|  |  |
| --- | --- |
| 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 | Рындин Д.П. |
| 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:

|  |
| --- |
| #include "stdafx.h" |
|  | #include <iostream> |
|  |  |
|  | using namespace std; |
|  | class Complex |
|  | { |
|  | double x, y; |
|  |  |
|  | public: |
|  | Complex(double x, double y); |
|  | Complex(); |
|  | void get\_Complex(); |
|  | string plus(Complex &other); |
|  | string minus(Complex & other); |
|  | string umno(Complex & other); |
|  | string razd(Complex & other); |
|  |  |
|  |  |
|  | }; |

Файл Complex.cpp:

|  |
| --- |
|  |
|  | #include "Complex.h" |
|  | #include<iostream> |
|  | #include<string> |
|  | using namespace std; |
|  |  |
|  |  |
|  | Complex::Complex(double x,double y ) |
|  | { |
|  | this->x = x; |
|  | this->y = y; |
|  |  |
|  | } |
|  |  |
|  | Complex::Complex() |
|  | { |
|  | cout << "vvedite veschesthennuyu i mnimuyu chasti " << endl; |
|  | cin >> x >> y; |
|  |  |
|  | } |
|  |  |
|  | void Complex::get\_Complex() |
|  | { |
|  | cout << x << "+i\*" << y << endl; |
|  |  |
|  | } |
|  |  |
|  | string Complex::plus(Complex & other) |
|  | { |
|  | return to\_string(x+other.x)+"+i\*"+to\_string(y + other.y); |
|  | } |
|  |  |
|  | string Complex::minus(Complex & other) |
|  | { |
|  | return to\_string(x - other.x) + "+i\*" + to\_string(y - other.y); |
|  | } |
|  |  |
|  | string Complex::umno(Complex & other) |
|  | { |
|  | return to\_string(x \* other.x-y\*other.y) + "+i\*" + to\_string(x\*other.y - y\*other.x); |
|  | } |
|  |  |
|  | string Complex::razd(Complex & other) |
|  | { |
|  | return to\_string(x \* other.x - y\*other.y/(other.x\*other.x + other.y\*other.y)) + "+i\*" + to\_string(x\*other.y - y\*other.x/(other.x\*other.x + other.y\*other.y)); |
|  | } |

Файл main.cpp:

|  |
| --- |
| #include "stdafx.h" |
|  | #include "Complex.h" |
|  | #include<string> |
|  | #include<iostream> |
|  | using namespace std; |
|  | class Complex; |
|  | int main() |
|  | { |
|  | Complex a; |
|  | Complex b; |
|  | a.get\_Complex(); |
|  | cout << a.plus(b) << endl; |
|  | cout << a.minus(b) << endl; |
|  | cout << a.umno(b) << endl; |
|  | cout << a.razd(b) << endl; |
|  | system("pause"); |
|  | return 0; |
|  | } |

1. Программа №2

|  |
| --- |
| #include "stdafx.h" |
|  |  |
|  |  |
|  | #include <iostream> |
|  | #include "Complex.h" |
|  | #include <vector> |
|  | #include <fstream> |
|  | using namespace std; |
|  |  |
|  | int main() { |
|  | ifstream file("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. https://github.com/dimaINBO0418/oop/blob/master/l1.1.txt
2. https://github.com/dimaINBO0418/oop/blob/master/l1.2.tx2

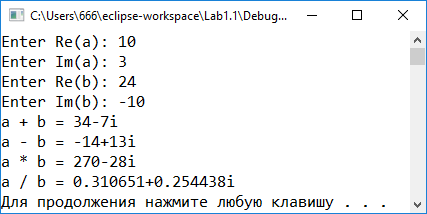
****

Рис. 1 Результат работы программы №1

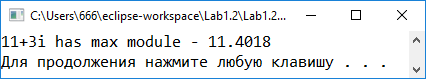


Рис. 2 Результат работы программы №2

## 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**

1. Программа №1

|  |
| --- |
|  |
|  | using namespace std; |
|  | double summa(double a, double b) { |
|  | return a + b; |
|  | } |
|  |  |
|  | double proizv(double a, double b) { |
|  | return a \* b; |
|  | } |
|  |  |
|  | double raznost(double a, double b) { |
|  | return a - b; |
|  | } |
|  | double(\*p)(double a, double b); |
|  | int main() |
|  | { |
|  | double a, b; |
|  | cout << "vvedite a i b" << endl; |
|  | cin >> a >> b; |
|  | int z; |
|  |  |
|  | cout << "1-raznost 2-summa 3-proizvedenie" << endl; |
|  | cin >> z; |
|  | switch (z) |
|  | { |
|  | case 1: p = raznost; |
|  | break; |
|  | case 2: p = summa; |
|  | break; |
|  | case 3: p = proizv; |
|  | break; |
|  | default: |
|  | break; |
|  | } |
|  |  |
|  | cout<<p(a, b)<<endl; |
|  | system("pause"); |
|  | return 0; |
|  | } |

1. Программа №2

|  |
| --- |
|  |
|  | #include <iostream> |
|  | #include<ctime> |
|  | #include<string> |
|  | using namespace std; |
|  | int compare(const void \* x1, const void \* x2) // функция сравнения элементов массива |
|  | { |
|  | return (\*(int\*)x1 - \*(int\*)x2); |
|  | } |
|  | int comp2(const void \* a, const void \* b) |
|  | { |
|  | return (\*(int\*)b - \*(int\*)a); |
|  | } |
|  |  |
|  | void f1(int \*a, int n) { for (int i = n - 1; i > -1; i--)cout << a[i]; cout << endl; } |
|  | void f2(int \*a, int n) { qsort(a, n, sizeof(int), compare); }; |
|  | void f3(int \*a, int n) { qsort(a, n, sizeof(int), comp2); }; |
|  |  |
|  | void (\*func(int\*a,int n))(int\*a,int n){ |
|  | int sum=0; |
|  | for (int i = 1; i < n; i++)sum += a[i]; |
|  | if (sum == a[0])return f1; |
|  | else if (sum > a[0])return f2; |
|  | else return f3; |
|  |  |
|  |  |
|  | } |
|  | int main() |
|  | { |
|  | srand(time(0)); |
|  | int n; |
|  | cin >> n; |
|  | int \*a= new int[n]; |
|  | a[0] = 1000; |
|  | for (int i = 0; i < n; i++) { |
|  | a[i] = rand() % 100+1; |
|  | cout << a[i] << endl; |
|  | } |
|  | //a[0] = 1000; |
|  |  |
|  | func(a, n)(a, n); |
|  | for (int i = 0; i < n; i++) { |
|  |  |
|  | cout << a[i] << " "; |
|  | } |
|  | cout << endl; |
|  | delete[] a; |
|  | system("pause"); |
|  | return 0; |
|  | } |

**Вывод**

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

1. [https://github.com/dimaINBO0418/oop/blob/master/l2.1.txt](https://github.com/VladRubel/laby/blob/master/lab%202.1)
2. <https://github.com/dimaINBO0418/oop/blob/master/l2.2.txt>

