Министерство науки и высшего образования Российской Федерации
Федеральное государственное бюджетное образовательное учреждение высшего образования
«Московский государственный технический университет имени Н. Э. Баумана
(национальный исследовательский университет)»
(МГТУ им. Н. Э. Баумана)

ФАКУЛЬТЕТ «Информатика и системы управления» (ИУ)
КАФЕДРА «Информационная безопасность» (ИУ8)

Лабораторная работа № 5

ПО КУРСУ

«Алгоритмические языки»

на тему «Изучение использования объектов своих классов в упорядоченных и неупорядоченных контейнерах библиотеки STL (set и map, unordered_set и unordered_map)»

Студент ИУ8-23 (Группа) В. С. Ажгирей (И. О. Фамилия)

Преподаватель: М. В. Малахов (И.О. Фамилия)

2024

Введение

Цели и задачи работы

Для класса, разработанного в ЛР4, обеспечить возможность добавления объектов в контейнер set (сортировка как указано в задании на ЛР4) и в контейнер unordered_set. Исходные данные как в ЛР4 читать из файла, вывести на печать для контроля объекты контейнеров.

Условия

Для класса, разработанного в ЛР4, обеспечить возможность добавления объектов в контейнер set (сортировка как указано в задании на ЛР4) и в контейнер unordered_set. Исходные данные как в ЛР4 читать из файла, вывести на печать для контроля объекты контейнеров.

Основная часть

Исходный текст программы:

Файл заголовка sources.hpp:

```
#pragma once
#include <fstream>
#include <iostream>
#include <map>
#include <set>
#include <string>
#include <unordered_map>
#include <unordered_set>
#include <vector>


class Employee
{
private:
    std::string fullname;
    std::string date_employment;
    std::string post;
    size_t salary;

public:
    Employee();

    Employee(std::string, std::string, std::string, size_t);

    Employee(const Employee&);

    Employee(Employee&&) noexcept;

    Employee& operator = (const Employee&);

    Employee& operator = (Employee&&) noexcept;

    bool operator == (const Employee&) const;

    bool operator<(const Employee&) const;

    bool operator()(const Employee&, const Employee&) const;

    std::string getName() const;

    friend std::ostream& operator << (std::ostream&, const Employee&);

    friend std::istream& operator >> (std::istream&, Employee&);
};

std::vector<Employee> readData(std::istream&);

bool sortingByName(const Employee&, const Employee&);
```

Файл описания sources.cpp:

```
#include "sources.hpp"

Employee::Employee() : fullname(""), date_employment(""), post(""), salary(0) {}
```

```cpp
Employee::Employee(std::string fullname, std::string date_employment, std::string
post, size_t salary) : fullname(fullname), date_employment(date_employment),
post(post), salary(salary) {}

Employee::Employee(const Employee& other) : fullname(other.fullname),
date_employment(other.date_employment), post(other.post), salary(other.salary) {}

Employee::Employee(Employee&& other) noexcept : fullname(other.fullname),
date_employment(other.date_employment), post(other.post), salary(other.salary)
{
	other.fullname = "";
	other.date_employment = "";
	other.post = "";
	other.salary = 0;
}

Employee& Employee::operator=(const Employee& other)
{
	if (this != &other) {
		fullname = other.fullname;
		date_employment = other.date_employment;
		post = other.post;
		salary = other.salary;
	}

	return *this;
}

Employee& Employee::operator=(Employee&& other) noexcept
{
	if (this != &other) {
		fullname = other.fullname;
		date_employment = other.date_employment;
		post = other.post;
		salary = other.salary;

		other.fullname = "";
		other.date_employment = "";
		other.post = "";
		other.salary = 0;
	}

	return *this;
}

bool Employee::operator==(const Employee& other) const
{
	return fullname == other.fullname;
}

bool Employee::operator<(const Employee& other) const
{
	return fullname < other.fullname;
}

bool Employee::operator()(const Employee& firstEmployee, const Employee&
secondEmployee) const
{
	return firstEmployee < secondEmployee;
}

std::string Employee::getName() const
{
	return fullname;
```

```
}

std::ostream& operator<<(std::ostream& output_stream, const Employee& employeer)
{
	output_stream << "Fullname: " << employeer.fullname << std::endl;
	output_stream << "Date of employment: " << employeer.date_employment <<
std::endl;
	output_stream << "Post: " << employeer.post << std::endl;
	output_stream << "Salary: " << employeer.salary << std::endl;
	output_stream << "########################################" << std::endl;

	return output_stream;
}

std::istream& operator>>(std::istream& input_stream, Employee& employeer)
{
	input_stream >> employeer.fullname;
	input_stream >> employeer.date_employment;
	input_stream >> employeer.post;
	input_stream >> employeer.salary;

	return input_stream;
}

std::vector<Employee> readData(std::istream& input_stream)
{
	size_t n;
	input_stream >> n;
	std::vector<Employee> deque_emloyee(n);
	for (size_t i = 0; i < n; ++i)
	{
		input_stream >> deque_emloyee[i];
	}
	return deque_emloyee;
}

bool sortingByName(const Employee& firstEmployee, const Employee& secondEmployee)
{
	return firstEmployee.getName() < secondEmployee.getName();
}

namespace std {

	template<>
	struct hash<Employee> {
		size_t operator()(const Employee& employee) const {
			return hash<string>()(employee.getName());
		}

	};
}
```

