МИНОБРНАУКИ РОССИИ

Федеральное государственное бюджетное образовательное учреждение высшего образования

«Чувашский государственный университет имени И. Н. Ульянова»

Факультет информатики и вычислительной техники

Кафедра математического и аппаратного обеспечения

информационных систем

Расчетно-графическая работа по дисциплине «Языки программирования»

на тему:

«Проектирование и реализация класса «Сотрудники в институте»»

Выполнил:

Студент группы ИВТ-21-22

Иванов Никита Андреевич

Проверил:

Ст. преподаватель Ильина Л.А.

Оценка\_\_\_\_\_\_\_\_\_\_\_\_\_

Дата\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Чебоксары 2022

# Содержание

[**Содержание**](#_x29fii96vjjk) **1**

[**Введение**](#_kvaiok47wa5x) **3**

[**Постановка задачи**](#_vm4i0x5j6w36) **5**

[**Теоретическая часть**](#_myqnasw0pgdd) **6**

[**Практическая часть**](#_jeeivabj2p0k) **7**

[Создаю класс «Inst»](#_3zy93i6y6n2) 8

[Контекстное меню.](#_rh3q7ng9rrhg) 8

[Результат](#_m1gdo9c10nm9) 11

[**Заключение**](#_u3bq4qz3o15) **14**

[**Список использованной литературы**](#_5bxsjqpyml60) **15**

[**Приложение 1**](#_t723ad7dymkq)

[**Приложение 2**](#_1izs9do45vg0)

# Введение

Объектно-ориентированное программирование (сокр. ООП) — методология программирования, основанная на представлении программы в виде совокупности объектов, каждый из которых является экземпляром определенного класса, а классы образуют иерархию наследования. Идеологически ООП — подход к программированию как к моделированию информационных объектов, решающий на новом уровне основную задачу структурного программирования: структурирование информации с точки зрения управляемости, что существенно улучшает управляемость самим процессом моделирования, что, в свою очередь, особенно важно при реализации крупных проектов. Управляемость для иерархических систем предполагает минимизацию избыточности данных (аналогичную нормализации) и их целостность, поэтому созданное удобно управляемым — будет и удобно пониматься. Таким образом, через тактическую задачу управляемости решается стратегическая задача — транслировать понимание задачи программистом в наиболее удобную для дальнейшего использования форму.

Основные принципы структурирования в случае ООП связаны с различными аспектами базового понимания предметной задачи, которое требуется для оптимального управления соответствующей моделью:

Абстракция для выделения в моделируемом предмете важного для решения конкретной задачи по предмету, в конечном счёте — контекстное понимание предмета, формализуемое в виде класса; инкапсуляция для быстрой и безопасной организации собственно иерархической управляемости: чтобы было достаточно простой команды «что делать», без одновременного уточнения как именно делать, так как это уже другой уровень управления;

Наследование для быстрой и безопасной организации родственных понятий: чтобы было достаточно на каждом иерархическом шаге учитывать только изменения, не дублируя всё остальное, учтенное на предыдущих шагах; полиморфизм для определения точки, в которой единое управление лучше распараллелить или наоборот — собрать воедино.

Целью расчетно-графической работы (РГР) является закрепление практических навыков самостоятельной работы и отладки любой сложности программного изделия с применением современных навыков и методов программирования; овладение современными методами наглядного представления результатов решения, организации диалога с ЭВМ.

# 

# Постановка задачи

Разработка и реализация класса содержащий сведения о сотрудниках

института: фамилия, имя, отчество, название отдела, год рождения, стаж работы, должность, оклад.

Написать программу, позволяющую получить следующую информацию: список сотрудников пенсионного возраста на текущую дату с указанием стажа работы, средний стаж работающих в отделе Х. Данные и результаты должны храниться в файлах.

Предусмотреть возможность добавления информации, удаления отдельных записей, сортировки по заданному полю.

# Теоретическая часть

Создаем класс, содержащий сведения о сотрудниках института.

В модификатор доступа private создаем переменные: last\_name (фамилия), first\_name (имя), middle\_name (отчество), departament\_name(отдел), birth\_year (год рождения), work\_exp (стаж работы), salary (зарплата), tbl\_nomer (табельный номер для реализации функции удаления).

В открытой части класса “public” создаем функции для получения значений и их редактирования и конструктор для табельного номера.

Сделать возможность добавления информации:

* создаем объект типа класс
* введем данные из консоли
* считываем объект в файл

Сотрудник добавлен

Удаления отдельных записей (удаление через табельный номер):

* проверяем что номер не совпадает
* присваиваем пустой объект типа класс в объект типа класс
* считываем вектор в файл (упуская объект с совпадающим номером)

Сотрудник удалена.

