|  |  |
| --- | --- |
| 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 | Мельник Н.В. |
| 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 |  |

**Практические 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 |  |
|  |  |  | Итоговый балл: | 36 |  |

## Практическая работа №1

**Изучение синтаксиса языка C++**

**Цель практической работы**

Целью данной практической работы является знакомство с синтаксисом, основными алгоритмическими конструкциями языка С++, а также приобретение практических навыков разработки объектно-ориентированных программ на языке программирования С++.

**Задачи**

1. Реализовать программу, которая считывает имя и здоровается
2. Реализовать программу, производящую операции над числами. (Сложение/вычитание, целые числа/числа с плавающей запятой)

**Ход работы**

1. Реализовать программу, которая считывает имя и здоровается

**#include** <iostream>

**#include** <string>

**using** **namespace** std;

**int** **main**() {

**setlocale**(LC\_ALL, "");

string name;

cout << "Enter your name: ";

cin >> name;

cout << "Hello, " << name << "!" << **endl**;

**system**("pause");

**return** 0;

}

1. Реализовать программу, производящую операции над числами. (Сложение/вычитание, целые числа/числа с плавающей запятой)

**#include** <iostream>

**using** **namespace** std;

**int** **main**() {

**setlocale**(LC\_ALL, "");

**double** a, b;

cout << "Enter two numbers: ";

cin >> a >> b;

cout << "a - b = " << (a - b) << **endl**;

cout << "a + b = " << (a + b) << **endl**;

**system**("pause");

**return** 0;

}

**Вывод**

В ходе выполнения практической работы была использована библиотека стандартного ввода и вывода, операторы ввода, вывода, сложения и вычитания. Были достигнуты результаты, изображённые на рисунках (рис. 1 и 2). Также исходный код программ был помещён на GitHub и доступен по ссылке:

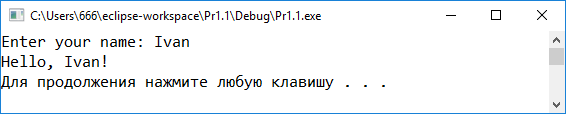
<https://github.com/Manskript/Praktika>

Рис. 1 Результат работы первой программы

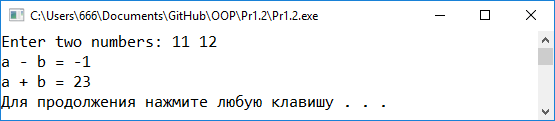


Рис. 2 Результат работы второй программы

## 

## Практическая работа №2

**Решение задач на нелинейные алгоритмические конструкции языка С++**

**Цель практической работы**

Целью данной практической работы является получение практических навыков по разработке нелинейных алгоритмов с использованием конструкции «if-else».

**Ход работы**

1. Вариант 1

**#include** <iostream>

**#include** <cmath>

**using** **namespace** std;

**int** **main**() {

**double** a, b, m, n;

cout << "Enter a and b: ";

cin >> a >> b;

**if** (a >= b) {

n = **cbrt**(a - b);

} **else** {

n = a \* a + (a - b) / (**sin**(a \* b));

}

**if** (n < b) {

m = (n + a) / (-b) + **sqrt**(**sin**(a) \* **sin**(a) - **cos**(n));

} **else** {

**if** (n == b) {

m = b \* b + **tan**(n \* a);

} **else** {

m = b \* b \* b + n \* a \* a;

}

}

**if** (isnan(m) || isnan(n) || isinf(m) || isinf(n)) {

cout << "Error: cannot count result" << **endl**;

} **else** {

cout << "a = " << a << **endl**;

cout << "b = " << b << **endl**;

cout << "m = " << m << **endl**;

cout << "n = " << n << **endl**;

}

**system**("pause");

**return** 0;

}

1. Вариант 2

**#include** <iostream>

**#include** <cmath>

**using** **namespace** std;

**int** **main**() {

**double** a, b, z, t;

cout << "Enter a and b: ";

cin >> a >> b;

**if** (a < b) {

z = **sqrt**(abs(a \* a - b \* b));

} **else** {

z = 1 - 2 \* **cos**(a) \* **sin**(b);

}

**if** (z < b) {

t = **cbrt**(z + a \* a \* b);

} **else** {

**if** (z == b) {

t = 1 - **log10**(z) + **cos**(a \* a \*b);

} **else** {

t = 1 / **cos**(z \* a);

}

}

**if** (isnan(t) || isnan(z) || isinf(z) || isinf(t)) {

cout << "Error: cannot count result" << **endl**;

} **else** {

cout << "a = " << a << **endl**;

cout << "b = " << b << **endl**;

cout << "t = " << t << **endl**;

cout << "z = " << z << **endl**;

}

**system**("pause");

**return** 0;

}

1. Вариант 3

**#include** <iostream>

**#include** <cmath>

**using** **namespace** std;

**int** **main**() {

**double** a, b, y, t;

cout << "Enter a and b: ";

cin >> a >> b;

**if** (a <= b) {

y = (a - b) \* (a + b) / ((a + b) \* (a \* a - a \* b + b \* b));

} **else** {

y = a + pow(**log**(b), 2);

}

**if** (y == b) {

t = (2 \* y + **sqrt**(y \* y - a)) / (2 \* b - **sqrt**(a \* a - y));

} **else** {

**if** (y < b) {

t = **sin**(y) \* **sin**(y) + 1 / **tan**(a - b);

} **else** {

t = **cbrt**(y \* **sin**(a)) + 1 / **sqrt**(y \* **cos**(b));

}

}

**if** (isnan(y) || isnan(t) || isinf(y) || isinf(t)) {

cout << "Error: cannot count result" << **endl**;

} **else** {

cout << "a = " << a << **endl**;

cout << "b = " << b << **endl**;

cout << "y = " << y << **endl**;

cout << "t = " << t << **endl**;

}

**system**("pause");

**return** 0;

}

1. Вариант 4

**#include** <iostream>

**#include** <cmath>

**using** **namespace** std;

**int** **main**() {

**double** a, x, y, t;

cout << "Enter a and x: ";

cin >> a >> x;

**if** (a <= x) {

y = a + **log**(x + a);

} **else** {

y = **sqrt**(**sin**(a \* x));

}

**if** (a > y) {

t = y / (a - x);

} **else** {

**if** (a == y) {

t = y / (a - x) + (a + x) / (y \* y);

} **else** {

t = **tan**(a \* x) + **cos**(2 \* a \* y);

}

}

**if** (isnan(y) || isnan(t) || isinf(y) || isinf(t)) {

cout << "Error: cannot count result" << **endl**;

} **else** {

cout << "a = " << a << **endl**;

cout << "x = " << x << **endl**;

cout << "y = " << y << **endl**;

cout << "t = " << t << **endl**;

}

**system**("pause");

**return** 0;

}

**Вывод**

В ходе выполнения практической работы была использована библиотека cmath, которая позволила вычислять сложные математические функции, а также быстро совершить проверку на ошибки, и конструкция ветвления «if-else». Были достигнуты результаты, изображённые на рисунках (рис. 3-6). Также код программ доступен по ссылке:

<https://github.com/Manskript/Praktika>

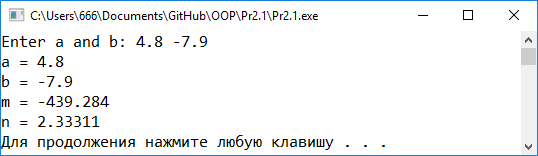


Рис. 3 Результат практической работы №2 вариант 1

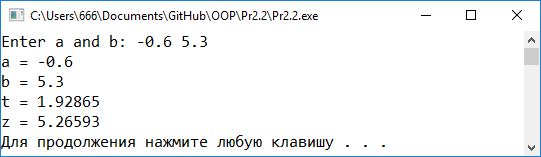


Рис. 4 Результат практической работы №2 вариант 2

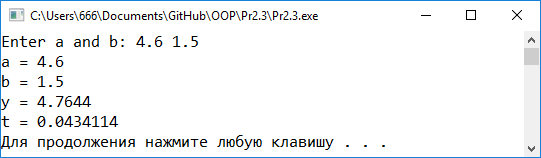


Рис. 5 Результат практической работы №2 вариант 3

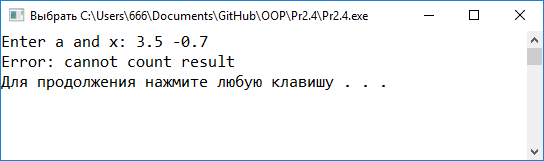


Рис. 6 Результат практической работы №2 вариант 4

## Практическая работа №3

**Классы в языке С++**

**Цель практической работы**

Целью данной практической работы является приобретение практических навыков при разработке классов на языке программирования С++.

