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 № 4

ПО КУРСУ

«Алгоритмические языки»

на тему «Использование своих классов в STL 1»

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

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

2024

Введение

Цели и задачи работы

В приложении организовать контейнер объектов своего класса (использовать шаблоны std::list, std::vector или std::deque в зависимости от варианта, элементы контейнера - объекты класса. Класс должен иметь необходимые конструкторы, конструктор копирования и перемещения при необходимости (обосновать отсутствие или наличие необходимости), перегруженные операции присваивания с копированием и перемещением при необходимости (обосновать отсутствие или наличие необходимости), перегруженную операцию вставки в поток <<.

Обеспечить копирование одного контейнера в другой с помощью алгоритма std::copy. А также сортировку объектов в исходном контейнере.

Условия для 1 варианта

Параметры приложений: Объект- сотрудник (поля: ФИО, дата приема на работу, должность, базовый оклад). Сортировка по ФИО. Исходный контейнер vector, копируем в deque

Основная часть

Исходный текст программы:

Файл заголовка sources.hpp:

```
#pragma once
#include <algorithm>
#include <deque>
#include <fstream>
#include <iostream>
#include <string>
#include <vector>


class Employee
{
    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&&);

    Employee& operator = (const Employee&);

    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) {}

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) : 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;
}

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();
}
```

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);

	for (const Employee employee : vector_employee)
	{
		std::cout << employee;
	}
	std::cout << std::endl << std::endl;

	std::deque<Employee> deque_employee(vector_employee.size());

	std::copy(vector_employee.cbegin(), vector_employee.cend(),
deque_employee.begin());

	for (const Employee employee : deque_employee)
	{
		std::cout << employee;
	}
	std::cout << std::endl << std::endl;

	//std::sort(deque_employee.begin(), deque_employee.end(), [](const Employee&
firstEmployee, const Employee& secondEmployee) -> bool {return
firstEmployee.getName() < secondEmployee.getName(); });

	std::sort(deque_employee.begin(), deque_employee.end());


	for (const Employee employee : deque_employee)
	{
		std::cout << employee;
	}

	for (const Employee employee : deque_employee)
	{
		outputFile << employee;
	}

	outputFile.close();

	return 0;
}
```

Снимки выполнения работы программы

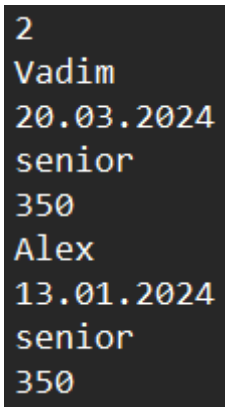

Рисунок 1 – Входные данные

Рисунок 2 – Выходные данные

Заключение

Задачи лабораторной работы были решены, результаты проверены. Обеспечено копирование одного контейнера в другой с помощью алгоритма std::copy. А также сортировку объектов в исходном контейнере.