|  |  |
| --- | --- |
| 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 | Автушко Д.А. |
| Принял ассистент кафедры | Хлебникова В.Л. |

|  |  |  |
| --- | --- | --- |
| Лабораторные работы выполнены | «23» февраля 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 |  |
| 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**

Файл заголовков Complex.h:

|  |
| --- |
| class Complex{ |
|  | double Re\_,Im\_; |
|  | public: |
|  | Complex(double Re = 0,double Im = 0); |
|  | double printRe(); |
|  | double printIm(); |
|  | double getRe(); |
|  | double getIm(); |
|  | double module(); |
|  | Complex operator+(Complex b); |
|  | Complex operator-(Complex b); |
|  | Complex operator\*(Complex b); |
|  | Complex operator/(Complex b); |
|  | Complex pow(int p); |
|  | }; |

Файл Complex.cpp:

|  |
| --- |
|  |
| |  | | --- | | #include "Complex.h" | |  | #include <cmath> | |  | Complex::Complex(double Re,double Im){ | |  | Re\_ = Re; | |  | Im\_ = Im; | |  | } | |  | double Complex::printRe(){ | |  | return Re\_; | |  | } | |  | double Complex::printIm(){ | |  | return Im\_; | |  | } | |  | double Complex::getRe(){ | |  | return this->Re\_; | |  | } | |  | double Complex::getIm(){ | |  | return this->Im\_; | |  | } | |  | double Complex::module(){ | |  | return sqrt(this->Re\_\*this->Re\_ + this->Im\_ \* this->Im\_); | |  | } | |  | Complex Complex::operator+(Complex b){ | |  | Complex res; | |  | res.Re\_ = this->Re\_ + b.Re\_; | |  | res.Im\_ = this->Im\_ + b.Im\_; | |  | return res; | |  | } | |  | Complex Complex::operator-(Complex b){ | |  | Complex res; | |  | res.Re\_ = this->Re\_ - b.Re\_; | |  | res.Im\_ = this->Im\_ - b.Im\_; | |  | return res; | |  | } | |  | Complex Complex::operator\*(Complex b){ | |  | Complex res; | |  | res.Re\_ = this->Re\_ \* b.Re\_ - this->Im\_ \* b.Im\_; | |  | res.Im\_ = this->Re\_ \* b.Im\_ + this->Im\_ \* b.Re\_; | |  | return res; | |  | } | |  | Complex Complex::operator/(Complex b){ | |  | Complex res; | |  | res.Re\_ = (this->Re\_ \* b.Re\_ + this->Im\_ \* b.Im\_)/(b.Re\_ \* b.Re\_ + b.Im\_ \* b.Im\_); | |  | res.Im\_ = (b.Re\_ \* this->Im\_ - this->Re\_ \* b.Im\_)/(b.Re\_ \* b.Re\_ + b.Im\_ \* b.Im\_); | |  | return res; | |  | } | |  | Complex Complex::pow(int p){ | |  | Complex res; | |  | res = \*this; | |  | while (p > 1) { | |  | res = res \* \*this; | |  | p--; | |  | } | |  | return res; | |  | } | |
|  |
|  |
|  |
|  |
|  |
|  |
|  |
|  |
|  |
|  |
|  |
|  |
|  |
|  |
|  |
|  |
|  |
|  |
|  |
|  |
|  |
|  |
|  |
|  |
|  |
|  |
|  |
|  |
|  |
|  |
|  |
|  |
|  |
|  |
|  |
|  |
|  |
|  |
|  |
|  |
|  |
|  |
|  |
|  |

Файл main.cpp:

|  |
| --- |
| #include <iostream> |
|  | #include "Complex.h" |
|  | #include <vector> |
|  | #include <fstream> |
|  | using namespace std; |
|  |  |
|  | int main(int argc, const char \* argv[]) { |
|  | ifstream file("/Users/daniilavtusko/Desktop/coding/Lab1OOP/Lab1/Lab1/TestCompl.txt"); |
|  | int n,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.printRe() << ' ' << max.printIm() << endl; |
|  | return 0; |
|  | } |

**Вывод**

Все поставленные задачи были выполнены. Для достижения требуемого результата был создан класс, разбитый на файлы с реализацией и объявлением. Также были использованы некоторые дерективы препрцессора для защиты от многократного подключения файлов и перегружены некоторые операторы. Результаты работы программ приведены на скриншотах (рис. 1, 2). Исходный код также доступен по ссылкам на GitHub:

1. hhttps://github.com/adanil/OOP/tree/master/Lab1

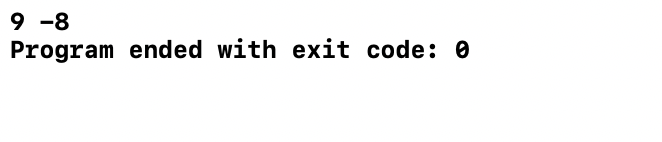
****

Рис. 1 Результат работы 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.

