

МИНОБРНАУКИ РОССИИ

федеральное государственное бюджетное образовательное учреждение высшего образования

«Санкт-Петербургский государственный технологический институт (технический университет)»

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| УГСН | | 09.00.00 | Информатика и вычислительная техника | | |
| Уровень образования | |  | Высшее образование – бакалавриат | | |
| Форма обучения | |  | Очная | | |
| Факультет | |  | Информационных технологий  и управления | | |
| Кафедра | |  | Систем автоматизированного проектирования и управления | | |
| Учебная дисциплина | |  | Информационные технологии  и программирование | | |
| Курс | I | | | Группа | 4307 |

**Отчёт по контрольной работе № 1**

**Вариант № 5**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Исполнитель: |  |  |  |  |
| обучающийся группы 4307 |  |  |  | Гапоняко Фёдор Дмитриевич |
|  |  | (дата, подпись) |  |  |
|  |  |  |  |  |
| Проверил: |  |  |  | Корниенко Иван Григорьевич |
|  |  |  |  | Федин Алексей Константинович |

# СОДЕРЖАНИЕ

[1 Задание №1 3](#_Toc161130715)

[1.1 Постановка задачи 3](#_Toc161130716)

[1.2 Исходные данные 3](#_Toc161130717)

[1.3 Особые ситуации 3](#_Toc161130718)

[1.4 Математические методы и алгоритмы решения задачи 3](#_Toc161130719)

[1.5 Блок-схема алгоритма решения задачи 4](#_Toc161130720)

[1.6 Форматы представления данных 7](#_Toc161130721)

[1.7 Структура программы 8](#_Toc161130722)

[1.8 Ход выполнения работы 9](#_Toc161130723)

[1.9 Результаты 9](#_Toc161130724)

[1.10 Исходный код полученного программного решения 12](#_Toc161130725)

# 1 Задание №1

## 1.1 Постановка задачи

Необходимо разработать класс для указанной предметной области. Доступ к данным реализовать с помощью методов Set, Get, Show. Предусмотреть необходимые проверки исходных данных. Класс - Сотрудник: фамилия, имя, отчество, должность, год поступления на работу,

зарплата. Создать массив объектов. Реализовать возможность получения:

– списка работников, стаж работы которых на данном предприятии превышает заданное число лет,

– списка работников, зарплата которых больше заданной,

– списка работников, занимающих заданную должность.

1.2 Исходные данные

В качестве исходных данных программа принимает список параметров для создания объекта класса Employee (при ручном вводе): фамилия, имя, отчество – строка, должность – строка, год поступления на работу – число, зарплата – число с плавающей точкой.

## 1.3 Особые ситуации

Из основный ситуаций, которые нужно обработать правильно, есть: обработка пользовательского ввода (как числа, так и строки) и реализация правильного получения/выгрузки файлов. Для обработки первой ситуации были написаны функции InputString, InputInt и InputDouble, которые продолжают свою работу до тех пор, пока пользователь не введет корректные данные. Также для работы программы были написаны функции получения и создания файлов в формате csv, которые так же работают до тех пор, пока корректно не реализуют свою задачу, либо пока пользователь не отменит выгрузку/загрузку файла.

## 1.4 Математические методы и алгоритмы решения задачи

В программе реализуется несколько алгоритмов: сравнение строк, поиск значений выше какого-то числа.

Для реализации сравнения строк требуется список объектов и строка, с которой будет производиться сравнение. Если эти строки совпадают, объект проходит проверку и добавляется в массив найденных объектов.

Поиск значений работает похожим образом: для каждого объекта в списке сравнивают значение свойства объекта и желаемое значение, выше которого должно быть значение свойства. Если это значение выше, объект проходит проверку и добавляется в массив найденных объектов.

## 1.5 Блок-схема алгоритма решения задачи

Для удобства разработки блок-схем были использованы следующие обозначения: E – изначальный массив объектов Employee, U – введенное пользователем значение порога стажа, S - введенное пользователем значение желаемого порога зарплат, P – введенная пользователем желаемая должность, V – массив найденных сотрудников по стажу, T – массив найденных сотрудников по зарплате, J – массив найденных сотрудников по должности, M – динамический размер массивов.



Рисунок 1 – Блок-схема алгоритма поиска сотрудников по стажу



Рисунок 2 – Блок-схема алгоритма поиска сотрудников по зарплатe



Рисунок 3 – Блок-схема алгоритма поиска сотрудников по занимаемой должности

## 1.6 Форматы представления данных

Для описания основной сущности программы был написан класс Employee.