Считывание данных с файла.

Вывод в файл. На выводе использую поток fstream c с режимом создания файла и добавлением в конец.

# Практическая часть

## Создаю класс «Inst»

В модификатор доступа private создаю переменные:

Переменные string last\_name, first\_name, middle\_name, department\_name, position.

Данные переменные хранят информацию о фамилии, имени, отчестве, названии департамента и должности сотрудника.

Переменные int birth\_year, work\_experience, salary, tbl\_nomer.

Данные переменные хранят информацию о годе рождения, опыте работы, заработной плате и номере табеля учета сотрудника.

Статическая переменная int count объявлена без инициализации значения. Она будет существовать в памяти на протяжении жизни программы и будет общей для всех объектов класса, т.е. ее значение будет совпадать для всех экземпляров класса.

Данная переменная может использоваться для отслеживания количества созданных объектов данного класса.

В модификатор доступа public создаю методы:

public:

InstituteEmployee() {

tbl\_nomer = ++count;

}

InstituteEmployee(string l\_name, string f\_name, string m\_name, string dep\_name, int b\_year, int w\_exp, string pos, int salary) :

last\_name(l\_name), first\_name(f\_name), middle\_name(m\_name), department\_name(dep\_name),

birth\_year(b\_year), work\_experience(w\_exp), position(pos), salary(salary), tbl\_nomer(++count) {}

string getLast\_name() const {

return last\_name;

}

void setLast\_name(string last\_name) {

this->last\_name = last\_name;

}

string getFirst\_name() const {

return first\_name;

}

void setFirst\_name(string first\_name) {

this->first\_name = first\_name;

}

string getMiddle\_name() const {

return middle\_name;

}

void setMiddle\_name(string middle\_name) {

this->middle\_name = middle\_name;

}

string getDepartment\_name() const {

return department\_name;

}

void setDepartment\_name(string department\_name) {

this->department\_name = department\_name;

}

int getBirth\_year() const {

return birth\_year;

}

void setBirth\_year(int birth\_year) {

this->birth\_year = birth\_year;

}

int getWork\_experience() const {

return work\_experience;

}

void setWork\_experience(int work\_experience) {

this->work\_experience = work\_experience;

}

string getPosition() const {

return position;

}

void setPosition(string position) {

this->position = position;

}

int getSalary() const {

return salary;

}

void setSalary(int salary) {

this->salary = salary;

}

int getTbl\_nomer() const {

return tbl\_nomer;

}

};