**Ход работы**

1. Программа №1

|  |
| --- |
|  |
| #include <iostream> |
|  | #include <vector> |
|  | #include <ctime> |
|  | #include <algorithm> |
|  | using namespace std; |
|  |  |
|  |  |
|  | int sum (int a,int b){ |
|  | return a + b; |
|  | } |
|  |  |
|  | int multiply(int a,int b){ |
|  | return a\*b; |
|  | } |
|  |  |
|  | int divis(int a,int b){ |
|  | return a/b; |
|  | } |
|  |  |
|  | int diff(int a,int b){ |
|  | return a-b; |
|  | } |
|  |  |
|  | void reverseArray(vector<int> &arr){ |
|  | int t; |
|  | for (int i = 0;i < 5;i++){ |
|  | t = arr[i]; |
|  | arr[i] = arr[9-i]; |
|  | arr[9-i] = t; |
|  | } |
|  | } |
|  |  |
|  | void neubArray(vector<int> &arr){ |
|  | int t; |
|  | for (int i = 0;i < 10;i++) |
|  | for (int j = i;j < 10;j++) |
|  | if (arr[j] < arr[i]){ |
|  | t = arr[j]; |
|  | arr[j] = arr[i]; |
|  | arr[i] = t; |
|  | } |
|  | } |
|  |  |
|  | void ubArray(vector<int> &arr){ |
|  | int t; |
|  | for (int i = 0;i < 10;i++) |
|  | for (int j = i;j < 10;j++) |
|  | if (arr[j] > arr[i]){ |
|  | t = arr[j]; |
|  | arr[j] = arr[i]; |
|  | arr[i] = t; |
|  | } |
|  | } |
|  |  |
|  | void (\*arrayFunc(vector<int> &arr))(vector<int> &arr){ |
|  | int sum = 0; |
|  | for (int i = 0;i < 10;i++) |
|  | sum += arr[i]; |
|  | if (sum == arr[0]) |
|  | return reverseArray; |
|  | else if (sum > arr[0]) |
|  | return neubArray; |
|  | else |
|  | return ubArray; |
|  | } |
|  |  |
|  |  |
|  | int (\*func)(int a,int b); |
|  | int main(int argc, const char \* argv[]) { |
|  | int a,b,action; |
|  | cout << "Enter a: "; |
|  | cin >> a; |
|  | cout << "Enter b: "; |
|  | cin >> b; |
|  | cout << "Enter 1 - sum of a and b \n 2 - different of a and b \n 3 - multiply of a and b \n 4 - division of a and b" << endl; |
|  | cin >> action; |
|  | switch (action) { |
|  | case 1: |
|  | func = sum; |
|  | break; |
|  | case 2: |
|  | func = diff; |
|  | break; |
|  | case 3: |
|  | func = multiply; |
|  | break; |
|  | case 4: |
|  | func = divis; |
|  | break; |
|  | } |
|  | cout << "Result is " << func(a,b) << endl; |
|  |  |
|  | //Вариант 2 |
|  | vector<int> arr; |
|  | srand(time(0)); |
|  | cout << "Created array" << endl; |
|  | for (int i = 0; i < 10;i++){ |
|  | arr.push\_back(rand()%20); |
|  | cout << arr[i] << ' '; |
|  | } |
|  | cout << endl; |
|  |  |
|  | arrayFunc(arr)(arr); |
|  | cout << "Array after function" << endl; |
|  | for (int i = 0;i < 10;i++) |
|  | cout << arr[i] << ' '; |
|  |  |
|  | cout << endl; |
|  | return 0; |
|  | } |

**Вывод**

Все поставленные задачи были выполнены. Для достижения поставленных целей были использованы указатели на функции с различными возвращаемыми значениями. Результаты работы программ можно увидеть на скриншотах (рис. 3, 4). Исходный код программ также доступен по ссылкам на GitHub:

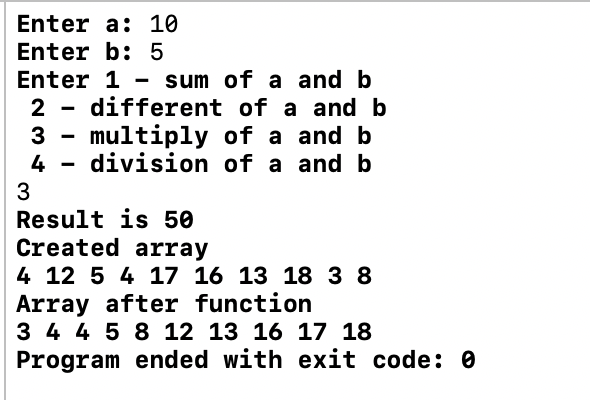
https://github.com/adanil/OOP/tree/master/Lab2

Рис. 2 Результат работы 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 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**

|  |
| --- |
| #include <iostream> |
|  | #include <string> |
|  | #include <cmath> |
|  | using namespace std; |
|  |  |
|  | class Child{ |
|  | string firstName; |
|  | string lastName; |
|  | int years; |
|  | public: |
|  | void enterData(){ |
|  | cout << "Enter firstName: "; |
|  | cin >> firstName; |
|  | cout << "Enter lastName: "; |
|  | cin >> lastName; |
|  | cout << "Enter years: "; |
|  | cin >> years; |
|  | } |
|  | void printData(){ |
|  | cout << "firstName: " << firstName << endl; |
|  | cout << "lastName: " << lastName << endl; |
|  | cout << "years: " << years << endl; |
|  | } |
|  | }; |
|  |  |
|  |  |
|  | class Tiles{ |
|  | public: |
|  | string brand; |
|  | int size\_h,size\_w,price; |
|  | void getData(){ |
|  | cout << "Brand: " << brand << endl; |
|  | cout << "Size\_h: " << size\_h << endl; |
|  | cout << "Size\_w: " << size\_w << endl; |
|  | cout << "Price: " << price << endl; |
|  | } |
|  | }; |
|  |  |
|  | class Complex{ |
|  | int Re\_,Im\_; |
|  | public: |
|  | Complex(double Re = 0,double Im = 0){ |
|  | Re\_ = Re; |
|  | Im\_ = Im; |
|  | } |
|  | double printRe(){ |
|  | return Re\_; |
|  | } |
|  | double printIm(){ |
|  | return Im\_; |
|  | } |
|  | double getRe(){ |
|  | return this->Re\_; |
|  | } |
|  | double getIm(){ |
|  | return this->Im\_; |
|  | } |
|  | double module(){ |
|  | return sqrt(this->Re\_\*this->Re\_ + this->Im\_ \* this->Im\_); |
|  | } |
|  | Complex operator+(Complex b){ |
|  | Complex res; |
|  | res.Re\_ = this->Re\_ + b.Re\_; |
|  | res.Im\_ = this->Im\_ + b.Im\_; |
|  | return res; |
|  | } |
|  | Complex operator-(Complex b){ |
|  | Complex res; |
|  | res.Re\_ = this->Re\_ - b.Re\_; |
|  | res.Im\_ = this->Im\_ - b.Im\_; |
|  | return res; |
|  | } |
|  | Complex operator\*(Complex b){ |
|  | Complex res; |
|  | res.Re\_ = this->Re\_ \* b.Re\_ - this->Im\_ \* b.Im\_; |
|  | res.Im\_ = this->Re\_ \* b.Im\_ + this->Im\_ \* b.Re\_; |
|  | return res; |
|  | } |
|  | Complex operator/(Complex b){ |
|  | Complex res; |
|  | res.Re\_ = (this->Re\_ \* b.Re\_ + this->Im\_ \* b.Im\_)/(b.Re\_ \* b.Re\_ + b.Im\_ \* b.Im\_); |
|  | res.Im\_ = (b.Re\_ \* this->Im\_ - this->Re\_ \* b.Im\_)/(b.Re\_ \* b.Re\_ + b.Im\_ \* b.Im\_); |
|  | return res; |
|  | } |
|  | Complex pow(int p){ |
|  | Complex res; |
|  | res = \*this; |
|  | while (p > 1) { |
|  | res = res \* \*this; |
|  | p--; |
|  | } |
|  | return res; |
|  | } |
|  | }; |
|  |  |
|  | class Vctr{ |
|  | public: |
|  | int x,y; |
|  | double module(){ |
|  | return sqrt(x\*x + y\*y); |
|  | } |
|  | Vctr sum(Vctr b){ |
|  | Vctr res; |
|  | res.x = this->x + b.x; |
|  | res.y = this->y + b.y; |
|  | return res; |
|  | } |
|  | Vctr diff(Vctr b){ |
|  | Vctr res; |
|  | res.x = this->x - b.x; |
|  | res.y = this->y - b.y; |
|  | return res; |
|  | } |
|  | }; |
|  |  |
|  | int main(int argc, const char \* argv[]) { |
|  | Child user1,user2; |
|  | user1.enterData(); |
|  | user1.printData(); |
|  | user2.enterData(); |
|  | user2.printData(); |
|  |  |
|  | //Вариант 2 |
|  |  |
|  | Tiles potato1,potato2; |
|  | potato1.brand = "Ferm"; potato1.size\_h = 15; potato1.size\_w = 10; potato1.price = 30; |
|  | potato1.getData(); |
|  | potato2.brand = "Organic"; potato2.size\_h = 5; potato2.size\_w = 6; potato2.price = 70; |
|  | potato2.getData(); |
|  |  |
|  | //Вариант 3 |
|  | Complex cmplx(5,8); |
|  | cout << cmplx.module() << endl; |
|  |  |
|  | //Вариант 4 |
|  | Vctr a,b; |
|  | a.x = 5; b.x = 3; |
|  | a.y = 7; b.y = 4; |
|  | cout << "Modul a: " <<a.module() << endl; |
|  | cout << "Sum of a and b: " << a.sum(b).x << ' ' << a.sum(b).y << endl; |
|  | return 0; |
|  | } |