**Задачи**

1. Реализовать класс «Книга»
2. Реализовать класс «Группа студентов»
3. Реализовать класс «Автомобиль»
4. Реализовать класс «Вектор»
5. Реализовать класс «Библиотека»

**Ход работы**

1. Реализовать класс «Книга»

|  |
| --- |
|  |
|  | #include <iostream>  #include <string> |
|  | using namespace std; |
|  | class Kniga |
|  | { |
|  | public: |
|  | string autor; |
|  | string Name; |
|  | double stranica; |
|  | void open(); |
|  | }; |
|  | int main() |
|  | { |
|  | Kniga Book; |
|  | Book.open(); |
|  | system("pause"); |
|  | return 0; |
|  |  |
|  | } |
|  | void Kniga::open() |
|  | { |
|  | double random; |
|  | cout << "vvedite avtora" << endl; |
|  | cin >> autor; |
|  | cout << "vvedite nazvanie" << endl; |
|  | cin >> Name; |
|  | random = 1 + rand() % 50; |
|  | cout << " " << autor << " " << "" << Name << " " << " otkryta na" <<" "<< random <<" "<< "stranice" << endl; |
|  | } |

1. Реализовать класс «Группа студентов»

|  |
| --- |
|  |
|  | #include <iostream>  #include <string> |
|  | using namespace std; |
|  |  |
|  | class Students { |
|  | string \*members; |
|  | int n; |
|  | public: |
|  | void makestudents(); |
|  | void getstudent(); |
|  | }; |
|  | void Students::makestudents() { |
|  | cout << "vvedite kolichestvo studentov" << endl; |
|  | cin >> n; |
|  | members = new string[n]; |
|  | for (int i = 0; i < n; i++){cout<<"vvedite familiyu studenta"<<endl; |
|  | cin >> members[i]; } |
|  |  |
|  | } |
|  | void Students::getstudent() { |
|  | for (int i = 0; i < n; i++)cout<<"student "<<i+1<<" " << members[i]<< endl; |
|  | } |
|  |  |
|  | int main() |
|  | { |
|  | Students a; |
|  | a.makestudents(); |
|  | a.getstudent(); |
|  | system("pause"); |
|  | return 0; |
|  | } |

1. Реализовать класс «Автомобиль»

|  |
| --- |
|  |
|  | #include <iostream>  #include <string> |
|  |  |
|  | using namespace std; |
|  |  |
|  | class avto{ |
|  | int skorost; |
|  | int god; |
|  | string nazvanie; |
|  |  |
|  | public: |
|  | avto() { |
|  | god = 0; |
|  | skorost = 0; |
|  | nazvanie = "net nazvaniya"; |
|  | } |
|  | 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; |
|  |  |
|  | } |
|  |  |
|  | }; |
|  |  |
|  | int main() |
|  | { |
|  | avto a; |
|  | a.set\_avto(); |
|  | a.get\_avto(); |
|  |  |
|  |  |
|  | system("pause"); |
|  | return 0; |
|  | } |

1. Реализовать класс «Вектор»

|  |
| --- |
|  |
|  | #include <iostream>  #include <string> |
|  | #include <cmath> |
|  | using namespace std; |
|  | class vector { |
|  | int x, y; |
|  |  |
|  | public: |
|  | vector(){ |
|  | x = 0, y = 0; |
|  |  |
|  | } |
|  | vector(int x, int y) { |
|  | this->x = x; |
|  | this->y = y; |
|  | } |
|  | void set\_vector() { |
|  | int x1, y1, x2, y2; |
|  | cout << "vvedite koordinati tochec" << endl; |
|  | cout << "x1, y1"; |
|  | cin >> x1 >> y1; |
|  | cout<<endl; |
|  | cout << "x2, y2"; |
|  | cin >> x2 >> y2; |
|  | cout << endl; |
|  | x = x2 - x1; |
|  | y = y2 - y1; |
|  |  |
|  | } |
|  |  |
|  | double modul() { |
|  | return sqrt(x\*x + y\*y); |
|  |  |
|  |  |
|  | } |
|  |  |
|  |  |
|  | }; |
|  |  |
|  | int main() |
|  | { |
|  | vector a; |
|  | vector b(3,4); |
|  | a.set\_vector(); |
|  | cout <<" modul vectora raven "<< a.modul() << endl;; |
|  |  |
|  |  |
|  |  |
|  |  |
|  | system("pause"); |
|  | return 0; |
|  | } |

1. Реализовать класс «Библиотека»

|  |
| --- |
|  |
|  |
|  | #include <iostream>  #include <string> |
|  | #include <cmath> |
|  | using namespace std; |
|  | class vector { |
|  | int x, y; |
|  |  |
|  | public: |
|  | vector(){ |
|  | x = 0, y = 0; |
|  |  |
|  | } |
|  | vector(int x, int y) { |
|  | this->x = x; |
|  | this->y = y; |
|  | } |
|  | void set\_vector() { |
|  | int x1, y1, x2, y2; |
|  | cout << "vvedite koordinati tochec" << endl; |
|  | cout << "x1, y1"; |
|  | cin >> x1 >> y1; |
|  | cout<<endl; |
|  | cout << "x2, y2"; |
|  | cin >> x2 >> y2; |
|  | cout << endl; |
|  | x = x2 - x1; |
|  | y = y2 - y1; |
|  |  |
|  | } |
|  |  |
|  | double modul() { |
|  | return sqrt(x\*x + y\*y); |
|  |  |
|  |  |
|  | } |
|  |  |
|  |  |
|  | }; |
|  |  |
|  | int main() |
|  | { |
|  | vector a; |
|  | vector b(3,4); |
|  | a.set\_vector(); |
|  | cout <<" modul vectora raven "<< a.modul() << endl;; |
|  |  |
|  |  |
|  |  |
|  |  |
|  | system("pause"); |
|  | return 0; |
|  | } |

**Вывод**

Все поставленные задачи были выполнены. Для этого были использованы библиотека STL, перегрузка оператора равенства, шаблоны функций. Все задачи были выполнены в виде много файлового проекта с разбиением классов на файлы объявления и реализации класса. Результаты работы программ приведены в рисунках (рис. 7-11). Код программ также доступен на GitHub по ссылке:

<https://github.com/Manskript/Praktika>

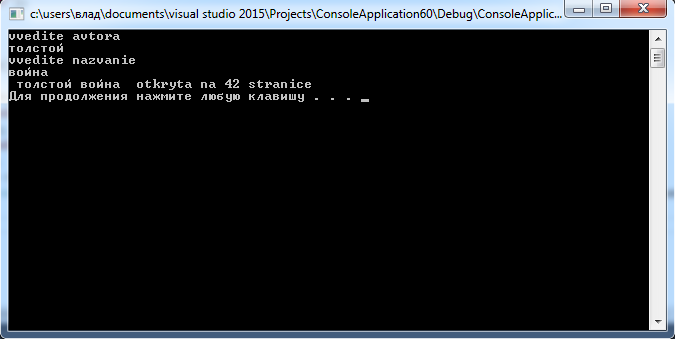


Рис. 7 Результат работы программы №1

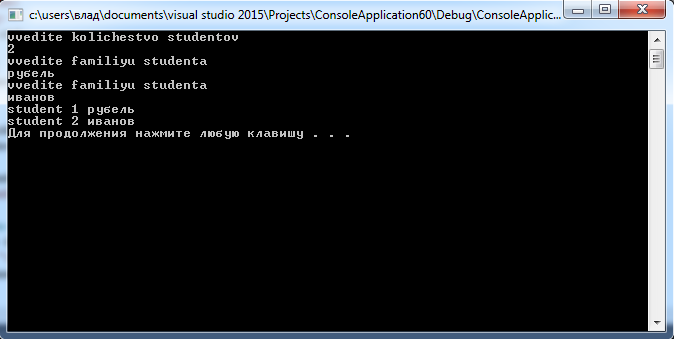


Рис. 8 Результат работы программы №2

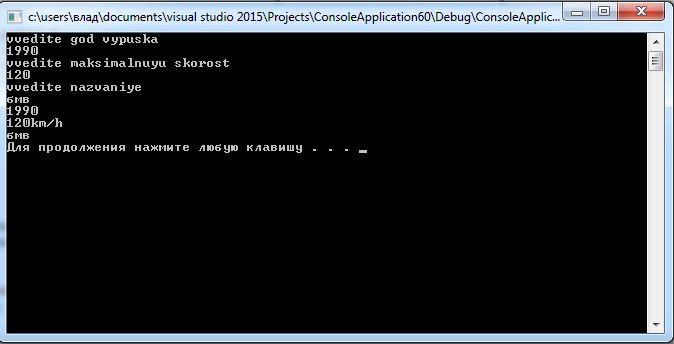


Рис. 9 Результат работы программы №3

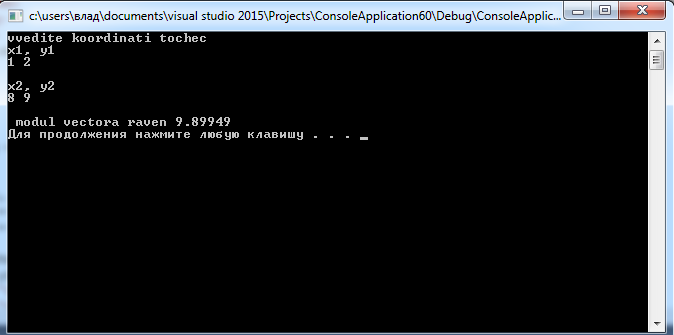


Рис 10 Результат работы программы №4

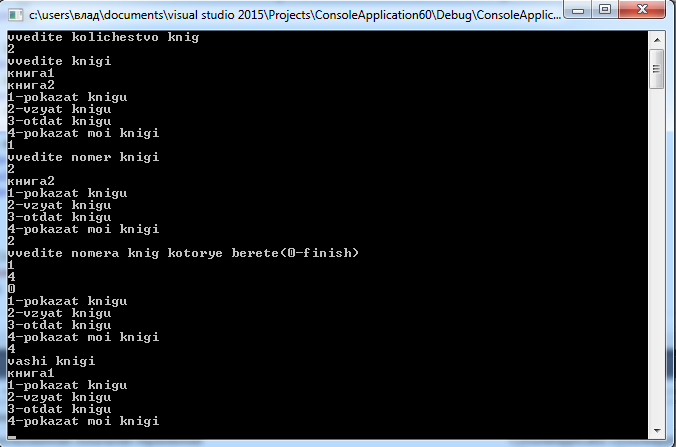


Рис. 11 Результат работы программы №5

## Практическая работа №4

**Динамическое выделение памяти для объектов в языке С++**

**Цель практической работы**

Целью данной практической работы является изучение динамического выделения и удаления памяти, а так же работа с указателями на языке С++.