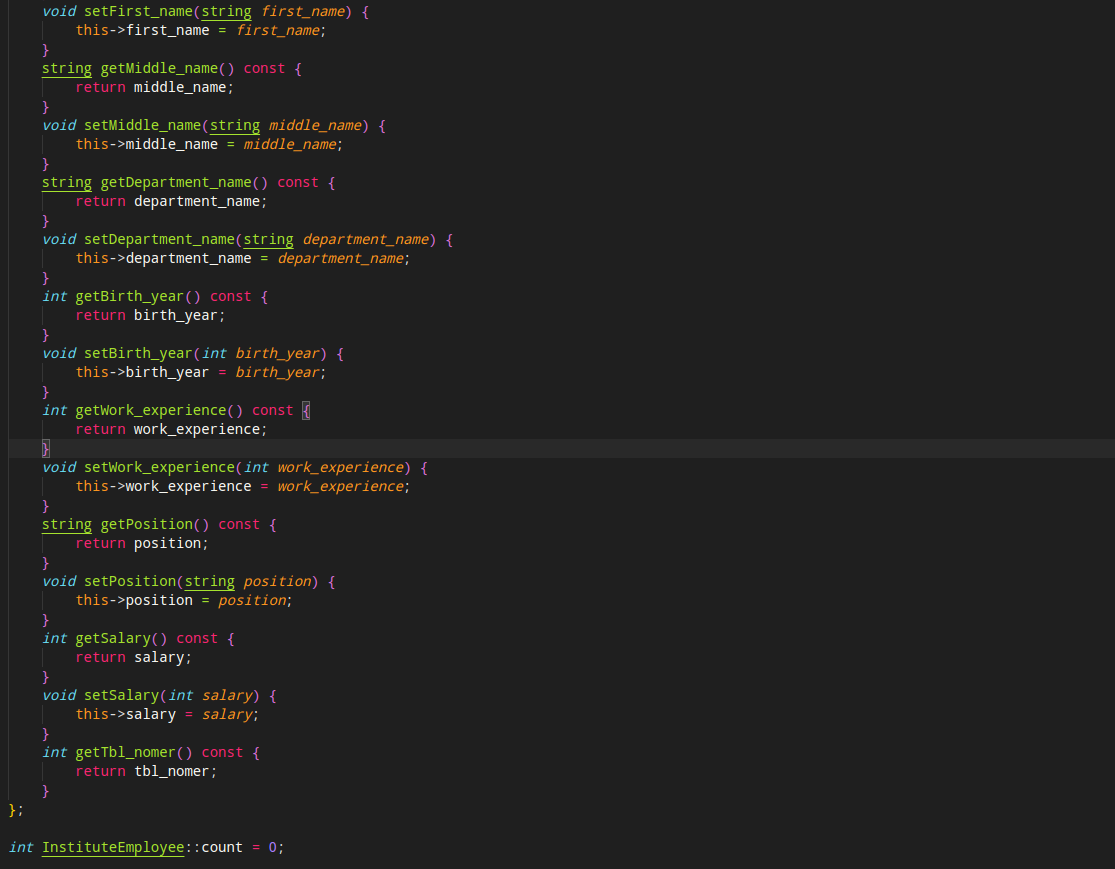
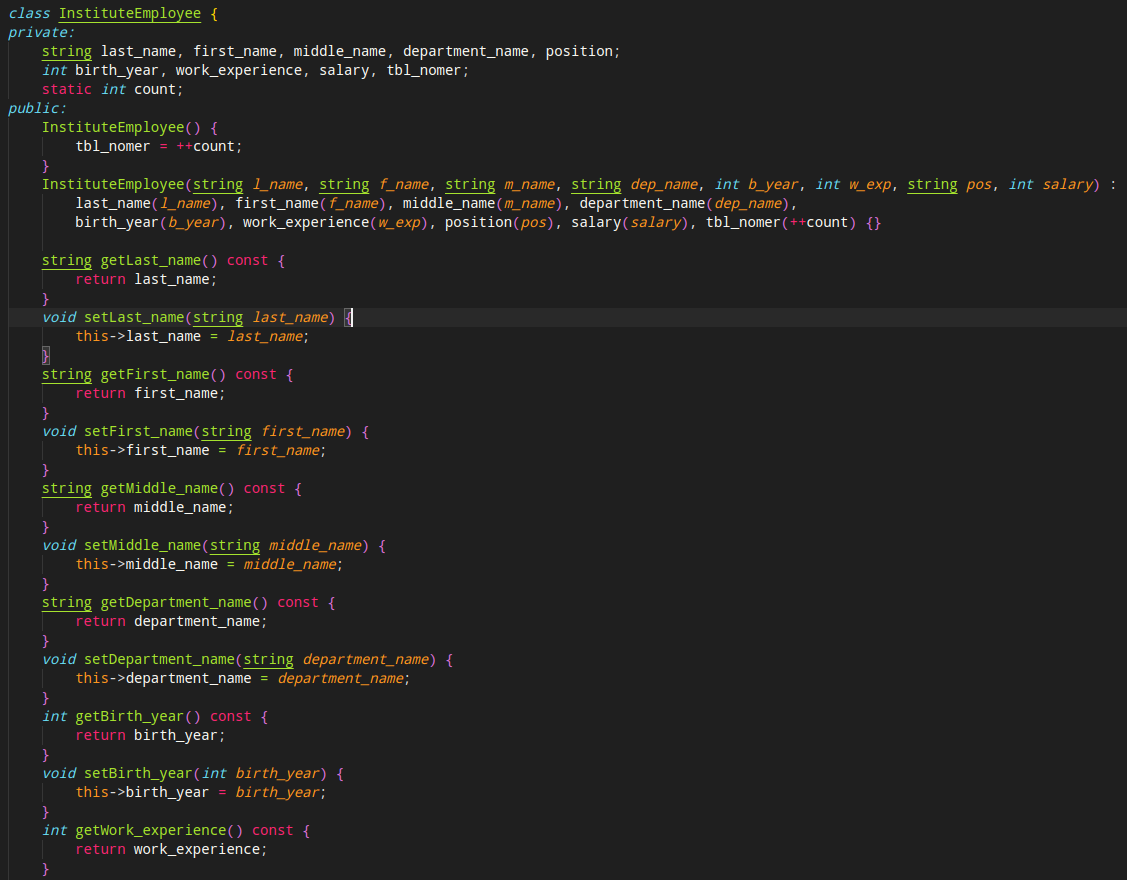
Выше приведен код определения класса `InstituteEmployee`.