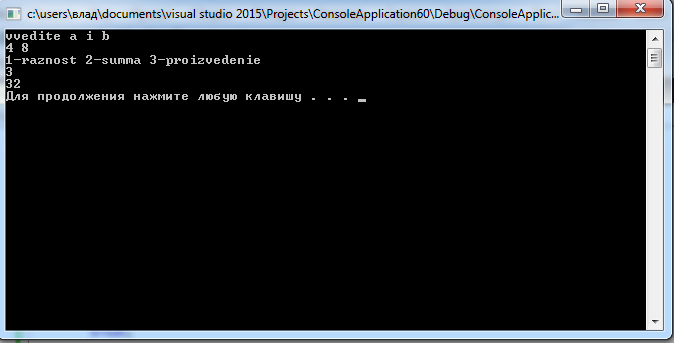


Рис. 3 Результат работы программы №1

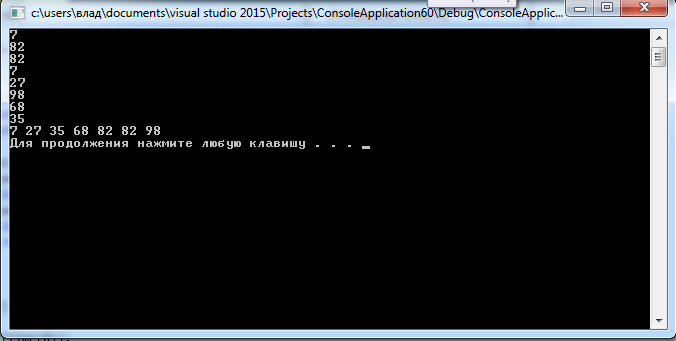


Рис. 4 Результат работы программы №2

## 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**

1. Программа №1

|  |
| --- |
|  |
|  | #include <iostream> |
|  | #include <string> |
|  | using namespace std; |
|  |  |
|  | class Child { |
|  | string name; |
|  | string surname; |
|  | int year; |
|  | public: |
|  | void set\_data() { |
|  | cout << "vvedite name surname year" << endl; |
|  | cin >> name; |
|  | cin >> surname; |
|  | cin >> year; |
|  | } |
|  | string get\_data() { |
|  | return name +' '+ surname+' ' + to\_string(year) ; |
|  | } |
|  | }; |
|  |  |
|  |  |
|  |  |
|  | int main() |
|  | { |
|  | Child a, b; |
|  | a.set\_data(); |
|  | cout<<a.get\_data()<<endl; |
|  | b.set\_data(); |
|  | cout << b.get\_data() << endl; |
|  | system("pause"); |
|  | return 0; |
|  | } |

1. Программы №2

|  |
| --- |
|  |
|  | #include <iostream> |
|  | #include <string> |
|  | using namespace std; |
|  |  |
|  | class Tiles { |
|  | public: |
|  | string brand; |
|  | int size\_h, size\_w, price; |
|  | string get\_Data() { |
|  | return brand + " " + to\_string(size\_h) + " " + to\_string(size\_w) + " " + to\_string(price); |
|  | } |
|  | }; |
|  |  |
|  |  |
|  |  |
|  | int main() |
|  | { |
|  | Tiles a,b; |
|  | a.brand = "pl"; |
|  | a.price = 1000; |
|  | a.size\_h = 10; |
|  | a.size\_w = 10; |
|  | cout<<a.get\_Data()<<endl; |
|  | b.brand = "ga"; |
|  | b.price = 2000; |
|  | b.size\_h = 20; |
|  | b.size\_w = 30; |
|  | cout << b.get\_Data()<<endl; |
|  |  |
|  | system("pause"); |
|  | return 0; |
|  | } |

1. Программа №3

|  |
| --- |
|  |
|  | #include <iostream> |
|  | #include <string> |
|  | #include<cmath> |
|  | using namespace std; |
|  |  |
|  | class Complex { |
|  | double Re, Im; |
|  | public: |
|  | void set\_c() { |
|  | cout << "vvedite deystvitelnuyu chast" << endl; |
|  | cin >> Re; |
|  | cout << "vvedite mnimuyu chast" << endl; |
|  | cin >> Im; |
|  | } |
|  | string get\_c() { |
|  | return to\_string(Re) + "+" + to\_string(Im) + "i"; |
|  | } |
|  | double modul() { |
|  | return sqrt(Re\*Re + Im\*Im); |
|  | } |
|  | double arg() { |
|  | return atan(Im / Re); |
|  | } |
|  | }; |
|  |  |
|  |  |
|  |  |
|  |  |
|  | int main() |
|  | { |
|  | Complex a; |
|  | a.set\_c(); |
|  | cout << a.get\_c() << endl;; |
|  | cout << a.modul()<<endl; |
|  | cout<<a.arg()<<endl; |
|  |  |
|  |  |
|  | system("pause"); |
|  | return 0; |
|  | } |

1. Программа №4

|  |
| --- |
|  |
|  | #include <iostream> |
|  | #include<string> |
|  | using namespace std; |
|  |  |
|  | class Vector { |
|  | double x, y, ugol; |
|  | public: |
|  | void set\_Vector() { |
|  | cout << "vvedite x i y koordinaty vectora" << endl; |
|  | cin >> x >> y; |
|  | } |
|  | double modul() { |
|  | return sqrt(x\*x + y \* y); |
|  | } |
|  |  |
|  | void Set\_ugol(Vector &other) { |
|  | ugol = this->x\*other.x + this->y\*other.y / (this->modul()\*other.modul()); |
|  |  |
|  | } |
|  | string summa(Vector &other) { |
|  | return to\_string(this->x + other.x) + "+" + to\_string(this->y + other.y); |
|  | } |
|  |  |
|  | string raznost(Vector &other) { |
|  | return to\_string(this->x - other.x) + "-" + to\_string(this->y - other.y); |
|  | } |
|  | double proizvsk(Vector &other) { |
|  |  |
|  | return this->modul()\*other.modul()\*cos(ugol); |
|  | } |
|  | double proizvvec(Vector &other) { |
|  |  |
|  | return this->modul()\*other.modul()\*sin(ugol); |
|  | } |
|  |  |
|  | }; |
|  |  |
|  |  |
|  |  |
|  | int main() |
|  | { |
|  | Vector a, b; |
|  | a.set\_Vector(); |
|  | b.set\_Vector(); |
|  | a.Set\_ugol(b); |
|  | cout <<"summa ravna"<< a.summa(b) << endl; |
|  | cout << "raznost ravna" << a.raznost(b) << endl; |
|  | cout << "proizv skal ravna" << a.proizvsk(b) << endl; |
|  | cout << "proizv vec ravna" << a.proizvvec(b) << endl; |
|  | system("pause"); |
|  | 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:

1. <https://github.com/dimaINBO0418/oop/blob/master/l3.1.txt>
2. <https://github.com/dimaINBO0418/oop/blob/master/l3.2.txt>
3. <https://github.com/dimaINBO0418/oop/blob/master/l3.3.txt>
4. <https://github.com/dimaINBO0418/oop/blob/master/l3.4.txt>

