выполнения:**

|  |
| --- |
| #include <iostream> |
|  | #include<string> |
|  | using namespace std; |
|  | class Human { |
|  | protected: |
|  | string surname, name, midname; |
|  | int age; |
|  | public: |
|  | Human() { |
|  | surname = ""; |
|  | name = ""; |
|  | midname = ""; |
|  | age = 0; |
|  | }; |
|  | Human(string surname, string name, string midname, int age) { |
|  | this->surname = surname; |
|  | this->name = name; |
|  | this->midname = midname; |
|  | this->age = age; |
|  | } |
|  | virtual void print() { |
|  | cout << surname << " " << name << " " << midname << " " << age << " годиков"<<endl; |
|  | } |
|  | ~Human() {} |
|  | }; |
|  |  |
|  | class Student :public Human { |
|  | bool on\_lesson; |
|  | public: |
|  | Student() { |
|  | on\_lesson = false; |
|  | } |
|  | Student(string surname, string name, string midname, int age, bool on\_lesson) : Human(surname, name, midname, age) { |
|  | this->surname = surname; |
|  | this->name = name; |
|  | this->midname = midname; |
|  | this->age = age; |
|  | this->on\_lesson = on\_lesson; |
|  | } |
|  | void print()override |
|  | { |
|  | cout << surname << " " << name << " " << midname << " " << age << " годиков "; |
|  | if (on\_lesson)cout << "на парах" << endl; |
|  | else cout << "НЕ на парах" << endl; |
|  | } |
|  | ~Student() {}; |
|  | }; |
|  |  |
|  | class Boss :public Human { |
|  | int number\_of\_workers; |
|  | public: |
|  | Boss() { |
|  | number\_of\_workers = 0; |
|  | } |
|  | Boss(string surname, string name, string midname, int age, int number\_of\_workers) : Human(surname, name, midname, age) { |
|  | this->surname = surname; |
|  | this->name = name; |
|  | this->midname = midname; |
|  | this->age = age; |
|  | this->number\_of\_workers = number\_of\_workers; |
|  | } |
|  | void print()override |
|  | { |
|  | cout << surname << " " << name << " " << midname << " " << age << " годиков "<< number\_of\_workers<<" работников"<<endl; |
|  | } |
|  | ~Boss() {}; |
|  | }; |
|  |  |
|  |  |
|  | int main() { |
|  | setlocale(LC\_ALL, "ru"); |
|  | Student student1("Иванов","Иван","Иванович",18,false); |
|  | Boss boss1("Петров", "Петр", "Петрович", 40, 30); |
|  | Human human1("Сидоров", "Сидр", "Сидорович", 27); |
|  | Human \*date1 = &student1; |
|  | Human \*date2 = &boss1; |
|  | Human \*date3 = &human1; |
|  | date1->print(); |
|  | date2->print(); |
|  | date3->print(); |
|  | system("pause"); |
|  | return 0; |
|  | } |

**Выводы:**

В ходе выполнения Лабораторной работы №7 были использованы библиотеки iostream, string. Была реализованы методы для демонстрации работоспособности производных классов. Были достигнуты результаты:

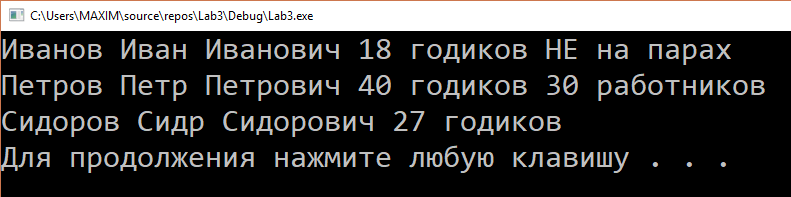


Рис. 18

<https://github.com/FunFuhrer/OOP/blob/master/Lab7.cpp>

# Лабораторная работа №8

**Задание:**

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 выполнения:**

Вариант 1

|  |
| --- |
| #include <iostream> |
|  | #include<string> |
|  | #include<vector> |
|  | using namespace std; |
|  |  |
|  | class Student { |
|  | bool on\_lesson; |
|  | string surname, name, midname; |
|  | int age; |
|  | public: |
|  | Student() { |
|  | on\_lesson = false; |
|  | } |
|  | Student(string surname, string name, string midname, int age, bool on\_lesson) { |
|  | this->surname = surname; |
|  | this->name = name; |
|  | this->midname = midname; |
|  | this->age = age; |
|  | this->on\_lesson = on\_lesson; |
|  | } |
|  | void print() |
|  | { |
|  | cout << surname << " " << name << " " << midname << " " << age << " godikov "; |
|  | if (on\_lesson)cout << "na parah" << endl; |
|  | else cout << "NE!!! NA PARAH" << endl; |
|  | } |
|  | ~Student() {}; |
|  | }; |
|  | int main() { |
|  | vector<Student> student; |
|  | Student studentik1("Ivanov", "Ivan", "Ivanovich", 22, false); |
|  | Student studentik2("Petrov", "Petr", "Petrovich", 18, true); |
|  | student.push\_back(studentik1); |
|  | student.push\_back(studentik2); |
|  | student[0].print(); |
|  | student[1].print(); |
|  | system("pause"); |
|  | return 0; |
|  | } |

Вариант 2

|  |
| --- |
| #include <iostream> |
|  | #include <string> |
|  | #include<list> |
|  | using namespace std; |
|  |  |
|  | class Complex { |
|  | double re, im; |
|  | public: |
|  | Complex(double re, double im) |
|  | { |
|  | this->re = re; |
|  | this->im = im; |
|  | } |
|  | Complex(const Complex &other) { |
|  | this->re = other.re; |
|  | this->im = other.im; |
|  | } |
|  | ~Complex() { |
|  | } |
|  | void Print() |
|  | { |
|  | cout << re << "+" << im << "i" << endl; |
|  | } |
|  |  |
|  | double module() |
|  | { |
|  | return sqrt(this->re\*this->re + this->im \* this->im); |
|  | } |
|  |  |
|  | double getRe() |
|  | { |
|  | return this->re; |
|  | } |
|  |  |
|  | double getIm() |
|  | { |
|  | return this->im; |
|  | } |
|  | void operator = (const Complex &other) { |
|  | this->re = other.re; |
|  | this->im = other.im; |
|  | } |
|  | void operator++() { |
|  | this->re++; |
|  | } |
|  | void operator--() { |
|  | this->re++; |
|  | } |
|  | bool operator == (const Complex &other) { |
|  | return ((this->re == other.re) && (this->im == other.im)); |
|  | } |
|  | bool operator > (const Complex &other) { |
|  | return ((this->re\*this->re + this->im\*this->im) > (other.re\*other.re + other.im\*other.im)); |
|  | } |
|  | bool operator < (const Complex &other) { |
|  | return ((this->re\*this->re + this->im\*this->im) < (other.re\*other.re + other.im\*other.im)); |
|  | } |
|  | bool operator != (const Complex &other) { |
|  | return ((this->re != other.re) && (this->im != other.im)); |
|  | } |
|  |  |
|  | friend istream& operator>>(istream &in, Complex &other); |
|  | friend ostream& operator<<(ostream &out, Complex &other); |
|  | }; |
|  |  |
|  | istream& operator>>(istream &in, Complex &other) { |
|  | in >> other.re; |
|  | in >> other.im; |
|  | return in; |
|  | } |
|  | ostream& operator<<(ostream &out, Complex &other) { |
|  | out << other.re << "+" << other.im << "i" << endl; |
|  | return out; |
|  | } |
|  |  |
|  | /\*void PrintList(const list<Complex> &lst) { |
|  | for (auto i=lst.cbegin() ; i!=lst.cend() ; ++i) |
|  | { |
|  | cout << \*i <<endl; |
|  | } |
|  |  |
|  | }\*/ |
|  |  |
|  | int main() { |
|  | Complex comp1(14.88, 13.37); |
|  | Complex comp2(14,4 ); |
|  | Complex comp3(27,17.5 ); |
|  | list<Complex> complex; |
|  | complex.push\_back(comp1); |
|  | complex.push\_back(comp2); |
|  | complex.push\_front(comp3); |
|  |  |
|  | cout << complex.back().getRe() << "+" << complex.back().getIm() <<"i"<< endl; |
|  | complex.pop\_back(); |
|  |  |
|  | cout << complex.back().getRe() << "+" << complex.back().getIm() << "i" << endl; |
|  | complex.pop\_back(); |
|  |  |
|  | cout << complex.back().getRe() << "+" << complex.back().getIm() << "i" << endl; |
|  |  |
|  | system("pause"); |
|  | return 0; |
|  | } |