**Задачи**

1. Объявите указатель на массив типа double и предложите пользователю выбрать его размер. Далее напишите четыре функции: первая должна выделить память для массива, вторая – заполнить ячейки данными, третья – показать данные на экран, четвёртая – освободить занимаемую память. Программа должна предлагать продолжать работу (создавать новые массивы) или выйти из программы.
2. Объявите указатель на массив типа int и выделите память для 12-ти элементов. Необходимо написать функцию, которая поменяет значения четных и нечетных ячеек массива.
3. Объявить и заполнить двумерный динамический массив случайными числами от 10 до 50. Показать его на экран. Для заполнения и показа на экран написать отдельные функции. Количество строк и столбцов выбирает пользователь

**Ход работы**

1. Программа №1

**#include** <iostream>

**#include** <ctime>

**using** **namespace** std;

**double** \***createArray**(**int**); // Создаёт массив

**void** **fillArray**(**double** \*, **int**); // Заполняет массив случайными числами

**void** **printArray**(**double** \*, **int**); // Выводит массив на экран

**void** **deleteArray**(**double** \*, **int**); // Удаляет массив

**int** **main**() {

**srand**(**time**(NULL));

**int** size;

cout << "Enter size of massive(lower than 1 if you want to exit the program): ";

cin >> size;

**while** (size > 0) {

**double** \*arr = createArray(size);

fillArray(arr, size);

printArray(arr, size);

deleteArray(arr, size);

cout << "Enter size of massive(lower than 1 if you want to exit the program): ";

cin >> size;

}

**return** 0;

}

**double** \***createArray**(**int** size) { **return** **new** **double**[size]; }

**void** **fillArray**(**double** \*arr, **int** size) {

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

arr[i] = **rand**() % 100;

}

}

**void** **printArray**(**double** \*arr, **int** size) {

cout << "Array: ";

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

cout << arr[i] << ' ';

}

cout << **endl**;

}

**void** **deleteArray**(**double** \*arr, **int** size) { **delete** [] arr; }

1. Программа №2

**#include** <iostream>

**#include** <ctime>

**using** **namespace** std;

**void** **fillArray**(**int** \*, **int**); // Заполняет массив случайными числами

**void** **printArray**(**int** \*, **int**); // Выводит массив на экран

**void** **changeElements**(**int** \*, **int**); // Меняет местами пары элементов массива

**int** **main**() {

**srand**(**time**(NULL));

**int** size = 12;

**int** \*arr = **new** **int**[size];

fillArray(arr, size);

printArray(arr, size);

changeElements(arr, 12);

printArray(arr, size);

**delete** [] arr;

**system**("pause");

**return** 0;

}

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

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

arr[i] = **rand**() % 100;

}

}

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

cout << "Array: ";

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

cout << arr[i] << ' ';

}

cout << **endl**;

}

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

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

**int** c = arr[i];

arr[i] = arr[i+1];

arr[i+1] = c;

}

}

1. Программа №3

**#include** <iostream>

**#include** <ctime>

**using** **namespace** std;

**void** **fillArray**(**int** \*\*, **int**, **int**); // Заполняет массив случайными числами

**void** **printArray**(**int** \*\*, **int**, **int**); // Выводит массив на экран

**int** **main**() {

**srand**(**time**(NULL));

**int** m, n;

cout << "Enter number of rows and columns: ";

cin >> n >> m;

**int** \*\*arr = **new** **int**\*[n];

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

arr[i] = **new** **int**[m];

}

fillArray(arr, n, m);

printArray(arr, n, m);

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

**delete** [] arr[i];

}

**delete** [] arr;

**system**("pause");

**return** 0;

}

**void** **fillArray**(**int** \*\*arr, **int** n, **int** m) {

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

**for** (**int** j = 0; j < m; j++) {

arr[i][j] = **rand**() % 41 + 10;

}

}

}

**void** **printArray**(**int** \*\*arr, **int** n, **int** m) {

cout << "Array:" << **endl**;

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

**for** (**int** j = 0; j < m; j++) {

cout << arr[i][j] << ' ';

}

cout << **endl**;

}

}

**Вывод**

Все поставленные задачи были выполнены. При выполнении использовались массивы с динамическим выделением памяти, которые создавались с помощью генератора случайных чисел. Для генератора случайных чисел была подключена библиотека ctime. Результат работы программ приведён на рисунках (рис.12-14). Так же исходный код программ доступен на GitHub по ссылке:

<https://github.com/Manskript/Praktika>

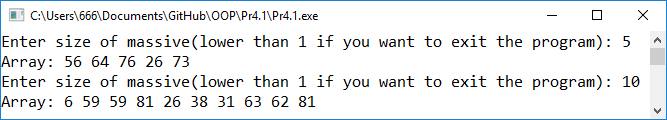


Рис. 12 Результат работы программы №1

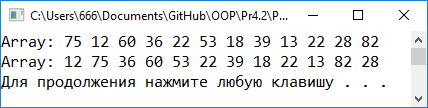


Рис. 13 Результат работы программы №2

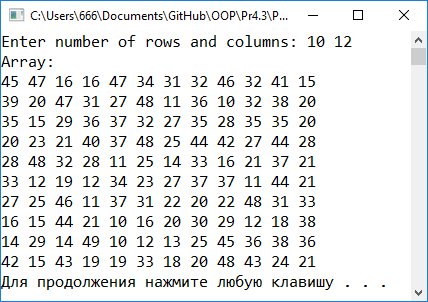


Рис. 14 Результат работы программы №3

## Практическая работа №5

**Перегрузка функций**

**Цель практической работы**

Целью данной практической работы является приобретение практических навыков по программированию перегрузки функций на языке C++.

**Задачи**

1. Реализовать сортировку пузырьком для целых чисел, а затем перегрузить её для дробных.
2. Реализовать сортировку выбором для целых чисел, а затем перегрузить её для дробных.
3. Реализовать сортировку вставками для целых чисел, а затем перегрузить её для дробных.
4. Реализовать программу-калькулятор, работающую с разными типами данных.

**Ход работы**

1. Программа №1

|  |
| --- |
|  |
|  | #include <iostream>  #include<ctime> |
|  | #include <iomanip> |
|  | using namespace std; |
|  | void sort(int \*a , int n ); |
|  | void sort(double \*a , int n ); |
|  | int main() { |
|  | srand(time(0)); |
|  | int n,z; |
|  | cout<<"vvedite kolichestvo chisel"<<endl; |
|  | cin >> n; |
|  | cout<<"kakiye chisla budete vvodit(0-double/1-int)"<<endl; |
|  | cin>>z; |
|  | if (z == 0) { |
|  | double\*a = new double[n]; |
|  | for (int i = 0; i < n; i++) { |
|  | cout<<" vvedite chislo"<<endl; |
|  | cin >> a[i]; |
|  | } |
|  | sort(a, n); |
|  |  |
|  | for (int i = 0; i < n; i++) |
|  | cout << a[i] << endl; |
|  | } |
|  | else { |
|  | int\*a = new int[n]; |
|  | for (int i = 0; i < n; i++) { |
|  | cout<<" vvedite chislo"<<endl; |
|  | cin >> a[i]; |
|  | } |
|  | sort(a, n); |
|  |  |
|  | for (int i = 0; i < n; i++) |
|  | cout << a[i] << endl; |
|  | } |
|  | system("pause"); |
|  | return 0; |
|  |  |
|  | } |
|  |  |
|  | void sort(double \*a, int n) |
|  | { |
|  | int i, j, tmp; |
|  | cout<<"idet sortirovka drobnyh chisel"<<endl; |
|  | for (i = 0; i < n; i++) |
|  | for (j = n - 1; j >= 0; j--) |
|  | if (a[j] < a[j + 1]) |
|  | { |
|  | tmp = a[j]; |
|  | a[j] = a[j + 1]; |
|  | a[j + 1] = tmp; |
|  | } |
|  | } |
|  |  |
|  | void sort(int \*a, int n) |
|  | { |
|  | int i, j, tmp; |
|  | cout<<"idet sortirovka celyh chisel"<<endl; |
|  | for (i = 0; i < n; i++) |
|  | for (j = n - 1; j >= 0; j--) |
|  | if (a[j] < a[j + 1]) |
|  | { |
|  | tmp = a[j]; |
|  | a[j] = a[j + 1]; |
|  | a[j + 1] = tmp; |
|  | } |
|  | } |