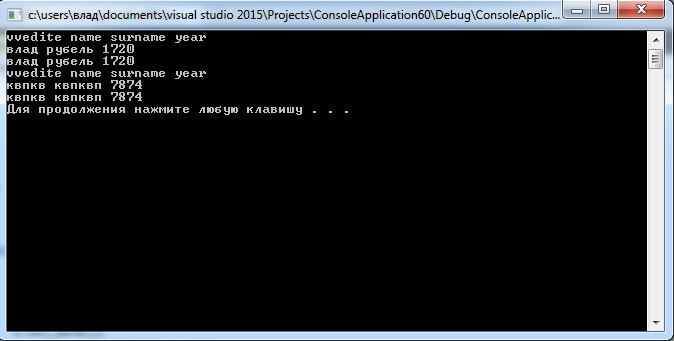
****

Рис. 5 Результат работы программы №1

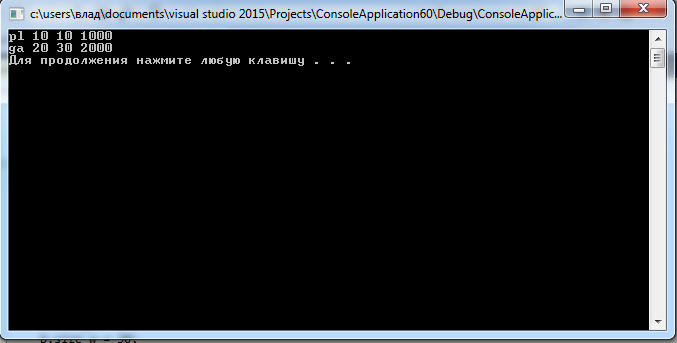
****

Рис. 6 Результат работы программы №2

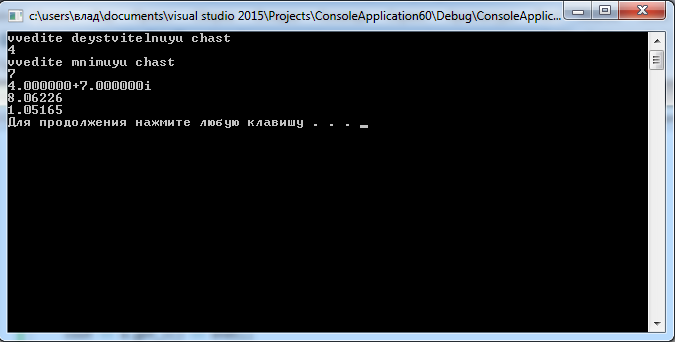
****

Рис. 7 Результат работы программы №3

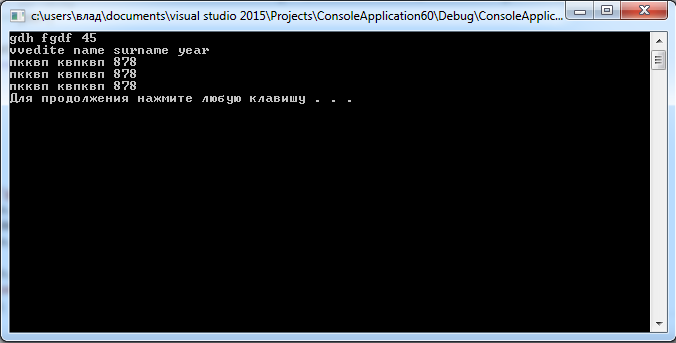
****

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**

1. Программа №1

|  |
| --- |
|  |
|  | #include <iostream> |
|  | #include <string> |
|  | using namespace std; |
|  |  |
|  | class Child { |
|  | string name; |
|  | string surname; |
|  | int year; |
|  | public: |
|  | Child(string name, string surname, int year) { |
|  |  |
|  | this->name = name; |
|  | this->surname = surname; |
|  | this->year = year; |
|  | } |
|  | Child(const Child &other) { |
|  | this->name = other.name; |
|  | this->surname = other.surname; |
|  | this->year = other.year; |
|  | } |
|  | ~Child() {}; |
|  | void reset\_data() { |
|  | cout << "vvedite name surname year" << endl; |
|  | cin >> name; |
|  | cin >> surname; |
|  | cin >> year; |
|  | } |
|  | string get\_data() { |
|  | return name + ' ' + surname + ' ' + to\_string(year); |
|  | } |
|  | }; |
|  |  |
|  |  |
|  |  |
|  | int main() |
|  | { |
|  | Child a("gdh", "fgdf", 45), b("", "", 0); |
|  | cout << a.get\_data() << endl; |
|  | a.reset\_data(); |
|  | cout << a.get\_data() << endl; |
|  | b = a; |
|  | cout << b.get\_data() << endl; |
|  | system("pause"); |
|  | return 0; |
|  | } |

1. Программа №2

|  |
| --- |
|  |
|  | #include <iostream> |
|  | #include <string> |
|  | using namespace std; |
|  |  |
|  | class Tiles { |
|  | string brand; |
|  | int size\_h, size\_w, price; |
|  | public: |
|  | Tiles() { |
|  | cout << "vvedite brand dliny shiriny ceny" << endl; |
|  | cin >> brand; |
|  | cin >> size\_h >> size\_w >> price; |
|  | } |
|  | Tiles(const Tiles &other) { |
|  | this->brand = other.brand; |
|  | this->size\_h = other.size\_h; |
|  | this->size\_w = other.size\_w; |
|  | this->price = other.price; |
|  | } |
|  | ~Tiles() {}; |
|  | string get\_Data() { |
|  | return brand + " " + to\_string(size\_h) + " " + to\_string(size\_w) + " " + to\_string(price); |
|  | } |
|  | }; |
|  |  |
|  |  |
|  |  |
|  | int main() |
|  | { |
|  | Tiles a; |
|  |  |
|  |  |
|  | cout<<"a=" << a.get\_Data() << endl; |
|  |  |
|  | Tiles b = a; |
|  | cout << "b=" << b.get\_Data() << endl; |
|  |  |
|  |  |
|  | system("pause"); |
|  | return 0; |
|  | } |

1. Программа №3

|  |
| --- |
|  |
|  | #include <iostream> |
|  | #include <string> |
|  | #include<cmath> |
|  | using namespace std; |
|  |  |
|  | class Complex { |
|  | double Re, Im; |
|  | public: |
|  | void set\_c() { |
|  | cout << "vvedite deystvitelnuyu chast" << endl; |
|  | cin >> Re; |
|  | cout << "vvedite mnimuyu chast" << endl; |
|  | cin >> Im; |
|  | } |
|  | string get\_c() { |
|  | return to\_string(Re) + "+" + to\_string(Im) + "i"; |
|  | } |
|  | double modul() { |
|  | return sqrt(Re\*Re + Im\*Im); |
|  | } |
|  | double arg() { |
|  | return atan(Im / Re); |
|  | } |
|  | }; |
|  |  |
|  |  |
|  |  |
|  |  |
|  | int main() |
|  | { |
|  | Complex a; |
|  | a.set\_c(); |
|  | cout << a.get\_c() << endl;; |
|  | cout << a.modul()<<endl; |
|  | cout<<a.arg()<<endl; |
|  |  |
|  |  |
|  | system("pause"); |
|  | return 0; |
|  | } |

1. Программа №4

|  |
| --- |
|  |
|  | #include <iostream> |
|  | #include<string> |
|  | using namespace std; |
|  |  |
|  | class Vector { |
|  | double x, y, ugol; |
|  | public: |
|  | void set\_Vector() { |
|  | cout << "vvedite x i y koordinaty vectora" << endl; |
|  | cin >> x >> y; |
|  | } |
|  | double modul() { |
|  | return sqrt(x\*x + y \* y); |
|  | } |
|  |  |
|  | void Set\_ugol(Vector &other) { |
|  | ugol = this->x\*other.x + this->y\*other.y / (this->modul()\*other.modul()); |
|  |  |
|  | } |
|  | string summa(Vector &other) { |
|  | return to\_string(this->x + other.x) + "+" + to\_string(this->y + other.y); |
|  | } |
|  |  |
|  | string raznost(Vector &other) { |
|  | return to\_string(this->x - other.x) + "-" + to\_string(this->y - other.y); |
|  | } |
|  | double proizvsk(Vector &other) { |
|  |  |
|  | return this->modul()\*other.modul()\*cos(ugol); |
|  | } |
|  | double proizvvec(Vector &other) { |
|  |  |
|  | return this->modul()\*other.modul()\*sin(ugol); |
|  | } |
|  |  |
|  | }; |
|  |  |
|  |  |
|  |  |
|  | int main() |
|  | { |
|  | Vector a, b; |
|  | a.set\_Vector(); |
|  | b.set\_Vector(); |
|  | a.Set\_ugol(b); |
|  | cout <<"summa ravna"<< a.summa(b) << endl; |
|  | cout << "raznost ravna" << a.raznost(b) << endl; |
|  | cout << "proizv skal ravna" << a.proizvsk(b) << endl; |
|  | cout << "proizv vec ravna" << a.proizvvec(b) << endl; |
|  | system("pause"); |
|  | return 0; |
|  | } |