**Вывод**

Все поставленные задачи были достигнуты. Для получения требуемого результата былииспользованы конструкции класса и перегрузка некоторых операторов. Результаты работы программ изображены на скриншотах (рис. 5-8). Исходные коды программ также доступны по ссылкам на GitHub:

https://github.com/adanil/OOP/tree/master/Lab3

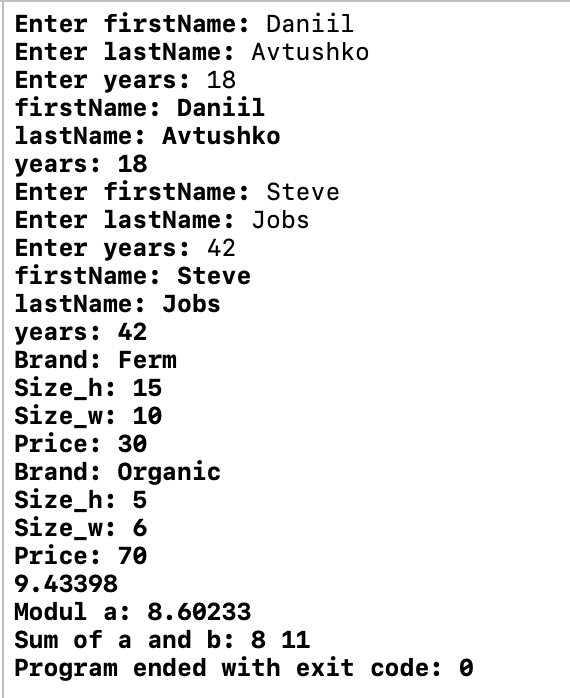
****

Рис. 3 Результат работы 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**

|  |
| --- |
| #include <iostream> |
|  | #include <string> |
|  | #include <cmath> |
|  | using namespace std; |
|  |  |
|  | //Вариант 1 |
|  | class Child{ |
|  | string firstName; |
|  | string lastName; |
|  | int years; |
|  | public: |
|  | Child(string a,string b,int c):firstName(a),lastName(b),years(c){} |
|  | ~Child(){ |
|  | firstName = ""; |
|  | lastName = ""; |
|  | years = 0; |
|  | cout << "Object destroyed" << endl; |
|  | } |
|  | void enterData(){ |
|  | cout << "Enter firstName: "; |
|  | cin >> firstName; |
|  | cout << "Enter lastName: "; |
|  | cin >> lastName; |
|  | cout << "Enter years: "; |
|  | cin >> years; |
|  | } |
|  | void printData(){ |
|  | cout << "firstName: " << firstName << endl; |
|  | cout << "lastName: " << lastName << endl; |
|  | cout << "years: " << years << endl; |
|  | } |
|  | }; |
|  | //Вариант 2 |
|  | class Tiles{ |
|  | public: |
|  | string brand; |
|  | int size\_h,size\_w,price; |
|  | Tiles(string b,int h,int w,int p):brand(b),size\_h(h),size\_w(w),price(p){} |
|  | ~Tiles(){ |
|  | brand = ""; |
|  | size\_h = size\_w = price = 0; |
|  | cout << "Object destroyed" << endl; |
|  | } |
|  | void getData(){ |
|  | cout << "Brand: " << brand << endl; |
|  | cout << "Size\_h: " << size\_h << endl; |
|  | cout << "Size\_w: " << size\_w << endl; |
|  | cout << "Price: " << price << endl; |
|  | } |
|  | }; |
|  | //Вариант 3 |
|  | class Complex{ |
|  | int Re\_,Im\_; |
|  | public: |
|  | Complex(double Re = 0,double Im = 0){ |
|  | Re\_ = Re; |
|  | Im\_ = Im; |
|  | } |
|  | ~Complex(){ |
|  | Re\_ = 0; |
|  | Im\_ = 0; |
|  | } |
|  | double module(){ |
|  | return sqrt(Re\_\*Re\_ + Im\_\*Im\_); |
|  | } |
|  | }; |
|  | //Вариант 4 |
|  | class Vctr{ |
|  | public: |
|  | int x,y; |
|  | Vctr(int a = 0,int b = 0):x(a),y(b){} |
|  | ~Vctr(){ |
|  | x = y = 0; |
|  | } |
|  | double module(){ |
|  | return sqrt(x\*x + y\*y); |
|  | } |
|  | Vctr sum(Vctr b){ |
|  | Vctr res; |
|  | res.x = this->x + b.x; |
|  | res.y = this->y + b.y; |
|  | return res; |
|  | } |
|  | Vctr diff(Vctr b){ |
|  | Vctr res; |
|  | res.x = this->x - b.x; |
|  | res.y = this->y - b.y; |
|  | return res; |
|  | } |
|  | }; |
|  | int main(int argc, const char \* argv[]) { |
|  | Child son("jonh","smith",18); |
|  | son.printData(); |
|  | Tiles exemp("organic",13,2,30); |
|  | exemp.getData(); |
|  | Complex cmplx(13,4); |
|  | cout << cmplx.module() << endl; |
|  | Vctr v(13,7); |
|  | cout << v.module() << endl; |
|  | return 0; |
|  | } |