- `InstituteEmployee()` - это конструктор по умолчанию, который присваивает значение переменной `tbl\_nomer` следующему значению `count` (что означает, что каждый раз, когда создается экземпляр класса, у него будет уникальный номер в таблице);

- `InstituteEmployee(string l\_name, string f\_name, string m\_name, string dep\_name, int b\_year, int w\_exp, string pos, int salary)` - это конструктор с параметрами, который инициализирует поля класса соответствующими значениями. В конце он также присваивает значение переменной `tbl\_nomer` следующему значению `count`.

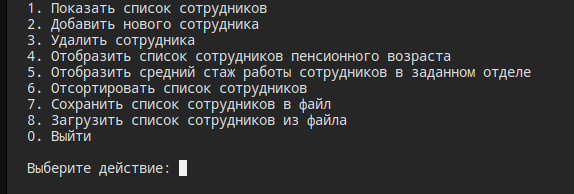
- `getLast\_name(), setLast\_name(), getFirst\_name(), setFirst\_name(), getMiddle\_name(), setMiddle\_name(), getDepartment\_name(), setDepartment\_name(), getBirth\_year(), setBirth\_year(), getWork\_experience(), setWork\_experience(), getPosition(), setPosition(), getSalary(), setSalary(), getTbl\_nomer()` - это методы доступа (геттеры и сеттеры) для членов класса.

- `static int InstituteEmployee::count = 0;` - это определение статической переменной `count` класса `InstituteEmployee`, которая используется для определения уникального номера `tbl\_nomer`. Она инициализируется значением `0`.



### Контекстное меню.

### (рис.2 “основное контекстное меню”)



1. Показать список сотрудников (1)

2. Добавить нового сотрудника (2)

3. Удалить сотрудника (3)

4. Отобразить список сотрудников пенсионного возраста (4)

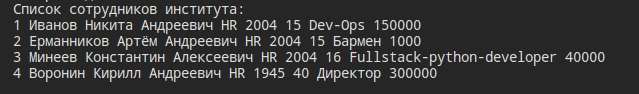
5. Отобразить средний стаж работы сотрудников в заданном отделе (5)

6. Отсортировать список сотрудников(6)

7. Сохранить список сотрудников в файл(7)

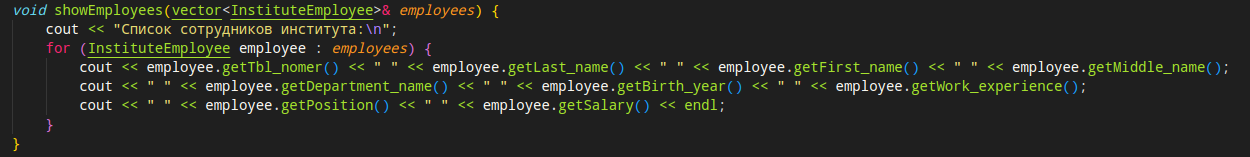
8. Загрузить список сотрудников из файла(8)

1. Показать список сотрудников:



После вывода программа возвращается в контекстное меню (рис.2).

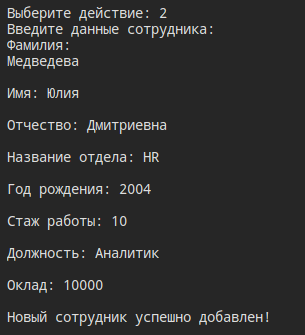
Как это выглядит в коде:



После, программа возвращается в контекстное меню.

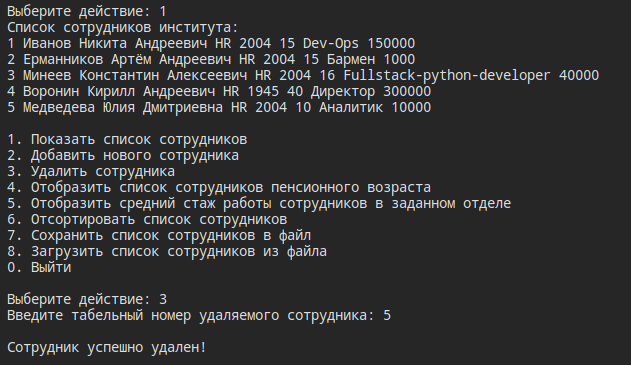
2. Добавить сотрудника.

При нажатии на (2) нужно заполнить некоторые данные: ФИО,Название отдела, год рождения , Стаж , Должность , Оклад (P.s. Табельный номер автоматически создаётся) (рис.3)



После, программа возвращается в контекстное меню.