**Вывод**

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

1. <https://github.com/dimaINBO0418/oop/blob/master/l4.1.txt>
2. <https://github.com/dimaINBO0418/oop/blob/master/l4.2.txt>
3. <https://github.com/dimaINBO0418/oop/blob/master/l4.3.txt>
4. <https://github.com/dimaINBO0418/oop/blob/master/l4.4.txt>

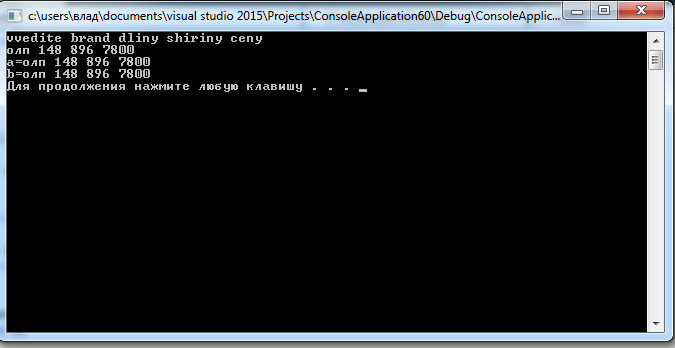


Рис. 9 Результат работы программы №1

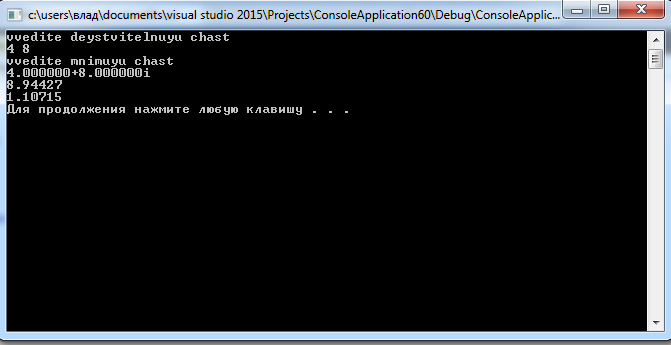
\

Рис. 10 Результат работы программы №2

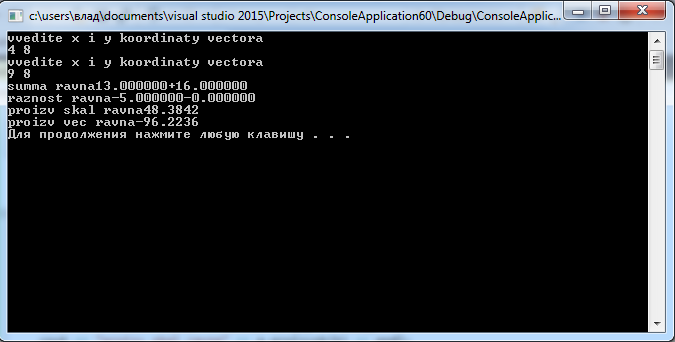


Рис. 11 Результат работы программы №3

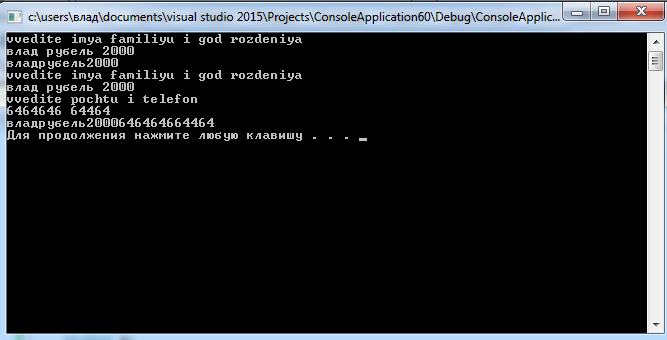


Рис. 12 Результат работы программы №4

## 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**

1. Программа №1

|  |
| --- |
|  |
|  | #include <iostream> |
|  | #include <string> |
|  | #include<cmath> |
|  | using namespace std; |
|  |  |
|  | class Complex { |
|  | double Re, Im; |
|  | public: |
|  | Complex() { |
|  | cout << "vvedite deystvitelnuyu chast" << endl; |
|  | cin >> Re; |
|  | cout << "vvedite mnimuyu chast" << endl; |
|  | cin >> Im; |
|  |  |
|  | } |
|  | Complex(const Complex &other) { |
|  | this->Re = other.Re; |
|  | this->Im = other.Im; |
|  | } |
|  | ~Complex() {}; |
|  | void set\_c() { |
|  | cout << "vvedite deystvitelnuyu chast" << endl; |
|  | cin >> Re; |
|  | cout << "vvedite mnimuyu chast" << endl; |
|  | cin >> Im; |
|  | } |
|  | string get\_c() { |
|  | return to\_string(Re) + "+" + to\_string(Im) + "i"; |
|  | } |
|  | double modul() { |
|  | return sqrt(Re\*Re + Im\*Im); |
|  | } |
|  | double arg() { |
|  | return atan(Im / Re); |
|  | } |
|  | friend istream& operator>> (std::istream &in, Complex &point) |
|  | { |
|  | cout << "vvedite deystvitelnuyu i mimuyu chast" << endl; |
|  | in >> point.Re; |
|  |  |
|  | in >> point.Im; |
|  |  |
|  |  |
|  | return in; |
|  | } |
|  | friend ostream& operator<< (std::ostream &out, const Complex &point) |
|  | { |
|  | // поскольку operator<< является другом класса Point, то мы имеем прямой доступ к членам Point |
|  | out << point.Re<<"+i\*"<<point.Im; |
|  |  |
|  | return out; |
|  | } |
|  | bool operator <(Complex &other) { |
|  | if (this->modul() < other.modul())return true; |
|  | else return false; |
|  | } |
|  | bool operator >(Complex &other) { |
|  | if (this->modul() > other.modul())return true; |
|  | else return false; |
|  | } |
|  | bool operator ==(Complex &other) { |
|  | if (this->modul() == other.modul())return true; |
|  | else return false; |
|  | } |
|  |  |
|  | Complex& operator ++() { |
|  | this->Re++; |
|  | return \*this; |
|  | } |
|  | Complex& operator --() { |
|  | this->Re--; |
|  | return \*this; |
|  | } |
|  | }; |
|  |  |
|  |  |
|  |  |
|  |  |
|  | int main() |
|  | { |
|  | Complex a; |
|  | Complex b = a; |
|  | cout <<"a-constructora" << a<<endl; |
|  | cout << "b=a" << b << endl; |
|  | cin >> a; |
|  | cout<<"a-izmenennoye" << a << endl ; |
|  |  |
|  | cout <<"modul a"<< a.modul() << endl; |
|  | cout << "argument a" << a.arg() << endl; |
|  | cout << "a++ " << ++a << endl; |
|  | cout << "a-- " << --a << endl; |
|  |  |
|  |  |
|  | if (a > b)cout << a << "a bolshe b" << endl; |
|  | else if (a < b)cout << b << "a bolshe b" << endl; |
|  | else if (a == b)cout << "a ravno b"; |
|  |  |
|  |  |
|  | system("pause"); |
|  | return 0; |
|  | } |