**Вывод**

Были выполнены все задачи. В процессе работы были более подробно изучены разные виды конструкторов и деструкторы. Для выполнения работы были использованы простые конструкторы, конструкторы копирования, а также перегруженный оператор присваивания. Результаты работы программ можно увидеть на скриншотах (рис. 9-12). Исходный код программ также доступен по ссылкам на GitHub:

https://github.com/adanil/OOP/tree/master/Lab4

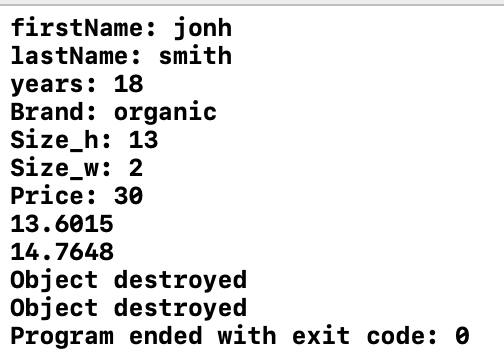


Рис. 4 Результат работы программы

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

**Ход работы**

|  |
| --- |
| #include <iostream> |
|  |  |
|  | using namespace std; |
|  |  |
|  | class Complex{ |
|  | int Re\_; |
|  | int Im\_; |
|  | public: |
|  | int getRe(){ |
|  | return Re\_; |
|  | } |
|  | int getIm(){ |
|  | return Im\_; |
|  | } |
|  | void setRe(int a){ |
|  | Re\_ = a; |
|  | } |
|  | void setIm(int a){ |
|  | Im\_ = a; |
|  | } |
|  | Complex(int a,int b):Re\_(a),Im\_(b){} |
|  | void operator= (Complex a){ |
|  | Re\_ = a.Re\_; |
|  | Im\_ = a.Im\_; |
|  | } |
|  | void operator++(){ |
|  | Re\_++; |
|  | Im\_++; |
|  | } |
|  | void operator--(){ |
|  | Re\_--; |
|  | Im\_--; |
|  | } |
|  | bool operator<(Complex a){ |
|  | return Re\_\*Re\_ + Im\_\*Im\_ < a.Re\_\*a.Re\_ + a.Im\_\*a.Im\_; |
|  | } |
|  | bool operator>(Complex a){ |
|  | return Re\_\*Re\_ + Im\_\*Im\_ > a.Re\_\*a.Re\_ + a.Im\_\*a.Im\_; |
|  | } |
|  | friend istream& operator>>(istream &in,Complex &a); |
|  | friend ostream& operator<<(ostream &out,Complex &a); |
|  | }; |
|  |  |
|  | istream& operator>>(istream &in,Complex &a){ |
|  | in >> a.Re\_; |
|  | in >> a.Im\_; |
|  | return in; |
|  | } |
|  | ostream& operator<<(ostream &out,Complex &a){ |
|  | out << a.Re\_ << ' ' << a.Im\_ << endl; |
|  | return out; |
|  | } |
|  |  |
|  | class Vctr{ |
|  | int x,y; |
|  | public: |
|  | Vctr(int a,int b):x(a),y(b){} |
|  | void operator=(Vctr a){ |
|  | x = a.x; |
|  | y = a.y; |
|  | } |
|  | bool operator<(Vctr a){ |
|  | return x \* x + y \* y < a.x \* a.x + a.y \* a.y; |
|  | } |
|  | bool operator>(Vctr a){ |
|  | return x \* x + y \* y > a.x \* a.x + a.y \* a.y; |
|  | } |
|  | friend istream& operator>>(istream &in,Vctr &a); |
|  | friend ostream& operator<<(ostream &out,Vctr &a); |
|  | }; |
|  |  |
|  | istream& operator>>(istream &in,Vctr &a){ |
|  | in >> a.x; |
|  | in >> a.y; |
|  | return in; |
|  | } |
|  | ostream& operator<<(ostream &out,Vctr &a){ |
|  | out << a.x << ' ' << a.y << endl; |
|  | return out; |
|  | } |
|  |  |
|  | int main(int argc, const char \* argv[]) { |
|  | Complex a(5,6),b(7,8),c(1,1); |
|  | a = b; |
|  | if (b > c) |
|  | cout << "True" << endl; |
|  | else |
|  | cout << "False" << endl; |
|  | cout << b << endl;; |
|  | Vctr k(8,8),d(7,3); |
|  | if (k < d) |
|  | cout << "True" << endl; |
|  | else |
|  | cout << "False" << endl; |
|  | cout << d << endl; |
|  | k = d; |
|  | cout << k << endl; |
|  | return 0; |
|  | } |