1. Программа №2

|  |
| --- |
|  |
|  | #include <iostream>  #include<ctime> |
|  | #include <iomanip> |
|  | using namespace std; |
|  | void sort(int \*a , int n ); |
|  | void sort(double \*a , int n ); |
|  | int main() { |
|  | srand(time(0)); |
|  | int n,z; |
|  | cout<<"vvedite kolichestvo chisel"<<endl; |
|  | cin >> n; |
|  | cout<<"kakiye chisla budete vvodit(0-double/1-int)"<<endl; |
|  | cin>>z; |
|  | if (z == 0) { |
|  | double\*a = new double[n]; |
|  | for (int i = 0; i < n; i++) { |
|  | cout<<" vvedite chislo"<<endl; |
|  | cin >> a[i]; |
|  | } |
|  | sort(a, n); |
|  |  |
|  | for (int i = 0; i < n; i++) |
|  | cout << a[i] << endl; |
|  | } |
|  | else { |
|  | int\*a = new int[n]; |
|  | for (int i = 0; i < n; i++) { |
|  | cout<<" vvedite chislo"<<endl; |
|  | cin >> a[i]; |
|  | } |
|  | sort(a, n); |
|  |  |
|  | for (int i = 0; i < n; i++) |
|  | cout << a[i] << endl; |
|  | } |
|  | system("pause"); |
|  | return 0; |
|  |  |
|  | } |
|  |  |
|  | void sort(double \*a, int n) |
|  | { |
|  | long i, j, k; |
|  | cout<<"idet sortirovka drobnyh chisel"<<endl; |
|  | double x; |
|  |  |
|  | for (i = 0; i < n; i++) { // i - номер текущего шага |
|  | k = i; x = a[i]; |
|  |  |
|  | for (j = i + 1; j < n; j++) // цикл выбора наименьшего элемента |
|  | if (a[j] < x) { |
|  | k = j; x = a[j]; // k - индекс наименьшего элемента |
|  | } |
|  |  |
|  | a[k] = a[i]; a[i] = x; // меняем местами наименьший с a[i] |
|  | } |
|  | } |
|  | void sort(int \*a, int n) |
|  | { |
|  | long i, j, k; |
|  | cout<<"idet sortirovka celyh chisel"<<endl; |
|  | double x; |
|  |  |
|  | for (i = 0; i < n; i++) { // i - номер текущего шага |
|  | k = i; x = a[i]; |
|  |  |
|  | for (j = i + 1; j < n; j++) // цикл выбора наименьшего элемента |
|  | if (a[j] < x) { |
|  | k = j; x = a[j]; // k - индекс наименьшего элемента |
|  | } |
|  |  |
|  | a[k] = a[i]; a[i] = x; // меняем местами наименьший с a[i] |
|  | } |
|  | } |

1. Программа №3

|  |
| --- |
|  |
|  | #include <iostream>  #include <ctime> |
|  | using namespace std; |
|  |  |
|  | void fillArray(int \*\*, int, int); // Заполняет массив случайными числами |
|  |  |
|  | void printArray(int \*\*, int, int); // Выводит массив на экран |
|  |  |
|  | int main() { |
|  | srand(time(NULL)); |
|  |  |
|  | int m, n; |
|  | cout << "Enter number of rows and columns: "; |
|  | cin >> n >> m; |
|  |  |
|  | int \*\*arr = new int\*[n]; |
|  | for (int i = 0; i < n; i++) { |
|  | arr[i] = new int[m]; |
|  | } |
|  |  |
|  | fillArray(arr, n, m); |
|  | printArray(arr, n, m); |
|  |  |
|  | for (int i = 0; i < n; i++) { |
|  | delete [] arr[i]; |
|  | } |
|  | delete [] arr; |
|  |  |
|  | system("pause"); |
|  | return 0; |
|  | } |
|  |  |
|  | void fillArray(int \*\*arr, int n, int m) { |
|  | for (int i = 0; i < n; i++) { |
|  | for (int j = 0; j < m; j++) { |
|  | arr[i][j] = rand() % 41 + 10; |
|  | } |
|  | } |
|  | } |
|  |  |
|  | void printArray(int \*\*arr, int n, int m) { |
|  | cout << "Array:" << endl; |
|  | for (int i = 0; i < n; i++) { |
|  | for (int j = 0; j < m; j++) { |
|  | cout << arr[i][j] << ' '; |
|  | } |
|  | cout << endl; |
|  | } |
|  | } |

1. Программа №4

|  |
| --- |
|  |
|  | #include<ctime> |
|  | #include <iomanip> |
|  | using namespace std; |
|  |  |
|  | double calc(int a, int b); |
|  | double calc(double a, double b); |
|  |  |
|  | int main() { |
|  | int z; |
|  | cout << "1-int,2-double" << endl; |
|  | cin >> z; |
|  | if (z == 1) { |
|  | int a , b ; |
|  | cout << "vvedite 2 chisla" << endl; |
|  | cin >> a >> b; |
|  | cout << calc(a, b) << endl; |
|  | } |
|  | else { |
|  | double a, b; |
|  | cout << "vvedite 2 chisla" << endl; |
|  | cin >> a >> b; |
|  | cout << calc(a, b) << endl; |
|  | } |
|  | system("pause"); |
|  | return 0; |
|  | } |
|  |  |
|  | double calc(int a, int b) |
|  | { |
|  | int k; |
|  | do { |
|  | cout << "i1+2-3\*4/" << endl; |
|  | cin >> k; |
|  | switch (k) |
|  | { |
|  | case 1: |
|  | return a + b; |
|  | break; |
|  | case 2: |
|  | return a - b; |
|  | break; |
|  | case 3: |
|  | return a\*b; |
|  | break; |
|  | case 4: |
|  | return a / b; |
|  | default: |
|  | break; |
|  | } |
|  |  |
|  | } while (k != 0); |
|  | } |
|  |  |
|  | double calc(double a, double b) |
|  | { |
|  | int k; |
|  | do { |
|  | cout << "d1+2-3\*4/" << endl; |
|  | cin >> k; |
|  | switch (k) |
|  | { |
|  | case 1: |
|  | return a + b; |
|  | break; |
|  | case 2: |
|  | return a - b; |
|  | break; |
|  | case 3: |
|  | return a\*b; |
|  | break; |
|  | case 4: |
|  | return a / b; |
|  | default: |
|  | break; |
|  | } |
|  |  |
|  | } while (k != 0); |
|  | } |

**Вывод**

Все задачи были выполнены. В ходе выполнения была изучена техника перегрузки функций. Так же для заполнения массивов использовался генератор случайных чисел. Результат работы программы приведён на рисунках (рис. 15-18). Исходный код программ так же доступен на GitHub по ссылке:

<https://github.com/Manskript/Praktika>

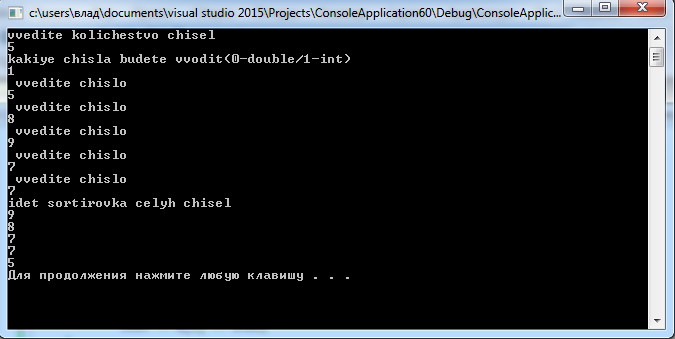


Рис. 15 Результат работы программы №1

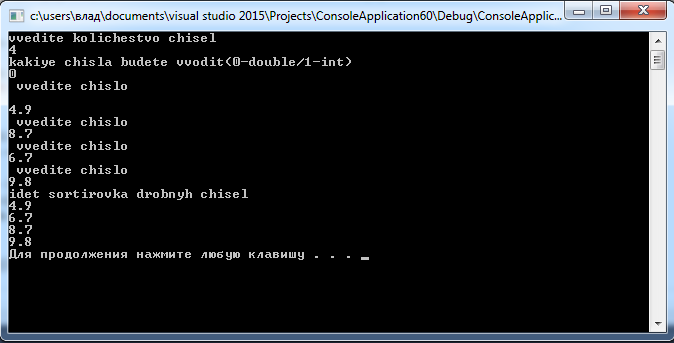


Рис. 16 Результат работы программы №2

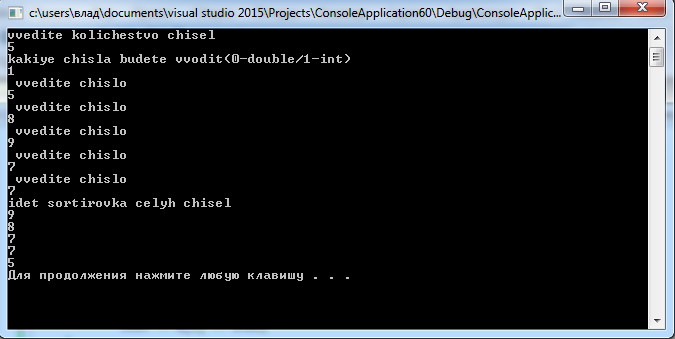


Рис. 17 Результат работы программы №3

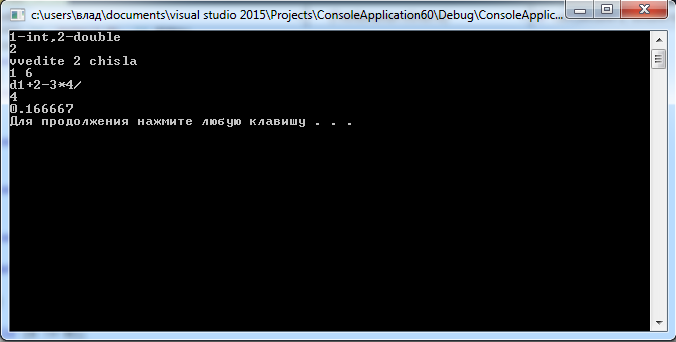


Рис. 18 Результат работы программы №4

## Практическая работа №6

**Решение задач на наследование на языке С++**

**Цель практической работы**

Целью данной практической работы является приобретение практических навыков использования принципа ООП – наследования для разработки программ на языке программирования C++.