1. Программа №2

|  |
| --- |
|  |
|  | #include <iostream> |
|  | #include<string> |
|  | using namespace std; |
|  |  |
|  | class Vector { |
|  | double x, y, ugol; |
|  | public: |
|  | Vector() { |
|  | cout << "vvedite x i y koordinaty vectora" << endl; |
|  | cin >> x >> y; |
|  | } |
|  | Vector(const Vector &other) { |
|  | this->x = other.x; |
|  | this->y = other.y; |
|  | } |
|  | ~Vector(){} |
|  | void set\_Vector() { |
|  | cout << "vvedite x i y koordinaty vectora" << endl; |
|  | cin >> x >> y; |
|  | } |
|  | double modul() { |
|  | return sqrt(x\*x + y \* y); |
|  | } |
|  |  |
|  | void Set\_ugol(Vector &other) { |
|  | ugol = this->x\*other.x + this->y\*other.y / (this->modul()\*other.modul()); |
|  |  |
|  | } |
|  | Vector summa(Vector &other) { |
|  | this->x = this->x + other.x; |
|  | this->y = this->y + other.y; |
|  | return \*this; |
|  | } |
|  |  |
|  | Vector raznost(Vector &other) { |
|  | this->x = this->x - other.x; |
|  | this->y = this->y - other.y; |
|  | return \*this; |
|  | } |
|  | double proizvsk(Vector &other) { |
|  |  |
|  | return this->modul()\*other.modul()\*cos(ugol); |
|  | } |
|  | double proizvvec(Vector &other) { |
|  |  |
|  | return this->modul()\*other.modul()\*sin(ugol); |
|  | } |
|  | friend ostream& operator<< (std::ostream &out, const Vector &vector) |
|  | { |
|  | // поскольку operator<< является другом класса Point, то мы имеем прямой доступ к членам Point |
|  | out <<"("<<vector.x<<","<< vector.y<<")"; |
|  |  |
|  | return out; |
|  | } |
|  | friend istream& operator>> (std::istream &in, Vector &vector) |
|  | { |
|  | // поскольку operator>> является другом класса Point, то мы имеем прямой доступ к членам Point |
|  | cout << "vvedite koorinaty x i y" << endl; |
|  | // обратите внимание, параметр point (объект класса Point) должен быть не константным, чтобы мы имели возможность изменить члены класса |
|  | in >> vector.x; |
|  | in >> vector.y; |
|  |  |
|  |  |
|  | return in; |
|  | } |
|  | bool operator > (Vector & other) { |
|  | if (this->modul() > other.modul())return true; |
|  | else return false; |
|  | } |
|  | bool operator < (Vector & other) { |
|  | if (this->modul() < other.modul())return true; |
|  | else return false; |
|  | } |
|  | bool operator == (Vector & other) { |
|  | if (this->modul() == other.modul())return true; |
|  | else return false; |
|  | } |
|  | Vector& operator ++ () { |
|  | this->x = x + 1; |
|  | this->y = y + 1; |
|  | return \*this; |
|  | } |
|  | Vector& operator -- () { |
|  | this->x = x - 1; |
|  | this->y = y - 1; |
|  | return \*this; |
|  | } |
|  | }; |
|  |  |
|  |  |
|  |  |
|  | int main() |
|  | { |
|  | Vector a, b ; |
|  | if (a > b)cout << a <<"bolshe"<< endl; |
|  | else if (a < b)cout << b << "bolshe" << endl; |
|  | else if (a == b)cout << "ravno"; |
|  | cout << "incr a" << ++a<<endl; |
|  | cout << "decr a" << --a<<endl; |
|  |  |
|  | a.Set\_ugol(b); |
|  | cout << "summa ravna" << a.summa(b) << endl; |
|  | cout << "raznost ravna" << a.raznost(b) << endl; |
|  | cout << "proizv skal ravna" << a.proizvsk(b) << endl; |
|  | cout << "proizv vec ravna" << a.proizvvec(b) << endl; |
|  | system("pause"); |
|  |  |
|  | return 0; |
|  | } |

**Вывод**

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

1. <https://github.com/dimaINBO0418/oop/blob/master/l5.1.txt>
2. <https://github.com/dimaINBO0418/oop/blob/master/l5.2.txt>

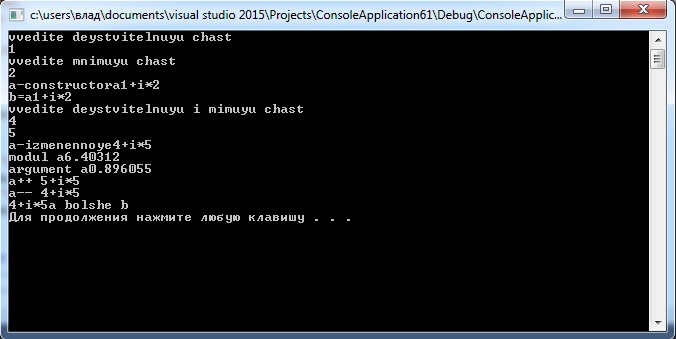


Рис. 13 Результат работы программы №1

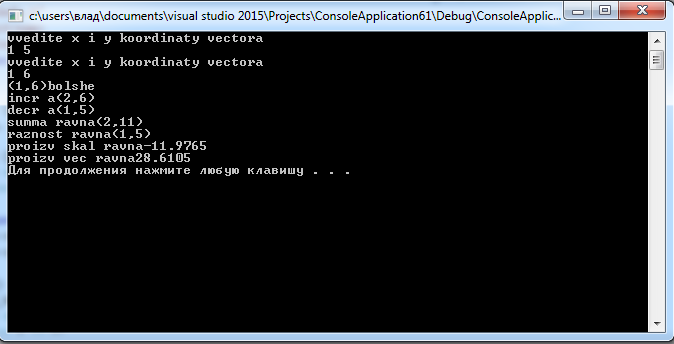


Рис. 14 Результат работы программы №2

## 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**

1. Программа №1

|  |
| --- |
|  |
|  |
|  | #include <iostream> |
|  | #include <string> |
|  | #include <vector> |
|  |  |
|  |  |
|  | using namespace std; |
|  | class Student { |
|  | protected: |
|  | string name; |
|  | string surname; |
|  | int year; |
|  | public: |
|  | Student(string &name, string &surname, int year) { |
|  | this->name = name; |
|  | this->surname = surname; |
|  | this->year = year; |
|  |  |
|  | } |
|  | Student() { |
|  | cout << "vvedite imya familiyu i god rozdeniya" << endl; |
|  | cin >> name; |
|  | cin >> surname; |
|  | cin >> year; |
|  | } |
|  | string get\_Data() { |
|  | return name + surname + to\_string(year); |
|  | } |
|  |  |
|  | }; |
|  |  |
|  | class Headman:public Student { |
|  | string email; |
|  | int phone; |
|  | public: |
|  | Headman(string& email, int phone):Student(name,surname,year){ |
|  | this->email = email; |
|  | this->phone = phone; |
|  | } |
|  | Headman() :Student() { |
|  | cout << "vvedite pochtu i telefon" << endl; |
|  | cin >> email; |
|  | cin>> phone; |
|  | } |
|  |  |
|  | string get\_HData() { |
|  | return get\_Data() + email + to\_string(phone); |
|  | } |
|  | }; |
|  |  |
|  |  |
|  | int main() { |
|  | Student a; |
|  | cout << a.get\_Data() << endl; |
|  | Headman b; |
|  | cout << b.get\_HData() << endl; |
|  | system("pause"); |
|  | return 0; |
|  | } |
|  |  |
|  |  |