Вариант 3

|  |
| --- |
| #include <iostream> |
|  | #include <string> |
|  | #include<queue> |
|  | using namespace std; |
|  |  |
|  | class Auto { |
|  | string brand, model; |
|  | public: |
|  | Auto() { |
|  | brand = ""; |
|  | model = ""; |
|  | } |
|  | Auto(string brand, string model) { |
|  | this->brand = brand; |
|  | this->model = model; |
|  | } |
|  | void Print() { |
|  | cout << "Brand : " << brand << " Model : " << model << endl; |
|  | } |
|  | }; |
|  |  |
|  | int main() { |
|  | queue<Auto> automobiles; |
|  | Auto first("Nissan", "Grand Cherokee"), second("Skoda", "Octavia"); |
|  | automobiles.push(first); |
|  | automobiles.push(second); |
|  | automobiles.back().Print(); |
|  | automobiles.front().Print(); |
|  |  |
|  | system("pause"); |
|  | return 0; |
|  | } |

**Выводы:**

В ходе выполнения Лабораторной работы №8 были использованы библиотеки iostream,string,vector,list,iterator, queue, cmath и ранее реализованные классы. Были достигнуты результаты:

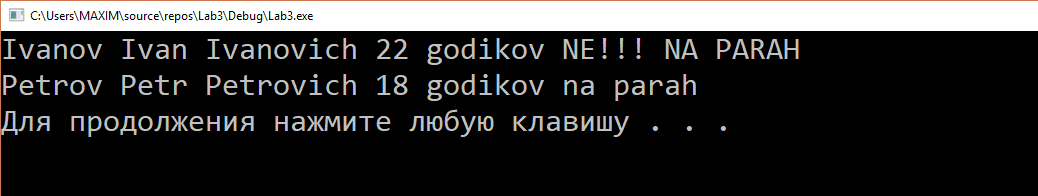


Рис. 19

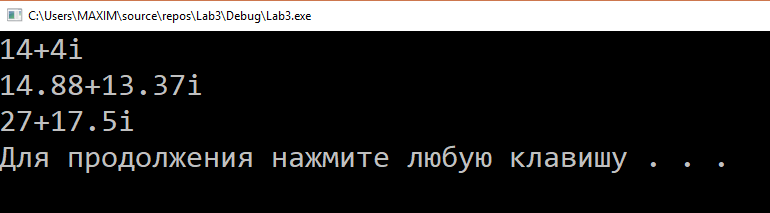


Рис. 20

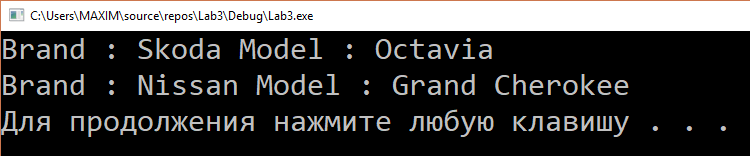


Рис. 21

<https://github.com/FunFuhrer/OOP/blob/master/Lab8.1.cpp>

<https://github.com/FunFuhrer/OOP/blob/master/Lab8.2.cpp>

<https://github.com/FunFuhrer/OOP/blob/master/Lab8.3.cpp>