**Задачи**

1. Написать иерархию классов, описывающих имущество налогоплательщиков. Она должна состоять из абстрактного базового класса Property и производных от него классов Appartment, Car и CountryHouse. Базовый класс должен иметь поле worth (стоимость), конструктор с одним параметром, заполняющий это поле, и чисто виртуальный метод рассчета налога, переопределенный в каждом из производных классов. Налог на квартиру вычисляется как 1/1000 ее стоимости, на машину – 1/200, на дачу – 1/500. Также, каждый производный класс должен иметь конструктор с одним параметром, передающий свой параметр конструктору базового класса. В функции main завести массив из 7 указателей на Property и заполнить его указателями на динамические объекты производных классов (первые 3 – Appartment, следующие 2 – Car и последние 2 – CountryHouse). Вывести на экран величину налога для всех 7 объектов. Не забудь также уничтожить динамические объекты перед завершением программы.
2. Написать набор классов, представляющий выражения. В этом наборе должен быть один абстрактный базовый тип, а также набор производных от него типов по видам выражений (константа, переменная, сумма, разность, произведение, частное, sin, cos, exp, ln). У каждого из классов должны быть следующие виртуальные функции: напечатать выражение (без параметров), вычислить выражение (параметр – значение переменной, результат – значение выражения), вернуть производную выражения (без параметров), создать копию выражения (тоже без параметров).

**Ход работы**

1. Программа №1

|  |
| --- |
|  |
|  | #include <iostream> |
|  | using namespace std; |
|  |  |
|  | class Property { |
|  | protected: |
|  | int worth; |
|  | public: |
|  | virtual double nalog() = 0; |
|  | Property(int worth) { |
|  | this->worth = worth; |
|  | } |
|  | virtual ~Property() {}; |
|  | }; |
|  |  |
|  | class Car: public Property |
|  | { |
|  | public: double nalog() { |
|  |  |
|  | return worth / 200; |
|  | } |
|  | Car(int worth) :Property(worth) { |
|  |  |
|  | } |
|  | }; |
|  |  |
|  | class CountryHouse : public Property |
|  | { |
|  | public: double nalog() { |
|  |  |
|  | return worth / 500; |
|  | } |
|  | CountryHouse(int worth) :Property(worth) { |
|  |  |
|  | } |
|  | }; |
|  |  |
|  | class Appartment : public Property |
|  | { |
|  | public: double nalog() { |
|  |  |
|  | return worth / 1000; |
|  | } |
|  | Appartment(int worth) :Property(worth) { |
|  |  |
|  | } |
|  | }; |
|  |  |
|  | int main() |
|  | { |
|  | Property\* a[7]; |
|  | for (int i = 0; i < 7; i++) { |
|  | int worth; |
|  | cout << "vvedite stoimost ob'ecta " << i + 1 << endl; |
|  | cin >> worth; |
|  |  |
|  | if (i < 3)a[i] = new Appartment(worth); |
|  | else if (i < 5)a[i] = new Car(worth); |
|  | else a[i] = new CountryHouse(worth); |
|  | cout << "nalog" << i + 1 << a[i]->nalog() << endl; |
|  | } |
|  | delete[] \*a; |
|  | system("pause"); |
|  | return 0; |
|  | } |

Программа №2:

**Вывод**

Все поставленные задачи были выполнены. В ходе выполнения работы был изучен механизм наследования классов в C++. Также для выполнения работы были использованы абстрактные классы и чисто виртуальные методы базовых абстрактных классов. Результаты работы программ представлены на рисунках (рис 19, 20). Исходный код программ также доступен на GitHub по ссылке:

<https://github.com/Manskript/Praktika>

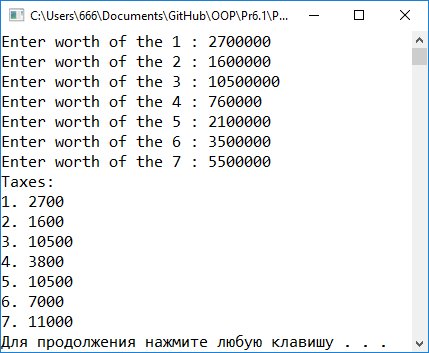


Рис. 19 Результат работы программы №1

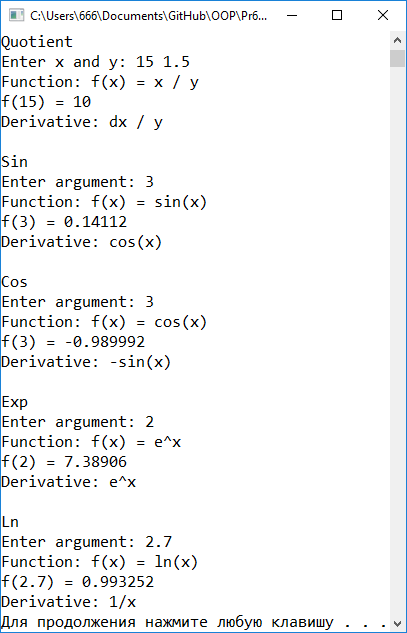
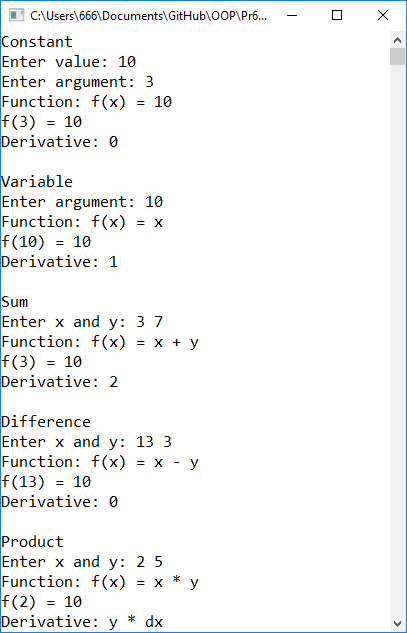


Рис. 20 Результат работы программы №2

## Практическая работа №7

**Абстрактные классы. Множественное наследование**

**Цель практической работы**

Целью данной практической работы является знакомство с абстрактными классами и механизмом виртуальных функций, а так же множественным наследованием на языке программирования C++.

**Задачи**

1. Реализовать абстрактный класс «Животное» и путём наследования от него получить классы «Кошка», «Собака», «Попугай».
2. Реализовать абстрактный класс «Фигура» и путём наследования от него получить абстрактный класс «Четырёхугольник», и затем путём наследования получить классы «Ромб», «Прямоугольник».
3. Реализовать абстрактный класс «Транспортное средство» и путём наследования от него получить классы «Автомобиль», «Автобус», «Велосипед».
4. Реализовать абстрактные классы «Экран» и «Клавиатура», путём наследования от них получить классы «Ноутбук», «Телефон», «Стационарный компьютер».

**Ход работы**

1. Программа №1

|  |
| --- |
|  |
|  | #include<string> |
|  |  |
|  | using namespace std; |
|  |  |
|  | class Animal { |
|  | protected: |
|  | string name; |
|  | int year; |
|  | public: |
|  | Animal(string name, int year) { |
|  | this->name = name; |
|  | this->year = year; |
|  | } |
|  | Animal() { |
|  | cout<<"vvedite imya i kolichestvo let"<<endl; |
|  | cin >> name; |
|  | cin >> year; |
|  | } |
|  | virtual void golos() = 0; |
|  | void get\_Data() { |
|  | cout << "Ya " << name << " " << year << endl; |
|  | } |
|  | }; |
|  |  |
|  | class Cat: public Animal { |
|  | public: |
|  | Cat(string name, int year):Animal(name,year){} |
|  | Cat():Animal(){} |
|  | void golos() { |
|  | cout << "myaaaaaau!!!"<<endl; |
|  | } |
|  |  |
|  | }; |
|  |  |
|  | class Dog : public Animal { |
|  | public: |
|  | Dog(string name, int year) :Animal(name, year) {} |
|  | Dog() :Animal() {} |
|  | void golos() { |
|  | cout << "gav gav gav!!!" << endl; |
|  | } |
|  |  |
|  | }; |
|  |  |
|  | class Parrot : public Animal { |
|  | public: |
|  | Parrot(string name, int year) :Animal(name, year) {} |
|  | Parrot() :Animal() {} |
|  | void golos() { |
|  | cout << "karrrr!!!" << endl; |
|  | } |
|  |  |
|  | }; |
|  | int main(){ |
|  | Cat a; |
|  | a.get\_Data(); |
|  | a.golos(); |
|  |  |
|  | Dog b; |
|  | b.get\_Data(); |
|  | b.golos(); |
|  |  |
|  | Parrot c; |
|  | c.get\_Data(); |
|  | c.golos(); |
|  |  |
|  |  |
|  |  |
|  |  |
|  | system("pause"); |
|  | return 0; |
|  | } |

1. Программа №2

|  |
| --- |
|  |
|  | #include <iostream> |
|  | #include <cmath> |
|  | using namespace std; |
|  |  |
|  |  |
|  | class Forug{ |
|  | protected: |
|  | int a1, a2, a3, a4; |
|  | public: |
|  | Forug() { |
|  | cout << "vvedite dliny storon" << endl; |
|  | cin >> a1 >> a2 >> a3 >> a4; |
|  | } |
|  | virtual void sqaire() = 0; |
|  |  |
|  |  |
|  | }; |
|  | class Romb:public Forug { |
|  | public: |
|  | double sinus; |
|  | Romb() :Forug() { |
|  | cout << "vvedite men'shiy ugol" << endl; |
|  | cin >> sinus; |
|  | } |
|  | void sqaire() { |
|  | cout << a1\*a1\*sin(sinus)<<endl;; |
|  | } |
|  | }; |
|  |  |
|  | class Pryamougolnic :public Forug { |
|  | public: |
|  | Pryamougolnic() :Forug() { |
|  |  |
|  | } |
|  | void sqaire() { |
|  | cout<< a1\*a3<<endl; |
|  | } |
|  | }; |
|  | class Figure { |
|  | public: |
|  |  |
|  | void sqaire(Forug \*forug) { |
|  | cout << "ploshad ravna:"; |
|  | forug->sqaire(); |
|  | } |
|  |  |
|  | }; |
|  |  |
|  | int main() |
|  | { |
|  | Pryamougolnic c; |
|  | Romb b; |
|  | Figure a; |
|  | a.sqaire(&b); |
|  | a.sqaire(&c); |
|  | system("pause"); |
|  | return 0; |
|  | } |