1. Программа №2

|  |
| --- |
|  |
|  | #include <iostream>  #include <string> |
|  | #include <Windows.h> |
|  | using namespace std; |
|  |  |
|  | class Alive { // Базовый класс |
|  | protected: |
|  | string name; |
|  | int age; |
|  | public: |
|  | Alive(string &name, int age) : name(name), age(age) {}; |
|  |  |
|  | string &getName() { return name; } |
|  |  |
|  | int getAge() { return age; } |
|  |  |
|  | virtual void move() = 0; |
|  | }; |
|  |  |
|  | class Animal : public Alive { // Производные классы |
|  | public: |
|  | Animal(string &name, int age) : Alive(name, age) {}; |
|  |  |
|  | void move() { cout << name << " is running" << endl; } |
|  | }; |
|  |  |
|  | class Fish : public Alive { |
|  | public: |
|  | Fish(string &name, int age) : Alive(name, age) {}; |
|  |  |
|  | void move() { cout << name << " is swimming" << endl; } |
|  | }; |
|  |  |
|  | class Bird : public Alive { |
|  | public: |
|  | Bird(string &name, int age) : Alive(name, age) {}; |
|  |  |
|  | void move() { cout << name << " is flying"; } |
|  | }; |
|  |  |
|  | int main() { |
|  | SetConsoleCP(1251); |
|  | SetConsoleOutputCP(1251); |
|  |  |
|  | string name; |
|  | int age; |
|  |  |
|  | cout << "Fish" << endl; |
|  | cout << "Enter name: "; |
|  | cin >> name; |
|  | cout << "Enter age: "; |
|  | cin >> age; |
|  | Fish f(name, age); |
|  |  |
|  | cout << endl << "Animal" << endl; |
|  | cout << "Enter name: "; |
|  | cin >> name; |
|  | cout << "Enter age: "; |
|  | cin >> age; |
|  | Animal a(name, age); |
|  |  |
|  | cout << endl << "Bird" << endl; |
|  | cout << "Enter name: "; |
|  | cin >> name; |
|  | cout << "Enter age: "; |
|  | cin >> age; |
|  | Bird b(name, age); |
|  |  |
|  | cout << endl << "Your bird" << endl << "Name: " << b.getName() << endl << "Age: " << |
|  | b.getAge() << endl; |
|  | b.move(); |
|  |  |
|  | cout << endl << "Your fish" << endl << "Name: " << f.getName() << endl << "Age: " << |
|  | f.getAge() << endl; |
|  | f.move(); |
|  |  |
|  | cout << endl << "Your animal" << endl << "Name: " << a.getName() << endl << "Age: " << |
|  | a.getAge() << endl; |
|  | a.move(); |
|  |  |
|  | system("pause"); |
|  | return 0; |
|  | } |

1. Программа №3

|  |
| --- |
|  |
|  | #include <iostream> |
|  | #include <string> |
|  | using namespace std; |
|  |  |
|  | class Animal { // Базовый класс |
|  | protected: |
|  | string name; |
|  | int age; |
|  | public: |
|  | Animal(string name, int age) : name(name), age(age) {}; |
|  |  |
|  | string getName() { return name; } |
|  |  |
|  | int getAge() { return age; } |
|  |  |
|  | virtual void golos() = 0; |
|  | }; |
|  |  |
|  | class Dog : public Animal { // Производные классы |
|  | public: |
|  | Dog(string name, int age) : Animal(name, age) {}; |
|  |  |
|  | void golos() { cout << "gav gav gav" << endl; } |
|  | }; |
|  |  |
|  | class Cat : public Animal { |
|  | public: |
|  | Cat(string name, int age) : Animal(name, age) {}; |
|  |  |
|  | void golos() { cout << "myau" << endl; } |
|  | }; |
|  |  |
|  | int main() { |
|  |  |
|  |  |
|  | string name; |
|  | int age; |
|  |  |
|  | cout << "kot" << endl; |
|  | cout << "vvedite imya: "; |
|  | cin >> name; |
|  | cout << "vvedite god: "; |
|  | cin >> age; |
|  | Cat c(name, age); |
|  |  |
|  | cout << endl << "sobaka" << endl; |
|  | cout << "vvedite imya: "; |
|  | cin >> name; |
|  | cout << "vvedite god: "; |
|  | cin >> age; |
|  | Dog d(name, age); |
|  |  |
|  | cout << endl << "tvoyua sobaka" << endl << "imya: " << d.getName() << endl << "god: " << |
|  | d.getAge() << endl; |
|  | d.golos(); |
|  |  |
|  | cout << endl << "tvoy cot" << endl << "imya: " << c.getName() << endl << "god: " << |
|  | c.getAge() << endl; |
|  | c.golos(); |
|  |  |
|  | system("pause"); |
|  | 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:

1. <https://github.com/dimaINBO0418/oop/blob/master/l6.1.txt>
2. <https://github.com/dimaINBO0418/oop/blob/master/l6.2.txt>
3. <https://github.com/dimaINBO0418/oop/blob/master/l6.3.txt>