**Вывод**

Все поставленные задачи были выполнены. В процессе выполнения работы был более плотно изучен механизм перегрузки. Для выполнения поставленных задач была использована перегрузка некоторых бинарных и унарных операторов. Результаты работы программ можно увидеть на скриншотах (рис. 13, 14). Исходный код программ также доступен по ссылкам на GitHub:

https://github.com/adanil/OOP/tree/master/Lab5

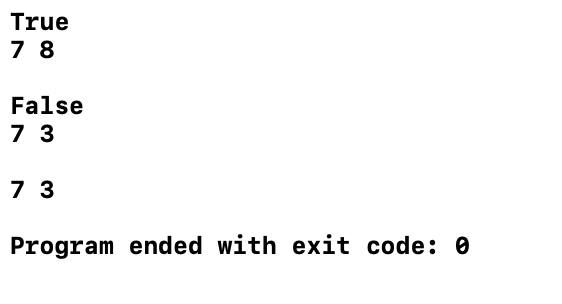


Рис. 5 Результат работы программы

## 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**

|  |
| --- |
| #include <iostream> |
|  | #include <string> |
|  | #include <vector> |
|  |  |
|  | using namespace std; |
|  |  |
|  | class Student{ |
|  | string firstName; |
|  | string lastName; |
|  | int id; |
|  | public: |
|  | Student(string a = "Jonh" , string b = "Smith"):firstName(a),lastName(b){} |
|  |  |
|  | }; |
|  |  |
|  | class Headman:Student{ |
|  | string email; |
|  | vector<Student> group; |
|  | public: |
|  | Headman(string em,vector<Student> &gr):email(em),group(gr){} |
|  | }; |
|  | //Вариант 2 |
|  | class Alive{ |
|  | string name; |
|  | int years; |
|  | public: |
|  | Alive(string nm = "Luccy",int y = 0):name(nm),years(y){} |
|  | }; |
|  |  |
|  | class Bird:Alive{ |
|  | int countWings; |
|  | public: |
|  | Bird(int wg):countWings(wg){} |
|  | }; |
|  |  |
|  | class Fish:Alive{ |
|  | string type; |
|  | public: |
|  | Fish(string tp):type(tp){} |
|  | }; |
|  |  |
|  | class Animal:Alive{ |
|  | string typeAnimal; |
|  | public: |
|  | Animal(string tpAn):typeAnimal(tpAn){} |
|  | }; |
|  | //Вариант 3 |
|  | class Animal3{ |
|  | string name; |
|  | int years; |
|  | public: |
|  | Animal3(string nm = "Tom",int ys = 0):name(nm),years(ys){} |
|  | }; |
|  | class Dog:Animal3{ |
|  | string breed; |
|  | public: |
|  | Dog(string brd):breed(brd){} |
|  | }; |
|  | class Cat:Animal3{ |
|  | int childs; |
|  | public: |
|  | Cat(int chld):childs(chld){} |
|  | }; |
|  |  |
|  | int main(int argc, const char \* argv[]) { |
|  | Student iam("Danil","Avtushko"); |
|  | vector<Student> gr; |
|  | gr.push\_back(iam); |
|  | Headman someone("exmpl@gmail.com",gr); |
|  | Fish forel("RedFish"); |
|  | Dog myDog("Pudel"); |
|  |  |
|  | return 0; |
|  | } |