1. Программа №3

|  |
| --- |
|  |
|  | #include <iostream> |
|  | #include <cmath> |
|  | #include <string> |
|  | using namespace std; |
|  |  |
|  | class Vehicle { |
|  | protected:int massa; |
|  | int skorost; |
|  | string name; |
|  | public:Vehicle() { |
|  | cout << "vvedite massu skorost i nazvanie" << endl; |
|  | cin >> massa >> skorost >> name; |
|  | } |
|  | virtual void kto\_ya() = 0; |
|  | }; |
|  | class Bycicle:public Vehicle { |
|  | public: |
|  | Bycicle() :Vehicle() { } |
|  | void kto\_ya() { |
|  | cout << "ya velosiped:" << name << ' ' << skorost << "km/h " << massa << "kg"<<endl; |
|  | } |
|  |  |
|  | }; |
|  |  |
|  | class Bus :public Vehicle { |
|  | public: |
|  | Bus() :Vehicle() {} |
|  | void kto\_ya() { |
|  | cout << "ya avtobus:" << name << ' ' << skorost << "km/h " << massa << "kg" << endl; |
|  | } |
|  |  |
|  | }; |
|  |  |
|  | class Avto :public Vehicle { |
|  | public: |
|  | Avto() :Vehicle() { } |
|  | void kto\_ya() { |
|  | cout << "ya mashina:" << name << ' ' << skorost << "km/h " << massa << "kg" << endl; |
|  | } |
|  |  |
|  | }; |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  | int main() |
|  | { |
|  | string z; |
|  | cout << "vvedite transportnoe sredstdo (bus, bycicle, avto)" << endl; |
|  | cin >> z; |
|  | if(z=="bus"){ Bus a; a.kto\_ya(); |
|  | } |
|  | else if (z == "avto") { |
|  | Avto a; a.kto\_ya(); |
|  | } |
|  | else if (z == "bycicle") { |
|  | Bycicle a; a.kto\_ya(); |
|  | } |
|  | system("pause"); |
|  | return 0; |
|  | } |

1. Программа №4

|  |
| --- |
|  |
|  | #include <iostream> |
|  | using namespace std; |
|  |  |
|  | class Screen { // Базовый абстрактный класс "Экран" |
|  | protected: |
|  | double razr; |
|  | public: |
|  | Screen(double size) { this->razr = size; } |
|  |  |
|  | virtual void izmScreen(double size) = 0; // Виртуальный метод заены экрана |
|  |  |
|  |  |
|  | }; |
|  |  |
|  | class Keyboard { // Базовый абстрактный класс "Клавиатура" |
|  | protected: |
|  | int buttons; |
|  | public: |
|  | Keyboard(int buttons) { this->buttons = buttons; } |
|  |  |
|  | virtual void izmKeyboard(int buttons) = 0; // Виртуальный метод замены клавиатуры |
|  | }; |
|  |  |
|  | class Computer : public Screen, Keyboard { // Класс "Стационарный компьютер" |
|  | public: |
|  | Computer(double razr, int buttons) : Screen(razr), Keyboard(buttons) {}; |
|  |  |
|  | void izmScreen(double size) { |
|  | cout << "ekran " << razr << |
|  | " byl izmenen na " << size << " razmer" << endl; |
|  | razr = size; |
|  | } |
|  |  |
|  | void izmKeyboard(int buttons) { |
|  | cout << "klaviatura s " << this->buttons << " knopkami " << |
|  | "byla izmenena na " << buttons << " knopok" << endl; |
|  | this->buttons = buttons; |
|  | } |
|  |  |
|  | ~Computer() {} |
|  | }; |
|  |  |
|  | class Notebook : public Screen, Keyboard { // Класс "Ноутбук" |
|  | public: |
|  | Notebook(double razr, int buttons) : Screen(razr), Keyboard(buttons) {}; |
|  |  |
|  | void izmScreen(double size) { |
|  | if (razr == size) { |
|  | cout << "ekran byl izmenen" << endl; |
|  | } |
|  | else { |
|  | cout << "ekran nouta ne byl izmenen" << endl; |
|  | } |
|  | } |
|  |  |
|  | void izmKeyboard(int buttons) { |
|  | if (this->buttons == buttons) { |
|  | cout << "klaviatura byla izmenena" << endl; |
|  | } |
|  | else { |
|  | cout << "klaviatura u niutbuka ne byla izmenena" << endl; |
|  | } |
|  | } |
|  |  |
|  | ~Notebook() {} |
|  | }; |
|  |  |
|  | class Phone : public Screen, Keyboard { // Класс "Телефон" |
|  | public: |
|  | Phone(double razr, int buttons) : Screen(razr), Keyboard(buttons) {}; |
|  |  |
|  | void izmScreen(double size) { |
|  | if (razr == size) { |
|  | cout << "ekran telefona byl izmenen" << endl; |
|  | } |
|  | else { |
|  | cout << "ekran telefona ne byl izmenen" << endl; |
|  | } |
|  | } |
|  |  |
|  | void izmKeyboard(int buttons) { |
|  | cout << "klaviatura telefona ne mozet byt izmenena" << endl; |
|  | } |
|  |  |
|  | ~Phone() {} |
|  | }; |
|  |  |
|  | int main() { |
|  |  |
|  | int z=1; |
|  | do { |
|  | cout << "s kakim ustroistvom budem rabokat 1-telefon, 2- notebook, 3- computer" << endl; |
|  | int e, r; |
|  | cin >> z; |
|  | switch (z) |
|  | { |
|  | case 1: {cout << "vvedite kolvo razmer ekrana i kolichestvo knopok" << endl; |
|  |  |
|  | cin >> e >> r; |
|  | Phone a(e, r); |
|  | cout << "vvedize na skol'ko vy hotite izmenit razmer ekrana i kolichestvo knopok" << endl; |
|  | cin >> e >> r; |
|  | a.izmKeyboard(e); |
|  | a.izmScreen(r); |
|  | break; } |
|  | case 2: {cout << "vvedite kolvo razmer ekrana i kolichestvo knopok" << endl; |
|  |  |
|  | cin >> e >> r; |
|  | Notebook b(e, r); |
|  | cout << "vvedize na skol'ko vy hotite izmenit razmer ekrana i kolichestvo knopok" << endl; |
|  | cin >> e >> r; |
|  | b.izmKeyboard(e); |
|  | b.izmScreen(r); |
|  | break; } |
|  | case 3: {cout << "vvedite kolvo razmer ekrana i kolichestvo knopok" << endl; |
|  |  |
|  | cin >> r >> e; |
|  | Computer c(e, r); |
|  | cout << "vvedize na skol'ko vy hotite izmenit razmer ekrana i kolichestvo knopok" << endl; |
|  | cin >> e >> r; |
|  | c.izmKeyboard(e); |
|  | c.izmScreen(r); |
|  | break; } |
|  | default: |
|  | break; |
|  | } |
|  |  |
|  | } while (z != 0); |
|  | system("pause"); |
|  | return 0; |
|  | } |

**Вывод**

Все задачи были выполнены. В ходе выполнения работы были более плотно изучены множественное наследование и абстрактные классы. Для достижения поставленной цели были использованы различные STL контейнеры и библиотека cmath. Результаты работы программ представлены на изображениях (рис. 21-24). Исходный кодвыполненных програм также доступе но ссылка на GitHub:

<https://github.com/Manskript/Praktika>

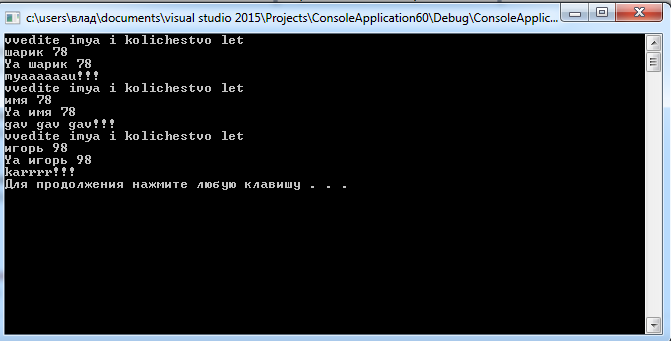
****

Рис. 21 Результат работы программы №1

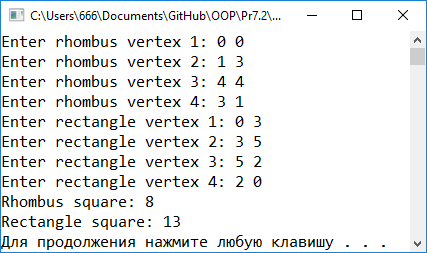


Рис. 22 Результат работы программы №2

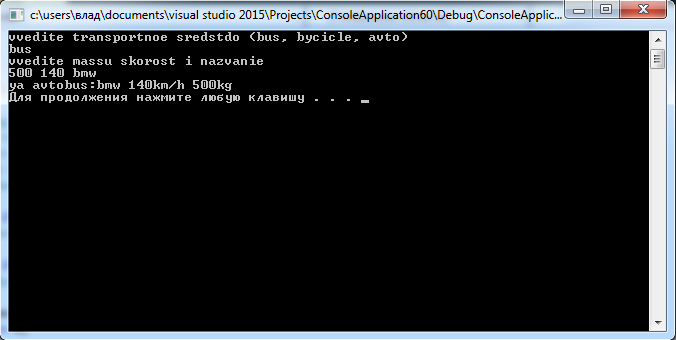


Рис. 23 Результат работы программы №3

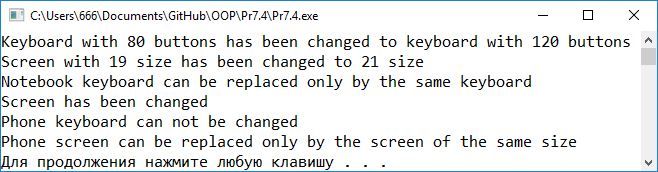


Рис. 24 Результат работы программы №4

## Практическая работа №8

**Работа с файлами в языке С++**

**Цель практической работы**

Целью данной практической работы является приобретение практических навыков по работе с файлами на языке программирования языке C++.