Таблица 1 – Класс Employee

|  |  |  |
| --- | --- | --- |
| Имя | Тип | Описание |
| last\_name | string | Фамилия сотрудника |
| first\_name | string | Имя сотрудника |
| middle\_name | string | Отчество сотрудника |
| job\_title | string | Должность сотрудника |
| enter\_year | int | Год вступления на должность |
| salary | double | Зарплата сотрудника |

Для управления списком файлов был написан класс Employee\_manager.

Таблица 2 – Класс Employee\_manager

|  |  |  |
| --- | --- | --- |
| Имя | Тип | Описание |
| employees | vector<Employee> | Список сотрудников |

В работе программы также были использованы следующие переменные.

Таблица 3 – Основные переменные, используемые в программе

|  |  |  |
| --- | --- | --- |
| Имя | Тип | Описание |
| employees\_to\_export | vector<Employee> | Список файлов для выгрузки |
| suitable\_employees | vector<Employee> | Список подходящих сотрудников |
| work\_experience | int | Желаемый стаж сотрудников |
| salary | double | Желаемая зарплата сотрудников |
| job\_title | string | Желаемая должность сотрудников |

## 1.7 Структура программы

Программа получилась довольно большой, разделена на много модулей и классов. Суммарно в программе больше пяти модулей, самые часто используемые относятся к манипуляциям с файлами, вводом строковых и других значений. Ниже приведен список используемых модулей и функций в них.

Таблица 4 – Функции, составляющие модуль Input

|  |  |
| --- | --- |
| Имя | Описание |
| InputInt | Получения целого числа от пользователя |
| InputString | Получения строки от пользователя |
| InputDouble | Получения числа с плавающей точкой от пользователя |

Таблица 5 – Функции, составляющие модуль Employee\_manager

|  |  |
| --- | --- |
| Имя | Описание |
| add\_employee | Добавление объекта сотрудника в список сотрудников |
| search\_by\_work\_experience | Получение списка сотрудников, стаж которых больше заданного |
| search\_by\_salary | Получение списка сотрудников, зарплата которых больше заданной |
| search\_by\_job\_title | Получение списка сотрудников, должность которых совпадает с заданной |

## 1.8 Ход выполнения работы

В ходе контрольной работы было создано решение в интегрированной среде разработки Microsoft Visual Studio C++ 2022. В ней был создан проект.

Также выяснилось, что для более удобных манипуляций над массивом объектов следует создать отдельный класс, после чего был создан класс Employee\_manager.

## 1.9 Результаты

В результате работы выполнения программы, пользователь может добавлять новых сотрудников, получать сотрудников по стажу, зарплате и должности. Также имеется возможно выгружать список в файл, и загружать исходные данные из файла.