**Вывод**

Все поставленные задачи были выполнены. В процессе выполнения работы было изучено и использовано наследование. Результат работы программ изображены на скриншотах (рис. 15-17). Исходный код программ также можно просмотреть на GitHub:

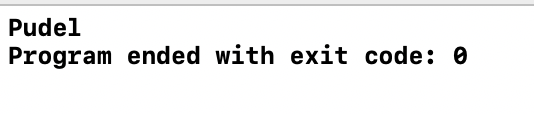
https://github.com/adanil/OOP/tree/master/Lab6

Рис. 6 Результат работы 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 UML диаграмме.

**Ход работы**

|  |
| --- |
|  |
| |  | | --- | | #include <iostream> | |  | #include <string> | |  |  | |  | using namespace std; | |  |  | |  | class Human{ | |  | string surname,name,midname; | |  | int age; | |  | public: | |  | Human(){ | |  | surname = "Avtushko"; | |  | name = "Danil"; | |  | age = 18; | |  | } | |  | Human(string n,string sn,string mn, int ag):name(n),surname(sn),midname(mn),age(ag){} | |  | ~Human(){ | |  | midname = ""; | |  | surname = ""; | |  | name = ""; | |  | age = 0; | |  | } | |  | virtual void print() = 0; | |  |  | |  | }; | |  |  | |  | class Student:Human{ | |  | bool on\_lesson; | |  | public: | |  | Student():Human(),on\_lesson(false){} | |  | Student(string n,string sn,string mn, int ag,bool ol):Human(n,sn,mn,ag),on\_lesson(ol){} | |  | void print(){ | |  | if (on\_lesson) | |  | cout <<"On lesson? True" << endl; | |  | else | |  | cout <<"On lesson? False" << endl; | |  | } | |  | ~Student(){ | |  | on\_lesson = false; | |  | } | |  |  | |  | }; | |  |  | |  | class Boss:Human{ | |  | int number\_of\_workers; | |  | public: | |  | Boss():Human(),number\_of\_workers(0){} | |  | Boss(string n,string sn,string mn, int ag,int now):Human(n,sn,mn,ag),number\_of\_workers(now){} | |  | void print(){ | |  | cout <<"Number of workers: "<< number\_of\_workers << endl; | |  | } | |  | ~Boss(){ | |  | number\_of\_workers = 0; | |  | } | |  | }; | |  |  | |  | int main(int argc, const char \* argv[]) { | |  | Student s("Jmishenko", "Dmitriy","Alekseevich", 54, true); | |  | s.print(); | |  | Boss b("Pushkin","Alexander","Sergeevich",33,15); | |  | b.print(); | |  | return 0; | |  | } | |
|  |
|  |
|  |
|  |
|  |
|  |
|  |
|  |
|  |
|  |
|  |
|  |
|  |
|  |
|  |
|  |
|  |
|  |
|  |
|  |
|  |
|  |
|  |
|  |
|  |
|  |
|  |
|  |
|  |
|  |
|  |
|  |
|  |
|  |
|  |
|  |
|  |
|  |
|  |
|  |
|  |
|  |
|  |
|  |
|  |
|  |
|  |
|  |
|  |
|  |
|  |
|  |
|  |
|  |
|  |
|  |
|  |
|  |
|  |
|  |
|  |
|  |
|  |
|  |
|  |
|  |
|  |
|  |
|  |
|  |
|  |
|  |
|  |
|  |
|  |
|  |
|  |
|  |
|  |
|  |
|  |
|  |
|  |
|  |
|  |
|  |
|  |
|  |
|  |
|  |
|  |
|  |
|  |
|  |
|  |
|  |
|  |
|  |
|  |
|  |
|  |