**Задачи**

1. Реализуйте программу, считывающую текст из файла и выводящую каждое слово с новой строки.
2. Реализуйте программу, считывающую текст с клавиатуры и записывающую его в файл.

**Ход работы**

1. Программа №1

|  |
| --- |
|  |
|  | #include <fstream> |
|  | #include <iostream> |
|  | #include <string> |
|  | using namespace std; |
|  |  |
|  | int main(int argc, char\* argv[]) |
|  | { |
|  | string s; |
|  | cin >> s; |
|  | ofstream fout("fff.txt"); // создаём объект класса ofstream для записи и связываем его с файлом cppstudio.txt |
|  | fout << s; // запись строки в файл |
|  | fout.close(); // закрываем файл |
|  | system("pause"); |
|  | return 0; |
|  | } |

1. Программа №2

|  |
| --- |
|  |
|  | #include <fstream> |
|  | #include <iostream> |
|  | #include <string> |
|  | using namespace std; |
|  |  |
|  | int main() |
|  | { |
|  | setlocale(LC\_ALL, "rus"); |
|  | string s; |
|  | ifstream file("fff.txt"); |
|  |  |
|  | if (file.is\_open()){ // вызов метода is\_open() |
|  | cout << "Все ОК! Файл открыт!\n\n" << endl; |
|  | for (file >> s; !file.eof(); file >> s) |
|  | cout << s << endl; |
|  | } |
|  | else |
|  | { |
|  | cout << "Файл не открыт!\n\n" << endl; |
|  | return -1; |
|  | } |
|  | system("pause"); |
|  | return 0; |
|  | } |

**Вывод**

Все задачи были выполнены успешно. В ходе работы была исследована работа с файлами. Для достижения результата использовалась библиотек fstream, позволяющая производить работу с файлами. Результаты работ программ приведены на рисунках (рис. 25, 26). Также исходный код програм доступен по ссылкам на GitHub:

<https://github.com/Manskript/Praktika>

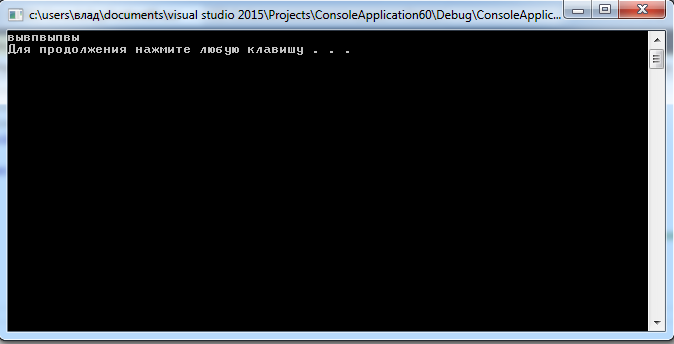


Рис. 25 Результат работы программы №1

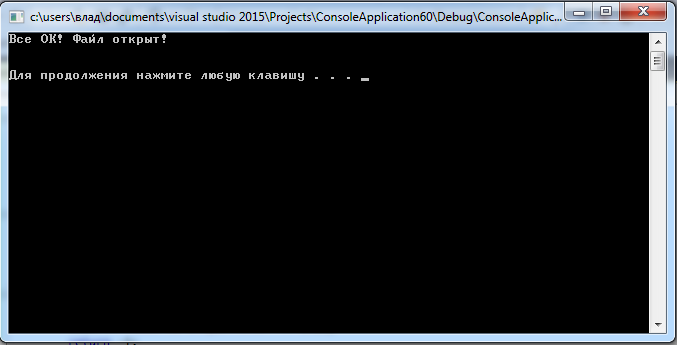


Рис. 26 Результат работы программы №2

## 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 рисунках (рис. 27, 28). Исходный код также доступен по ссылкам на GitHub:

<https://github.com/Manskript/Lab_1_1>

<https://github.com/Manskript/Lab_1_2>

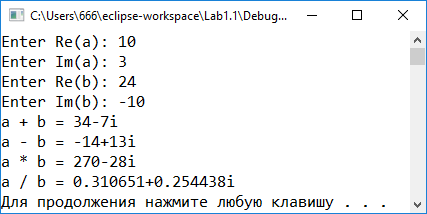
****

Рис. 27 Результат работы программы №1

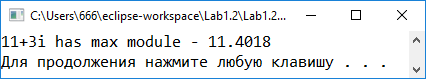


Рис. 28 Результат работы программы №2

## 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 рисунках (рис. 29, 30). Исходный код программ также доступен по ссылке на GitHub:

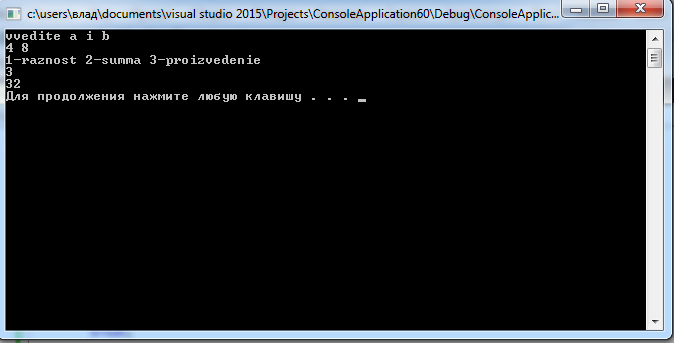
<https://github.com/Manskript/Other_labs_report>

Рис. 29 Результат работы программы №1

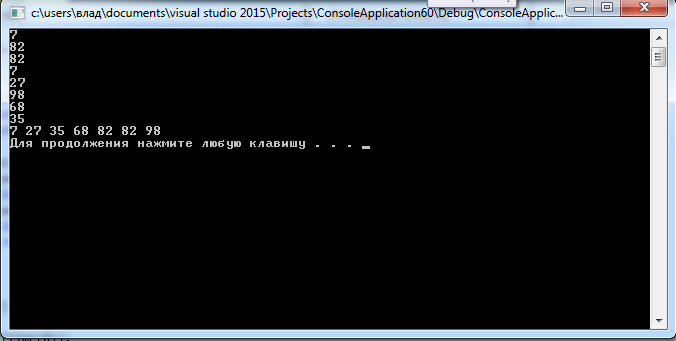


Рис. 30 Результат работы программы №2

## 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 рисунках (рис. 31-34). Исходные коды программ также доступны по ссылке на GitHub:

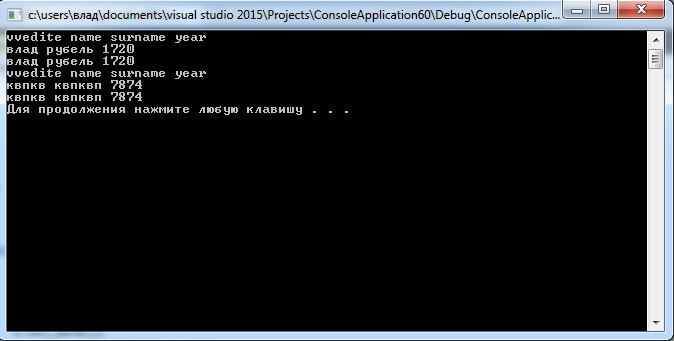
<https://github.com/Manskript/Other_labs_report> ****

Рис. 31 Результат работы программы №1

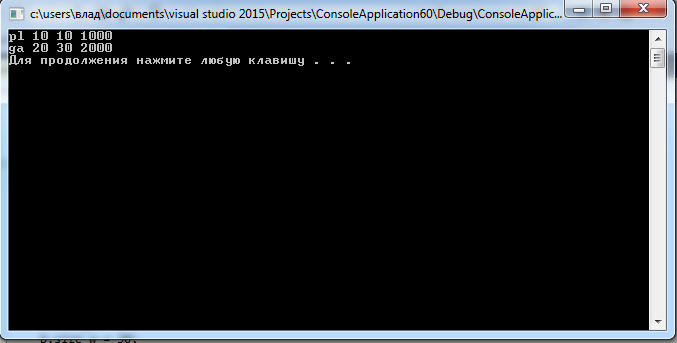
****

Рис. 32 Результат работы программы №2

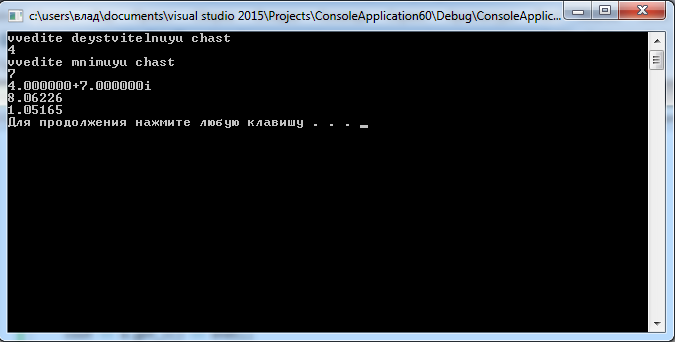
****

Рис. 33 Результат работы программы №3

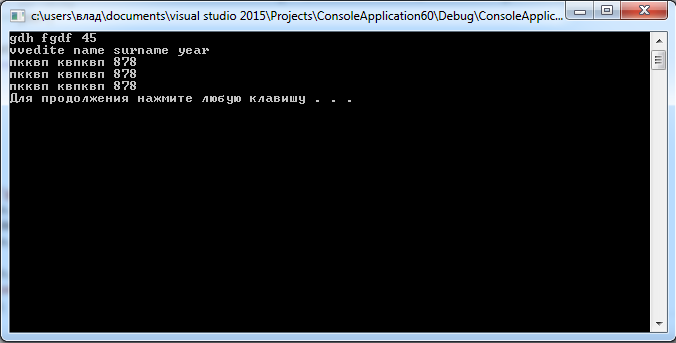
****

Рис. 34 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 рисунках (рис. 35-38). Исходный код программ также доступен по ссылке на GitHub:

<https://github.com/Manskript/Other_labs_report>

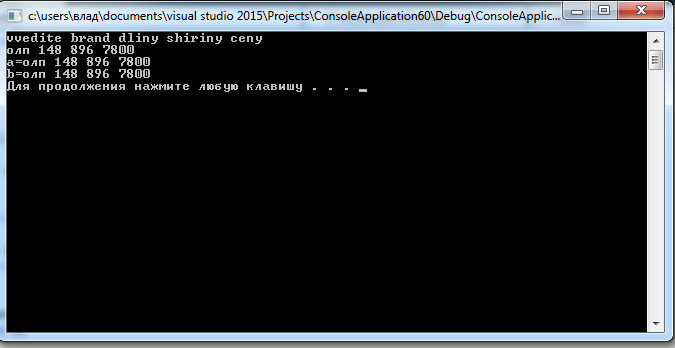


Рис. 35 Результат работы программы №1

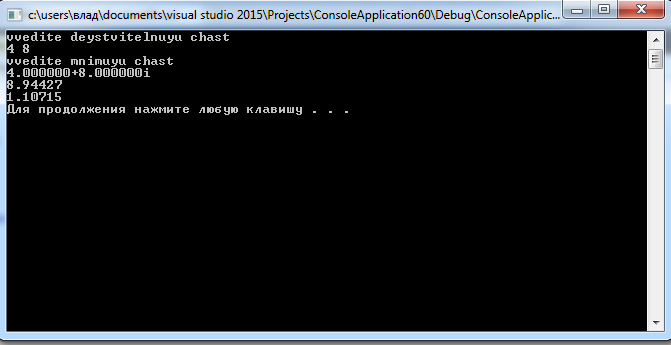


Рис. 36 Результат работы программы №2

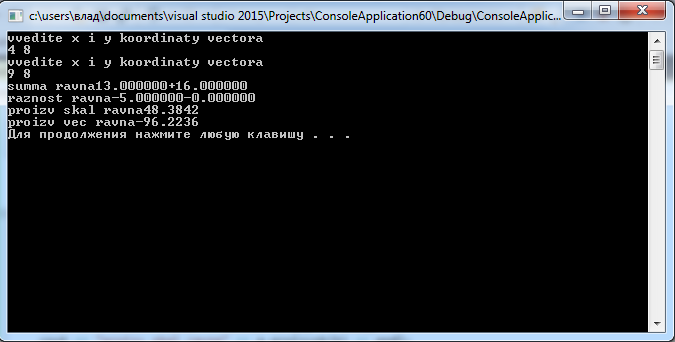


Рис. 37 Результат работы программы №3

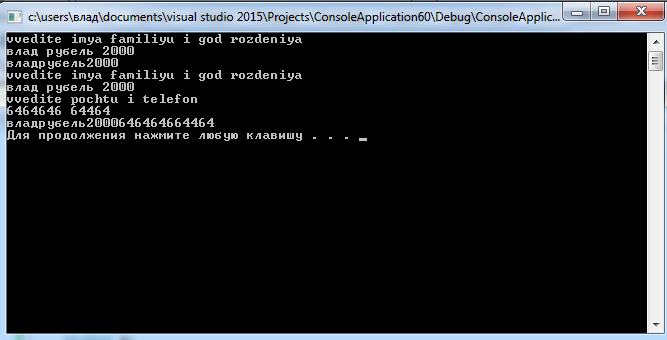


Рис. 38 Результат работы программы №4

## 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 рисунках (рис. 39, 40). Исходный код программ также доступен по ссылке на GitHub:

<https://github.com/Manskript/Other_labs_report>

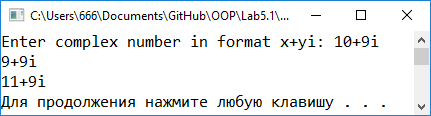


Рис. 39 Результат работы программы №1

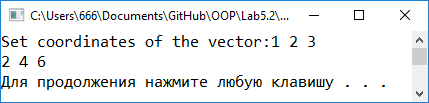


Рис. 40 Результат работы программы №2

## 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 рисунках (рис. 41-43). Исходный код программ также можно просмотреть на GitHub:

<https://github.com/Manskript/Other_labs_report>

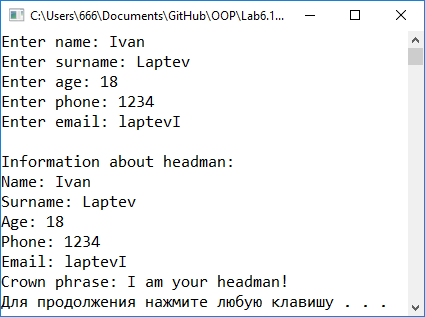


Рис. 41 Результат работы программы №1

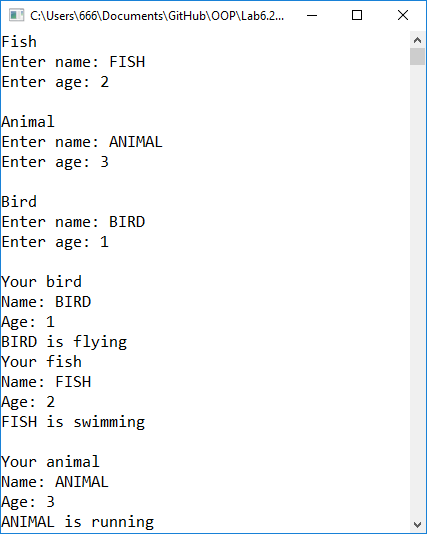


Рис. 42 Результат работы программы №2

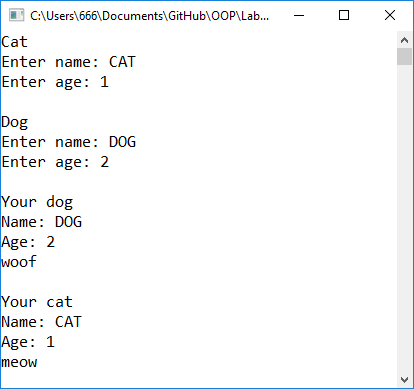


Рис. 43 Результат работы программы №3

## 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 рисунке (рис. 44). Код программы также доступен на GitHub по ссылке:

<https://github.com/Manskript/Other_labs_report>

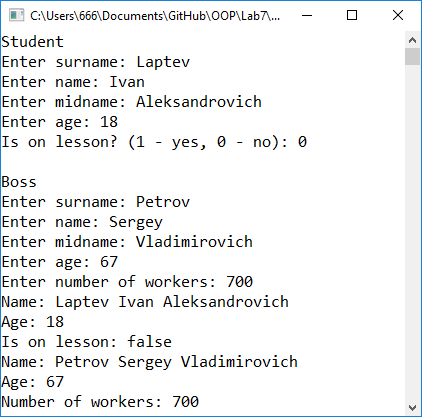


Рис. 44 Результат работы программы

## 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 рисунках (рис. 45-47). Исходный код программ так же доступен на GitHub по ссылке:

<https://github.com/Manskript/Other_labs_report>

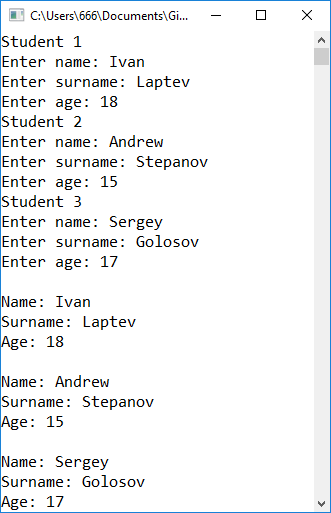


Рис. 45 Результат работы программы №1

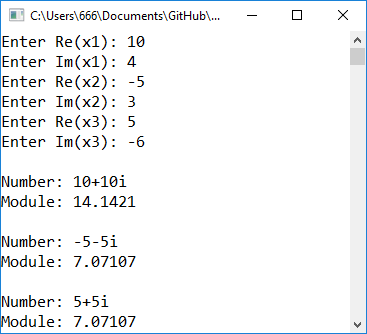


Рис. 46 Результат работы программы №2

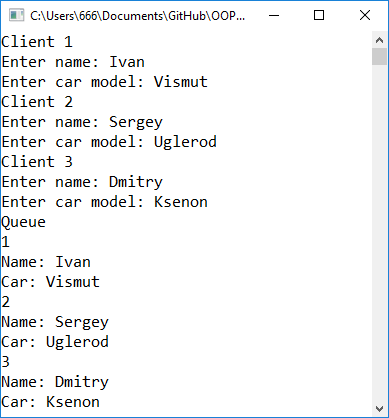


Рис. 47 Результат работы программы №3