Рисунок 4 – Скриншот результата добавления нового объекта в список

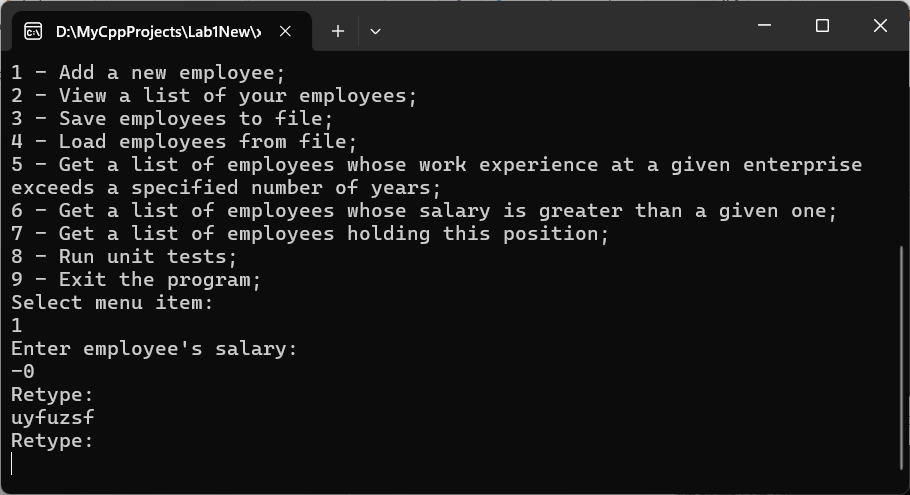


Рисунок 5 – Скриншот обработки ошибки добавления

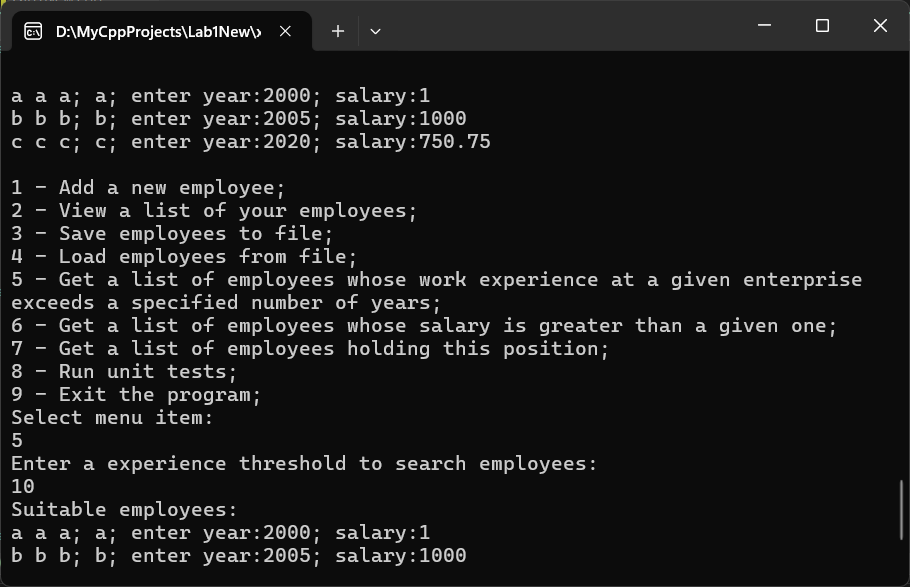


Рисунок 6 – Скриншот результата работы поиска по стажу

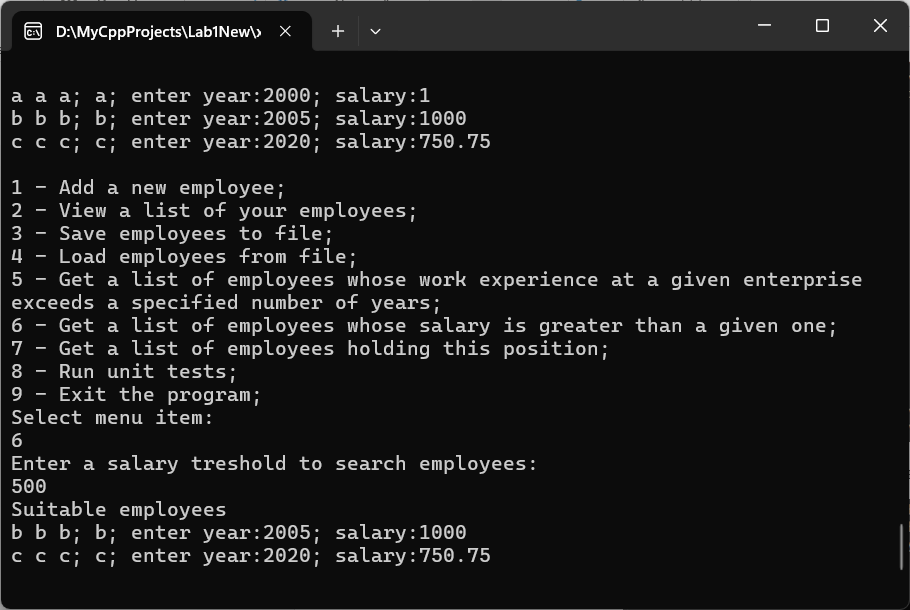


Рисунок 7 – Скриншот результата работы поиска по зарплате

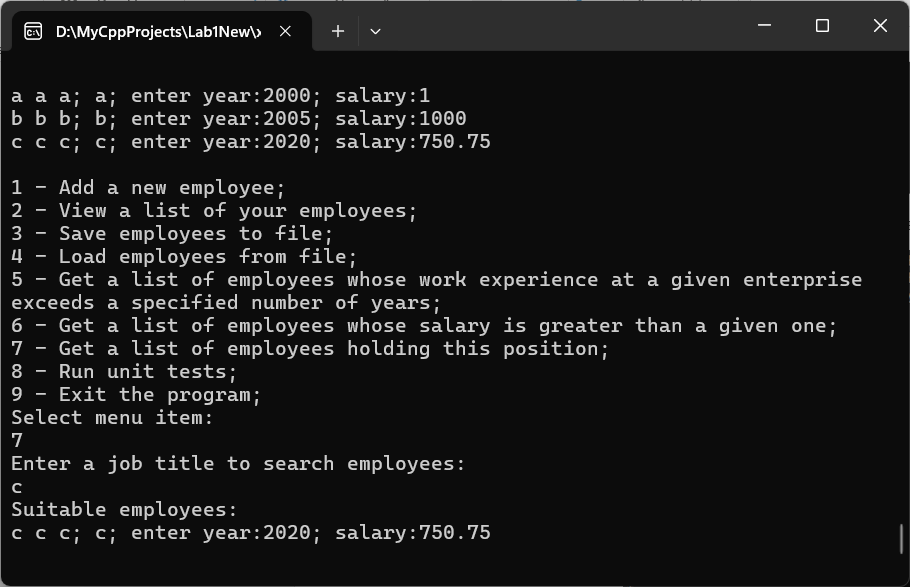


Рисунок 8 – Скриншот результата работы поиска по должности

## 1.10 Исходный код полученного программного решения

// Main Programm

#include "Employee\_manager.h"