**Вывод**

Все поставленные задачи были выполнены. При выплнении работы были изучены основные элементы UML диаграмм. Для достижения поставленной цели были использованы механизмы наследования, переопределения функций и абстрактные классы. Результат работы программы изображён на скриншоте (рис. 18). Код программы также доступен на GitHub по ссылке:

https://github.com/adanil/OOP/tree/master/Lab7

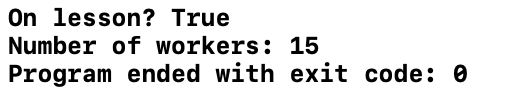


Рис. 7 Результат работы программы

## 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**

|  |
| --- |
| #include <iostream> |
|  | #include <vector> |
|  | #include <list> |
|  | #include <queue> |
|  |  |
|  | using namespace std; |
|  |  |
|  | class Student{ |
|  | string firstName; |
|  | string lastName; |
|  | int id; |
|  | public: |
|  | Student(string a = "Jonh" , string b = "Smith",int d = 0):firstName(a),lastName(b),id(d){} |
|  | void print(){ |
|  | cout << firstName << ' ' << lastName << endl; |
|  | } |
|  | }; |
|  | /////Zadanie 2 |
|  |  |
|  | class Complex{ |
|  | int Re\_; |
|  | int Im\_; |
|  | public: |
|  | int getRe(){ |
|  | return Re\_; |
|  | } |
|  | int getIm(){ |
|  | return Im\_; |
|  | } |
|  | void setRe(int a){ |
|  | Re\_ = a; |
|  | } |
|  | void setIm(int a){ |
|  | Im\_ = a; |
|  | } |
|  | Complex(int a,int b):Re\_(a),Im\_(b){} |
|  | void operator= (Complex a){ |
|  | Re\_ = a.Re\_; |
|  | Im\_ = a.Im\_; |
|  | } |
|  | void operator++(){ |
|  | Re\_++; |
|  | Im\_++; |
|  | } |
|  | void operator--(){ |
|  | Re\_--; |
|  | Im\_--; |
|  | } |
|  | bool operator<(Complex a){ |
|  | return Re\_\*Re\_ + Im\_\*Im\_ < a.Re\_\*a.Re\_ + a.Im\_\*a.Im\_; |
|  | } |
|  | bool operator>(Complex a){ |
|  | return Re\_\*Re\_ + Im\_\*Im\_ > a.Re\_\*a.Re\_ + a.Im\_\*a.Im\_; |
|  | } |
|  | }; |
|  |  |
|  |  |
|  | //Вариант 3 |
|  | class Auto{ |
|  | string brand; |
|  | int year; |
|  | int maxspeed; |
|  | string model; |
|  | public: |
|  | Auto(string b,int y,int ms,string md):brand(b),year(y),maxspeed(ms),model(md){} |
|  | void print(){ |
|  | cout << brand << ' ' << year << ' ' << maxspeed << ' ' << model << endl; |
|  | } |
|  | }; |
|  |  |
|  | int main(int argc, const char \* argv[]) { |
|  | Student a("Ivan","Ivanov",13567),b("Alex","Ryzhov",13883); |
|  | vector<Student> stdnts; |
|  | stdnts.push\_back(a); |
|  | stdnts.push\_back(b); |
|  | stdnts[1].print(); |
|  | //Zadanie 2 |
|  | Complex x(13,4),y(2,7),z(55,6); |
|  | list<Complex> cmplx; |
|  | cmplx.push\_back(x); |
|  | cmplx.push\_back(y); |
|  | cmplx.push\_back(z); |
|  | cout << cmplx.back().getIm() << ' ' << cmplx.back().getRe() << endl; |
|  | cmplx.pop\_back(); |
|  | cout << cmplx.back().getIm() << ' ' << cmplx.back().getRe() << endl; |
|  | cmplx.pop\_back(); |
|  | cout << cmplx.back().getIm() << ' ' << cmplx.back().getRe() << endl; |
|  |  |
|  | //Zadanie3 |
|  | Auto q("Opel",2001,200,"Astro"),w("Nissan",2010,240,"Qashkai"); |
|  | queue<Auto> at; |
|  | at.push(q); |
|  | at.push(w); |
|  | at.back().print(); |
|  | at.front().print(); |
|  |  |
|  |  |
|  |  |
|  | return 0; |
|  | } |

**Вывод**

Все поставленные задачи были выполнены. Была изучена библиотека STL. Для выполнения работы были использованы коллекции queue, string, list и vector. Результат работы программ изображён на скриншотах (рис. 19-21). Исходный код программ так же доступен на GitHub по ссылкам:

1. https://github.com/adanil/OOP/tree/master/Lab8

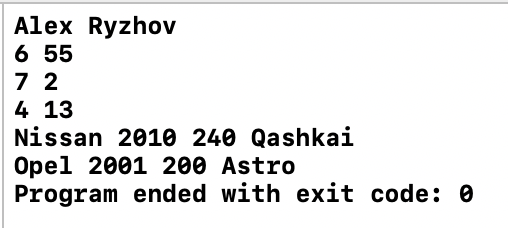


Рис. 8 Результат работы программы