3. Удаление сотрудника: Мы просто вводим табельный номер который мы можем узнать при показе списка сотрудника(рис.2) и после этого вводим просто табельный номер для удаления сотрудника вот как это выглядит:

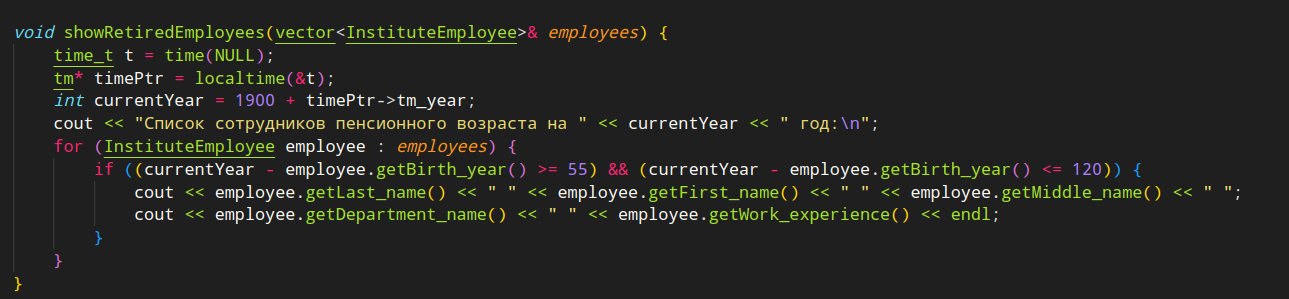


После, программа возвращается в контекстное меню.

4. Отображение списка сотрудников пенсионного возраста:

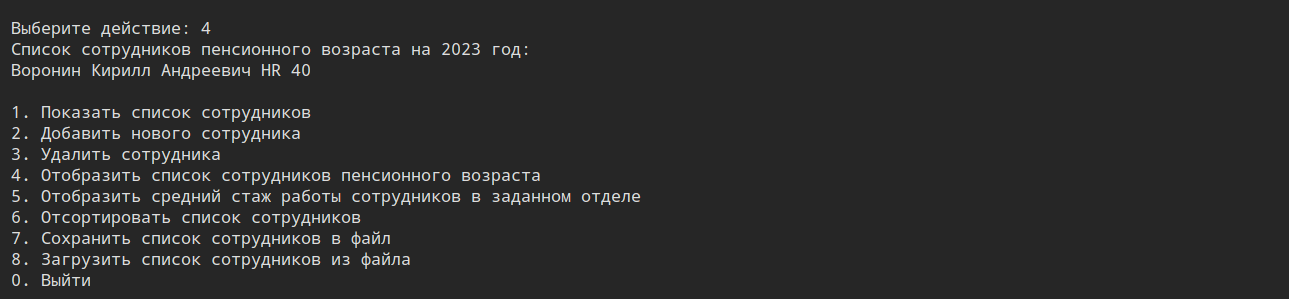
При нажатии на (4) будет выведен список сотрудников по всем отделам которым больше 55 и меньше 120

Вот код:



Программа считывает данные и выводит ФИО, Название отдела и его стаж.

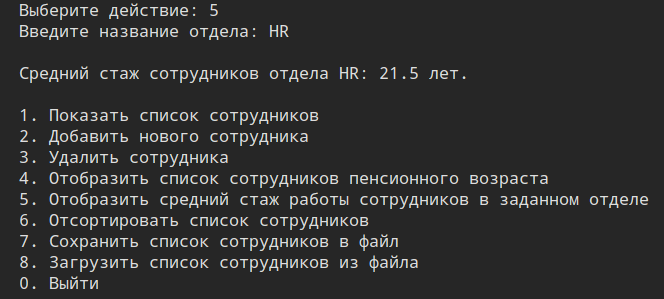
Вот как это выглядит:



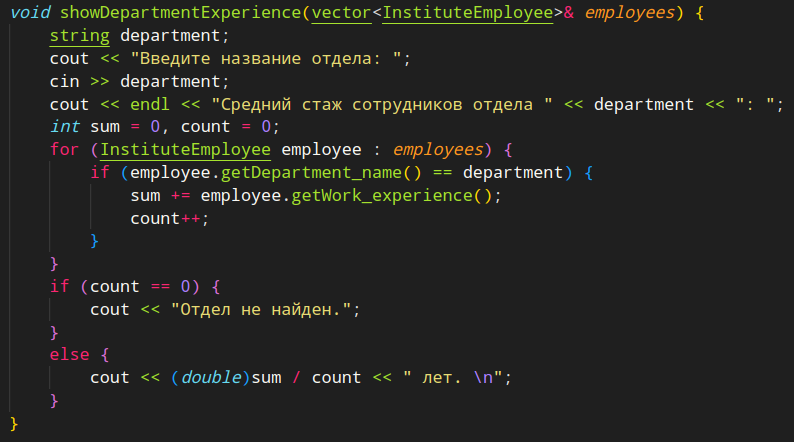
После, программа возвращается в контекстное меню.