#include "ShowMenu.h"

#include "Files.h"

#include "Input.h"

#include "Tasks.h"

#include "Tests.h"

#include <iostream>

#include <vector>

enum inems { ADD\_EMPLOYEE = 1, SHOW\_EMPLOYEES, SAVE\_DATA, LOAD\_DATA, SEARCH\_BY\_WORK\_EXPERIENCE, SEARCH\_BY\_SALARY, SEARCH\_BY\_JOB\_TITLE, RUN\_TESTS, EXIT };

using namespace std;

int main()

{

ShowGreeting();

int menu\_item;

bool repeat = true;

string file;

Employee\_manager employees;

do {

ShowMenu();

menu\_item = InputInt("Select menu item: ", 0, 10);

switch (menu\_item)

{

case(ADD\_EMPLOYEE): { add\_employee(employees); break; }

case(SHOW\_EMPLOYEES): { employees.show\_employees(); break; }

case(SAVE\_DATA): { save\_data(employees); break; }

case(LOAD\_DATA): { load\_data(employees); break; }

case(SEARCH\_BY\_WORK\_EXPERIENCE): { search\_by\_work\_experience(employees); break; }

case(SEARCH\_BY\_SALARY): { search\_by\_salary(employees); break; }

case(SEARCH\_BY\_JOB\_TITLE): { search\_by\_job\_title(employees); break; }

case(RUN\_TESTS): { run\_tests(); break; }

case(EXIT): { repeat = false; }

}

} while (repeat);

}

// Employee class

#pragma once

#include <iostream>

#include "Input.h"

using namespace std;

class Employee {

private:

string last\_name;

string first\_name;

string middle\_name;

string job\_title;

int enter\_year;

double salary;

public:

Employee(string last\_name, string first\_name, string middle\_name, string job\_title, int enter\_year, double salary) :

last\_name(last\_name), first\_name(first\_name), middle\_name(middle\_name),job\_title(job\_title), enter\_year(enter\_year), salary(salary) {}

string get\_last\_name() { return last\_name; }

string get\_first\_name() { return first\_name; }

string get\_middle\_name() { return middle\_name; }

string get\_job\_title() { return job\_title; }

int get\_work\_experience() { return 2024 - enter\_year; }

int get\_enter\_year() { return enter\_year; }

double get\_salary() { return salary; }

void show\_employee() { cout << last\_name << ' ' << first\_name << ' ' << middle\_name << "; " << job\_title << "; enter year:" << enter\_year << "; salary:" << salary << endl; }

};

// Employee\_manager class

#pragma once

#include <vector>

#include "Employee.h"

class Employee\_manager

{

private:

vector<Employee> employees;

public:

void set\_employees(vector<Employee> new\_employees) {

if(new\_employees.empty()){ cout << "There is no employees" << endl; return; }

employees = new\_employees;

}

void show\_employees() {

cout << endl;

if (employees.empty()) { cout << "You haven't added any employees yet" << endl; }

for (Employee employee : employees) { employee.show\_employee(); }

cout << endl;

}

void show\_suitable\_employees(vector<Employee> suitable\_employees) {

for (Employee employee : suitable\_employees) { employee.show\_employee(); }

cout << endl;

}

void add\_employee(Employee employee) { employees.push\_back(employee); }

vector<Employee> search\_by\_work\_experience(int work\_experience) {

vector<Employee> suitable\_employees;

for (Employee employee : employees) {

if (employee.get\_work\_experience() > work\_experience) { suitable\_employees.push\_back(employee); }

}

return suitable\_employees;

}

vector<Employee> search\_by\_salary(double salary) {

vector<Employee> suitable\_employees;

for (Employee employee : employees) {

if (employee.get\_salary() > salary) { suitable\_employees.push\_back(employee); }

}

return suitable\_employees;

}

vector<Employee> search\_by\_job\_title(string job\_title) {

vector<Employee> suitable\_employees;

for (Employee employee : employees) {

if (employee.get\_job\_title() == job\_title) { suitable\_employees.push\_back(employee); }

}

return suitable\_employees;

}

vector<Employee>get\_employees() {

return employees;

}

};