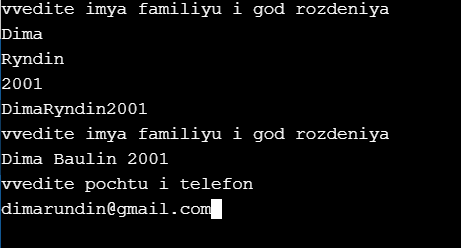


Рис. 15 Результат работы программы №1

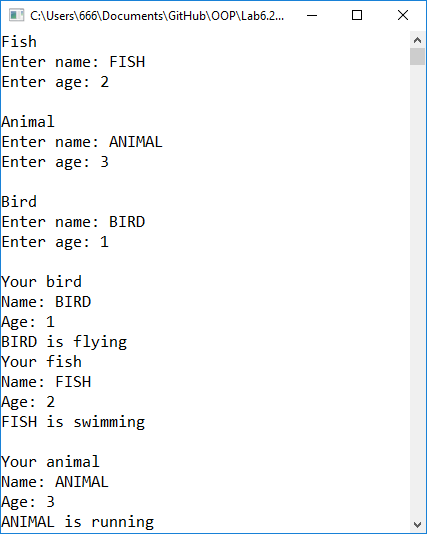


Рис. 16 Результат работы программы №2

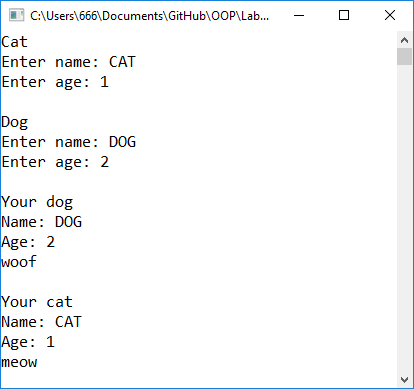


Рис. 17 Результат работы программы №3

## 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**

|  |
| --- |
|  |
|  | #include <iostream>  #include <string> |
|  | #include <Windows.h> |
|  | using namespace std; |
|  |  |
|  | class Human { |
|  | protected: |
|  | string name; |
|  | string surname; |
|  | string midname; |
|  | int age; |
|  | public: |
|  | Human() : name(""), surname(""), midname(""), age(0) {}; |
|  |  |
|  | Human(string surname, string name, string midname, int age) : |
|  | name(name), surname(surname), midname(midname), age(age) {}; |
|  |  |
|  | virtual void print() = 0; |
|  |  |
|  | ~Human() {} |
|  | }; |
|  |  |
|  | class Student : public Human { |
|  | bool on\_lesson; |
|  | public: |
|  | Student() : Human(), on\_lesson(false) {}; |
|  |  |
|  | Student(string surname, string name, string midname, int age, bool on\_lesson) : |
|  | Human(surname, name, midname, age), on\_lesson(on\_lesson) {}; |
|  |  |
|  | void print(); |
|  |  |
|  | ~Student() {}; |
|  | }; |
|  |  |
|  | class Boss : public Human { |
|  | int number\_of\_workers; |
|  | public: |
|  | Boss() : Human(), number\_of\_workers(0) {}; |
|  |  |
|  | Boss(string surname, string name, string midname, int age, int number\_of\_workers) : |
|  | Human(surname, name, midname, age), number\_of\_workers(number\_of\_workers) {}; |
|  |  |
|  | void print(); |
|  |  |
|  | ~Boss() {}; |
|  | }; |
|  |  |
|  | int main() { |
|  | SetConsoleCP(1251); |
|  | SetConsoleOutputCP(1251); |
|  |  |
|  | string surname, name, midname; |
|  | int age, number\_of\_workers; |
|  | bool on\_lesson; |
|  |  |
|  | cout.setf(ios::boolalpha); |
|  |  |
|  | cout << "Student" << endl; |
|  | cout << "Enter surname: "; |
|  | cin >> surname; |
|  | cout << "Enter name: "; |
|  | cin >> name; |
|  | cout << "Enter midname: "; |
|  | cin >> midname; |
|  | cout << "Enter age: "; |
|  | cin >> age; |
|  | cout << "Is on lesson? (1 - yes, 0 - no): "; |
|  | cin >> on\_lesson; |
|  |  |
|  | Student s(surname, name, midname, age, on\_lesson); |
|  |  |
|  | cout << endl << "Boss" << endl; |
|  | cout << "Enter surname: "; |
|  | cin >> surname; |
|  | cout << "Enter name: "; |
|  | cin >> name; |
|  | cout << "Enter midname: "; |
|  | cin >> midname; |
|  | cout << "Enter age: "; |
|  | cin >> age; |
|  | cout << "Enter number of workers: "; |
|  | cin >> number\_of\_workers; |
|  |  |
|  | Boss b(surname, name, midname, age, number\_of\_workers); |
|  | s.print(); |
|  | b.print(); |
|  |  |
|  | system("pause"); |
|  | return 0; |
|  | } |
|  |  |
|  | void Student::print() { |
|  | cout << "Name: " << surname << " " << name << " " << midname << endl; |
|  | cout << "Age: " << age << endl; |
|  | cout << "Is on lesson: " << on\_lesson << endl; |
|  | } |
|  |  |
|  | void Boss::print() { |
|  | cout << "Name: " << surname << " " << name << " " << midname << endl; |
|  | cout << "Age: " << age << endl; |
|  | cout << "Number of workers: " << number\_of\_workers << endl; |
|  | } |

**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/dimaINBO0418/oop/blob/master/l7.txt

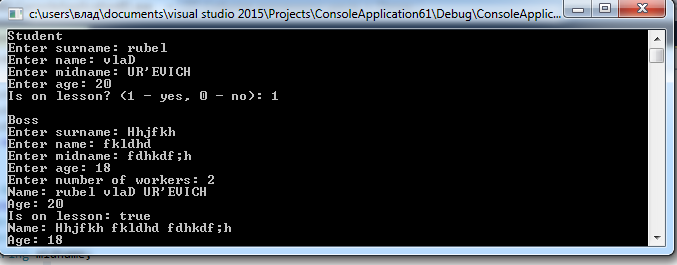


Рис. 18 Результат работы программы

## Лабораторная работа №8

**Бибилиотека STL**

**Цель практической работы**

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

**Задачи**

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

**Ход работы**

1. Программа №1

|  |
| --- |
|  |
|  |
|  | #include <iostream> |
|  | #include <string> |
|  | #include <vector> |
|  |  |
|  |  |
|  | using namespace std; |
|  | class Student { |
|  | protected: |
|  | string name; |
|  | string surname; |
|  | int year; |
|  | public: |
|  | Student(string &name, string &surname, int year) { |
|  | this->name = name; |
|  | this->surname = surname; |
|  | this->year = year; |
|  |  |
|  | } |
|  | Student() { |
|  | cout << "vvedite imya familiyu i god rozdeniya" << endl; |
|  | cin >> name; |
|  | cin >> surname; |
|  | cin >> year; |
|  | } |
|  | string get\_Data() { |
|  | return name + surname + to\_string(year); |
|  | } |
|  | friend ostream& operator<< (std::ostream &out, const Student &student) |
|  | { |
|  | // поскольку operator<< является другом класса Point, то мы имеем прямой доступ к членам Point |
|  | out << student.name<<student.surname<<student.year; |
|  |  |
|  | return out; |
|  | } |
|  | }; |
|  |  |
|  | class Headman:public Student { |
|  | string email; |
|  | int phone; |
|  | public: |
|  | Headman(string& email, int phone):Student(name,surname,year){ |
|  | this->email = email; |
|  | this->phone = phone; |
|  | } |
|  | Headman() :Student() { |
|  | cout << "vvedite pochtu i telefon" << endl; |
|  | cin >> email; |
|  | cin>> phone; |
|  | } |
|  |  |
|  | string get\_HData() { |
|  | return get\_Data() + email + to\_string(phone); |
|  | } |
|  |  |
|  | }; |
|  |  |
|  |  |
|  | int main() { |
|  |  |
|  | Student a; |
|  | vector<Student> p; |
|  | p.push\_back(a); |
|  | p.push\_back(a); |
|  | p.push\_back(a); |
|  | p.push\_back(a); |
|  |  |
|  | cout << "studenty:" << endl; |
|  | for (int i = 0; i < p.size(); i++)cout << p[i] << endl; |
|  | cout << "starosta:" << endl; |
|  | Headman b; |
|  | cout << b.get\_HData() << endl; |
|  | system("pause"); |
|  | return 0; |
|  | } |
|  |  |

1. Программа №2

Класс Complex был взят из лабораторной работы №1.