5. Отобразить средний стаж работы сотрудников в заданном отделе:

При нажатии на (5) нужно ввести название отдела, вот как это выглядит:



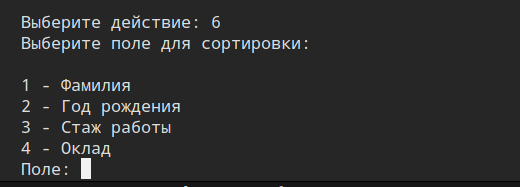
Код выполнения:



После, программа возвращается в контекстное меню.

6. Отсортировать список сотрудников по заданному полю:

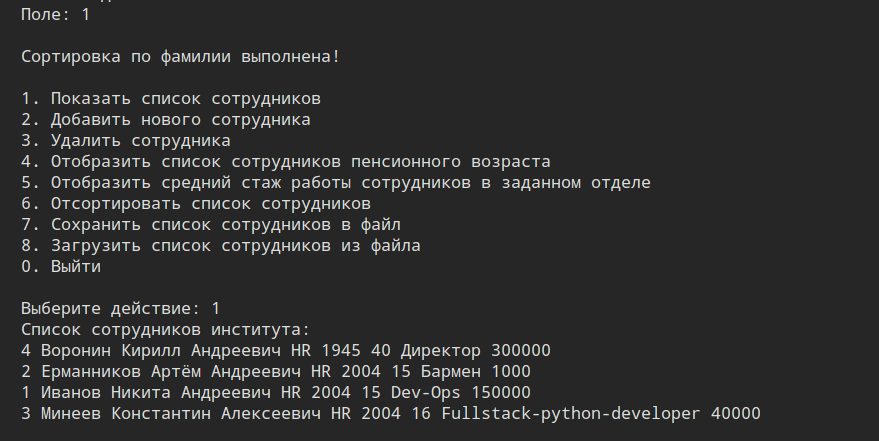
При нажатии (6) нам выходит такое меню где мы можем выбрать по какому полю мы будем сортировать наш список сотрудников:



Мы выбираем из 4 пунктов:

1. По фамилии
2. По году рождения
3. По стажу работы
4. По окладу

Дальше при нажатии из 4 вариантов наш список сотрудников сортируется по заданному полю , пример по фамилии:

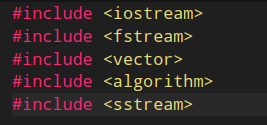


7 и 8 пункт отвечают за загрузку файла (т.е. Загрузка списка сотрудников и сохранения в файле изменений). (7) и (8)

0. Выход из меню.

Из меню можно выйти только при нажатии на (0).

## Список подключенной библиотеки



<iostream>

Реализует основы ввода и вывода языка C++

<fstream>

Реализует инструменты для файлового ввода и вывода.

<algorithm>

Реализует стандартные алгоритмы по типу сортировки.

<vector>

Реализует шаблон класса контейнеров std::vector — динамический массив.

## Объявляю пространство имен

(рис.11 “пространство имен - std”)



# Заключение

Таким образом, через тактическую задачу управляемости разрешил задачу — транслировать понимание задачи программистом в наиболее удобную для дальнейшего использования форму.

Основные принципы структурирования в случае ООП связаны с различными аспектами базового понимания предметной задачи, которое требуется для оптимального управления соответствующей моделью.

В расчетно-графической работе (РГР) закрепил практические навыки самостоятельной работы и отладки любой сложности программного изделия с применением современных навыков и методов программирования; овладение современными методами наглядного представления результатов решения, организации диалога с ЭВМ.

# Список использованной литературы

1. Павловская Т. А. C/C++. Программирование на языке высокого уровня: [учебник для вузов по направлению "Информатика и вычислительная техника"] / Павловская Т. А. – СПб. и др.: Питер, 2010. – 460с.

2. Лаптев В. В. C++. Объектно-ориентированное программирование: задачи и упражнения: [учебное пособие для вузов по направлению "Информатика и вычислительная техника"] / Лаптев В. В., Бокова А. В., Морозов А. В. – СПб.: Питер, 2008. – 287с.