Исполняемый файл main.cpp:

```
#include "sources.hpp"

int main()
{
	std::ifstream inputFile("input.txt");
	std::ofstream outputFile("output.txt");
```

```
	std::vector<Employee> vector_employee  = readData(inputFile);

	std::set<Employee> set_employee;

	std::copy(vector_employee.begin(), vector_employee.end(),
std::inserter(set_employee, set_employee.begin()));

	for (const Employee employee : set_employee) {
		outputFile << employee <<std::endl;
	}

	outputFile.close();

	return 0;
}
```

Снимки выполнения работы программы

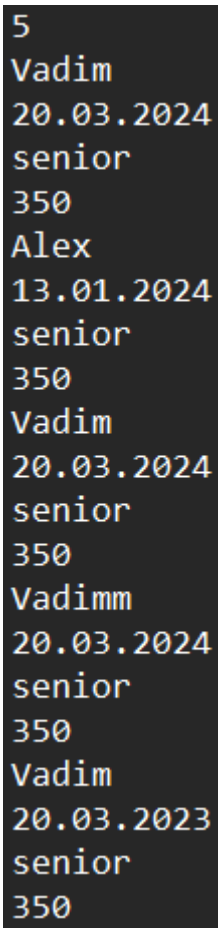

Рисунок 1 – Входные данные

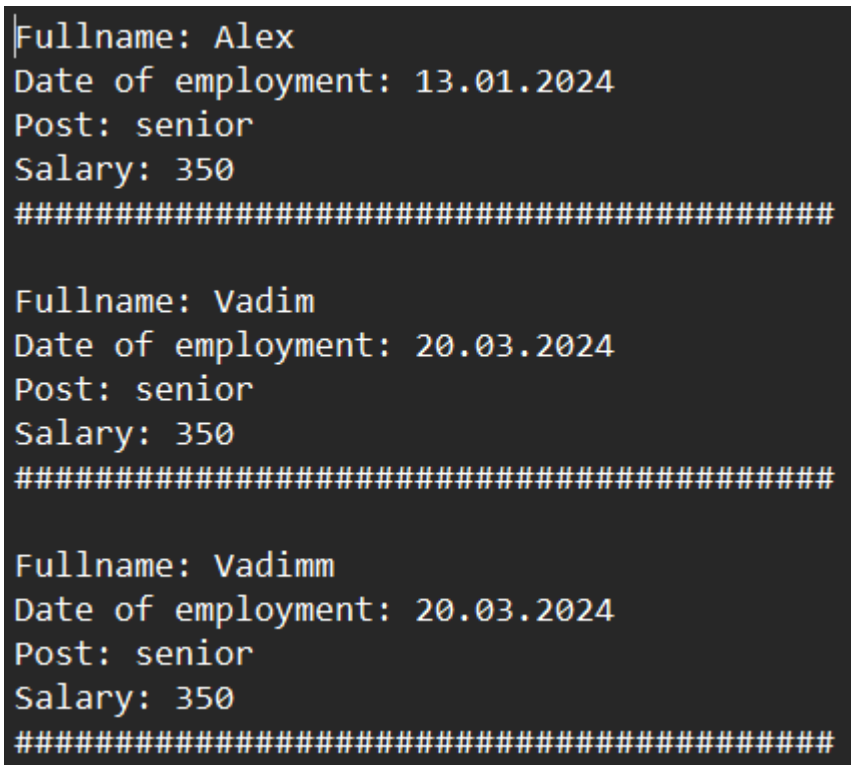

Рисунок 2 – Выходные данные

Заключение

Задачи лабораторной работы были решены, результаты проверены. Обеспечена возможность добавления объектов в контейнер set (сортировка по ФИО) и в контейнер unordered_set.