|  |
| --- |
|  |
|  | #include <iostream> |
|  | #include <string> |
|  | #include<cmath> |
|  | #include<list> |
|  | using namespace std; |
|  |  |
|  | class Complex { |
|  | double Re, Im; |
|  | public: |
|  |  |
|  | Complex() { |
|  | cout << "vvedite deystvitelnuyu chast" << endl; |
|  | cin >> Re; |
|  | cout << "vvedite mnimuyu chast" << endl; |
|  | cin >> Im; |
|  |  |
|  | } |
|  |  |
|  | Complex(double im, double re) { |
|  | this->Im = im; |
|  | this->Re = re; |
|  | } |
|  | Complex(const Complex &other) { |
|  | this->Re = other.Re; |
|  | this->Im = other.Im; |
|  | } |
|  | ~Complex() {}; |
|  | void set\_c() { |
|  | cout << "vvedite deystvitelnuyu chast" << endl; |
|  | cin >> Re; |
|  | cout << "vvedite mnimuyu chast" << endl; |
|  | cin >> Im; |
|  | } |
|  | string get\_c() { |
|  | return to\_string(Re) + "+" + to\_string(Im) + "i"; |
|  | } |
|  | double modul() { |
|  | return sqrt(Re\*Re + Im\*Im); |
|  | } |
|  | double arg() { |
|  | return atan(Im / Re); |
|  | } |
|  | friend istream& operator>> (std::istream &in, Complex &point) |
|  | { |
|  | cout << "vvedite deystvitelnuyu i mimuyu chast" << endl; |
|  | in >> point.Re; |
|  |  |
|  | in >> point.Im; |
|  |  |
|  |  |
|  | return in; |
|  | } |
|  | friend ostream& operator<< (std::ostream &out, const Complex &point) |
|  | { |
|  | // поскольку operator<< является другом класса Point, то мы имеем прямой доступ к членам Point |
|  | out << point.Re << "+i\*" << point.Im; |
|  |  |
|  | return out; |
|  | } |
|  | bool operator <(Complex &other) { |
|  | if (this->modul() < other.modul())return true; |
|  | else return false; |
|  | } |
|  | bool operator >(Complex &other) { |
|  | if (this->modul() > other.modul())return true; |
|  | else return false; |
|  | } |
|  | bool operator ==(Complex &other) { |
|  | if (this->modul() == other.modul())return true; |
|  | else return false; |
|  | } |
|  |  |
|  | Complex& operator ++() { |
|  | this->Re++; |
|  | return \*this; |
|  | } |
|  | Complex& operator --() { |
|  | this->Re--; |
|  | return \*this; |
|  | } |
|  | }; |
|  |  |
|  |  |
|  |  |
|  |  |
|  | int main() |
|  | { |
|  | list<Complex> a; |
|  | a.push\_back(Complex()); |
|  | auto it = a.begin(); |
|  |  |
|  | int z=-1; |
|  | do { |
|  | cout << "vvedite 1 dlya dobavleniya 0-exit" << endl; |
|  |  |
|  | cin >> z; |
|  | if (z == 1)a.push\_back(Complex()); |
|  | } while (z != 0); |
|  |  |
|  | for (it = a.begin(); it != a.end();++it)cout << \*it << endl; |
|  |  |
|  | system("pause"); |
|  | return 0; |
|  | } |

1. Программа №3

|  |  |
| --- | --- |
|  | #include <iostream |
|  | #include <string> |
|  | #include<cmath> |
|  | #include<queue> |
|  | using namespace std; |
|  |  |
|  | class avto { |
|  | int skorost; |
|  | int god; |
|  | string nazvanie; |
|  |  |
|  | public: |
|  | avto() { |
|  | cout << "vvedite god vypuska" << endl; |
|  | cin >> this->god; |
|  | cout << "vvedite maksimalnuyu skorost" << endl; |
|  | cin >> this->skorost; |
|  | cout << "vvedite nazvaniye" << endl; |
|  | cin >> this->nazvanie; |
|  | } |
|  | void set\_avto() { |
|  | cout << "vvedite god vypuska" << endl; |
|  | cin >> this->god; |
|  | cout << "vvedite maksimalnuyu skorost" << endl; |
|  | cin >> this->skorost; |
|  | cout << "vvedite nazvaniye" << endl; |
|  | cin >> this->nazvanie; |
|  |  |
|  | } |
|  | void get\_avto() { |
|  | cout << god << endl; |
|  | cout << skorost << "km/h" << endl; |
|  | cout << nazvanie << endl; |
|  |  |
|  | } |
|  | friend ostream& operator<< (std::ostream &out, const avto &avto) |
|  | { |
|  | // поскольку operator<< является другом класса Point, то мы имеем прямой доступ к членам Point |
|  | out << avto.god << endl << avto.skorost << "km/h" << endl << avto.nazvanie << endl; |
|  |  |
|  | return out; |
|  | } |
|  | }; |
|  |  |
|  |  |
|  | int main() |
|  | { |
|  | queue<avto> a; |
|  | int z ; |
|  |  |
|  | do { |
|  | cout << "1-zaehat 2-vyehat" << endl; |
|  | cin >> z; |
|  | switch (z) |
|  | { |
|  | case 1: a.push(avto()); |
|  | cout << "avto v ocheredi" << endl; |
|  | break; |
|  | case 2: |
|  | cout<<a.front() << "avto vyshlo iz ocheredi" << endl ; |
|  |  |
|  | default: |
|  |  |
|  | break; |
|  | } |
|  | } while (z != 0); |
|  | system("pause"); |
|  | return 0; |
|  | } |

**Вывод**

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

1. <https://github.com/dimaINBO0418/oop/blob/master/l8.1.txt>
2. <https://github.com/dimaINBO0418/oop/blob/master/l8.2.txt>
3. <https://github.com/dimaINBO0418/oop/blob/master/l8.3.txt>

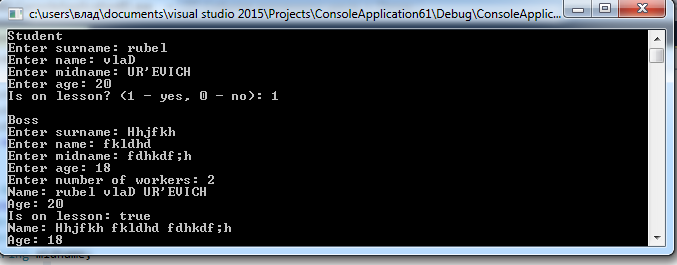


Рис. 19 Результат работы программы №1

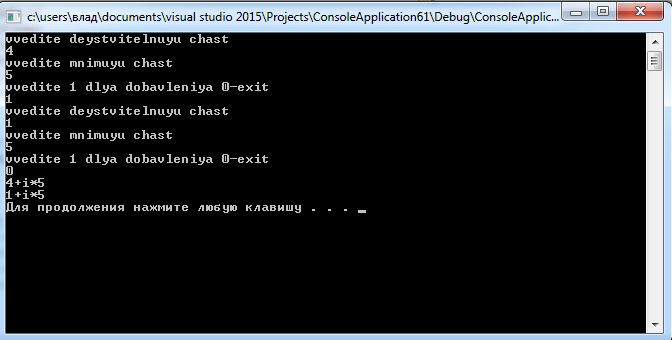


Рис. 20 Результат работы программы №2

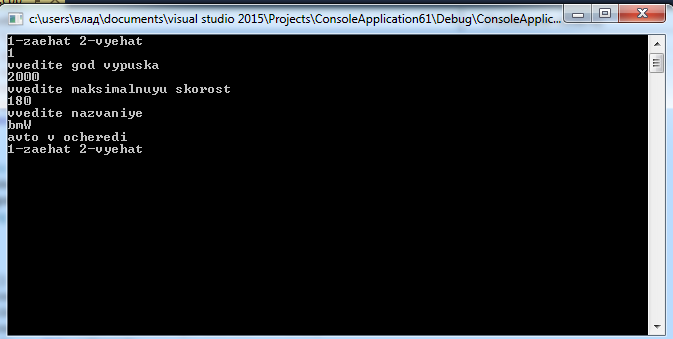


Рис. 21 Результат работы программы №3