# Приложение 1

#include <iostream>

#include <fstream>

#include <vector>

#include <algorithm>

#include <sstream>

using namespace std;

class InstituteEmployee {

private:

string last\_name, first\_name, middle\_name, department\_name, position;

int birth\_year, work\_experience, salary, tbl\_nomer;

static int count;

public:

InstituteEmployee() {

tbl\_nomer = ++count;

}

InstituteEmployee(string l\_name, string f\_name, string m\_name, string dep\_name, int b\_year, int w\_exp, string pos, int salary) :

last\_name(l\_name), first\_name(f\_name), middle\_name(m\_name), department\_name(dep\_name),

birth\_year(b\_year), work\_experience(w\_exp), position(pos), salary(salary), tbl\_nomer(++count) {}

string getLast\_name() const {

return last\_name;

}

void setLast\_name(string last\_name) {

this->last\_name = last\_name;

}

string getFirst\_name() const {

return first\_name;

}

void setFirst\_name(string first\_name) {

this->first\_name = first\_name;

}

string getMiddle\_name() const {

return middle\_name;

}

void setMiddle\_name(string middle\_name) {

this->middle\_name = middle\_name;

}

string getDepartment\_name() const {

return department\_name;

}

void setDepartment\_name(string department\_name) {

this->department\_name = department\_name;

}

int getBirth\_year() const {

return birth\_year;

}

void setBirth\_year(int birth\_year) {

this->birth\_year = birth\_year;

}

int getWork\_experience() const {

return work\_experience;

}

void setWork\_experience(int work\_experience) {

this->work\_experience = work\_experience;

}

string getPosition() const {

return position;

}

void setPosition(string position) {

this->position = position;

}

int getSalary() const {

return salary;

}

void setSalary(int salary) {

this->salary = salary;

}

int getTbl\_nomer() const {

return tbl\_nomer;

}

};

int InstituteEmployee::count = 0;

void showEmployees(vector<InstituteEmployee>& employees) {

cout << "Список сотрудников института:\n";

for (InstituteEmployee employee : employees) {

cout << employee.getTbl\_nomer() << " " << employee.getLast\_name() << " " << employee.getFirst\_name() << " " << employee.getMiddle\_name();

cout << " " << employee.getDepartment\_name() << " " << employee.getBirth\_year() << " " << employee.getWork\_experience();

cout << " " << employee.getPosition() << " " << employee.getSalary() << endl;

}

}

void showRetiredEmployees(vector<InstituteEmployee>& employees) {

time\_t t = time(NULL);

tm\* timePtr = localtime(&t);

int currentYear = 1900 + timePtr->tm\_year;

cout << "Список сотрудников пенсионного возраста на " << currentYear << " год:\n";

for (InstituteEmployee employee : employees) {

if ((currentYear - employee.getBirth\_year() >= 55) && (currentYear - employee.getBirth\_year() <= 120)) {

cout << employee.getLast\_name() << " " << employee.getFirst\_name() << " " << employee.getMiddle\_name() << " ";

cout << employee.getDepartment\_name() << " " << employee.getWork\_experience() << endl;

}

}

}

void showDepartmentExperience(vector<InstituteEmployee>& employees) {

string department;

cout << "Введите название отдела: ";

cin >> department;

cout << endl << "Средний стаж сотрудников отдела " << department << ": ";

int sum = 0, count = 0;

for (InstituteEmployee employee : employees) {

if (employee.getDepartment\_name() == department) {

sum += employee.getWork\_experience();

count++;

}

}

if (count == 0) {

cout << "Отдел не найден.";

}

else {

cout << (double)sum / count << " лет. \n";

}

}

void addEmployee(vector<InstituteEmployee>& employees) {

string l\_name, f\_name, m\_name, dep\_name, pos;

int b\_year, w\_exp, salary;

cout << "Введите данные сотрудника:\n";

cout << "Фамилия: \n";

cin >> l\_name;

cout << "\nИмя: ";

cin >> f\_name;

cout << "\nОтчество: ";

cin >> m\_name;

cout << "\nНазвание отдела: ";

cin >> dep\_name;

cout << "\nГод рождения: ";

cin >> b\_year;

cout << "\nСтаж работы: ";

cin >> w\_exp;

cout << "\nДолжность: ";

cin >> pos;

cout << "\nОклад: ";

cin >> salary;

InstituteEmployee employee(l\_name, f\_name, m\_name, dep\_name, b\_year, w\_exp, pos, salary);

employees.push\_back(employee);

cout << endl << "Новый сотрудник успешно добавлен!" << endl;

}

void deleteEmployee(vector<InstituteEmployee>& employees) {

int tbl\_nomer;

bool found = false;

cout << "Введите табельный номер удаляемого сотрудника: ";

cin >> tbl\_nomer;

for (vector<InstituteEmployee>::iterator it = employees.begin(); it != employees.end(); ) {

if (it->getTbl\_nomer() == tbl\_nomer) {

it = employees.erase(it);

found = true;

}

else {

++it;

}

}

if (found) {

cout << endl << "Сотрудник успешно удален!" << endl;

}

else {

cout << endl << "Сотрудник не найден." << endl;

}

}

void sortEmployees(vector<InstituteEmployee>& employees) {

cout << "Выберите поле для сортировки:\n";

cout << "\n1 - Фамилия";

cout << "\n2 - Год рождения";

cout << "\n3 - Стаж работы";

cout << "\n4 - Оклад";

cout << "\nПоле: ";

int field;

cin >> field;

switch (field) {

case 1:

sort(employees.begin(), employees.end(), [](InstituteEmployee& e1, InstituteEmployee& e2) {return e1.getLast\_name() < e2.getLast\_name(); });

cout << "\nСортировка по фамилии выполнена!\n";

break;

case 2:

sort(employees.begin(), employees.end(), [](InstituteEmployee& e1, InstituteEmployee& e2) {return e1.getBirth\_year() < e2.getBirth\_year(); });

cout << "\nСортировка по году рождения выполнена!\n";

break;

case 3:

sort(employees.begin(), employees.end(), [](InstituteEmployee& e1, InstituteEmployee& e2) {return e1.getWork\_experience() < e2.getWork\_experience(); });

cout << "\nСортировка по стажу работы выполнена!\n";

break;

case 4:

sort(employees.begin(), employees.end(), [](InstituteEmployee& e1, InstituteEmployee& e2) {return e1.getSalary() < e2.getSalary(); });

cout << "\nСортировка по окладу выполнена!\n";

break;

default:

cout << "\nНекорректный ввод.\n";

break;

}

}

void saveToFile(vector<InstituteEmployee>& employees) {

ofstream fout("employees.txt");

for (InstituteEmployee employee : employees) {

fout << employee.getLast\_name() << ";" << employee.getFirst\_name() << ";" << employee.getMiddle\_name() << ";" << employee.getDepartment\_name() << ";"

<< employee.getBirth\_year() << ";" << employee.getWork\_experience() << ";" << employee.getPosition() << ";" << employee.getSalary() << endl;

}

fout.close();

cout << endl << "Список сотрудников успешно сохранен в файл!\n";

}

void readFromFile(vector<InstituteEmployee>& employees) {

employees.clear();

ifstream fin("employees.txt");

if (!fin.is\_open()) {

cout << "Файл не найден.\n";

return;

}

string line;

while (getline(fin, line)) {

stringstream ss(line);

string l\_name, f\_name, m\_name, dep\_name, pos;

int b\_year, w\_exp, salary;

getline(ss, l\_name, ';');

getline(ss, f\_name, ';');

getline(ss, m\_name, ';');

getline(ss, dep\_name, ';');

ss >> b\_year;

ss.ignore();

ss >> w\_exp;

ss.ignore();

getline(ss, pos, ';');

ss >> salary;

InstituteEmployee employee(l\_name, f\_name, m\_name, dep\_name, b\_year, w\_exp, pos, salary);

employees.push\_back(employee);

}

fin.close();

cout << endl << "Список сотрудников успешно загружен из файла!\n";

}

void showMenu() {

cout << "1. Показать список сотрудников\n";

cout << "2. Добавить нового сотрудника\n";

cout << "3. Удалить сотрудника\n";

cout << "4. Отобразить список сотрудников пенсионного возраста\n";

cout << "5. Отобразить средний стаж работы сотрудников в заданном отделе\n";

cout << "6. Отсортировать список сотрудников\n";

cout << "7. Сохранить список сотрудников в файл\n";

cout << "8. Загрузить список сотрудников из файла\n";

cout << "0. Выйти\n";

cout << "\nВыберите действие: ";

}

int main()

{

setlocale(LC\_ALL, "Russian");

vector<InstituteEmployee> employees;

int choice;

do {

showMenu();

cin >> choice;

switch (choice) {

case 0:

break;

case 1:

showEmployees(employees);

break;

case 2:

addEmployee(employees);

break;

case 3:

deleteEmployee(employees);

break;

case 4:

showRetiredEmployees(employees);

break;

case 5:

showDepartmentExperience(employees);

break;

case 6:

sortEmployees(employees);

break;

case 7:

saveToFile(employees);

break;

case 8:

readFromFile(employees);

break;

default:

cout << "Некорректный ввод.\n";

break;

}

cout << endl;

} while (choice != 0);

return 0;

}

# Приложение 2

Файл:

Иванов;Никита;Андреевич;HR;2004;15;Dev-Ops;150000

Ерманников;Артём;Андреевич;HR;2004;15;Бармен;1000

Минеев;Константин;Алексеевич;HR;2004;16;Fullstack-python-developer;40000

Воронин;Кирилл;Андреевич;HR;1945;40;Директор;300000