



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, искусственный интеллект и системы управления» (ИУ)
КАФЕДРА «Информационная безопасность» (ИУ8)

Лабораторная работа №4
ПО КУРСУ
«Использование объектов своих классов в
последовательных контейнерах библиотеки STL»

Студент

ИУ8-21
(Группа)

Г. А. Карев
(И. О. Фамилия)

Преподаватель:

В. В. Соборова
(И.О. Фамилия)

Цель работы:

Цель работы состоит в овладении навыком использования объектов классов в последовательных контейнерах. Написать рабочую программу для выполнения задачи.

Вариант 12

Задача:

В приложении организовать контейнер объектов своего класса (использовать шаблоны `std::list`, `std::vector` или `std::deque` в зависимости от варианта, элементы контейнера - объекты класса, не указатели!!!!). Варианты заданий заданы в ячейках таблицы 1.

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, для шаблона `list` при сортировке использовать метод `list::sort` без параметров, для шаблона `vector` или `deque` при сортировке использовать алгоритм `std::sort` с двумя параметрами: итератор на начало и итератор на конец контейнера.

Исходные данные прочитать из текстового файла `input.txt`. Вывести в выходной файл `output.txt` исходный контейнер, контейнер после сортировки, использовать при этом перегруженную операцию вставки в поток, также вывести в выходной файл контейнер, в который скопирован исходный контейнер.

Параметры приложений		Исходный контейнер <code>vector</code> , копируем в <code>deque</code>	Исходный контейнер <code>list</code> , копируем в <code>vector</code>
Объект- сотрудник (поля: ФИО, дата приема на работу, должность, базовый оклад)	Сортировка по ФИО	1	34
	Сортировка по окладу	9	4

Объект- банковский вклад (поля: название, сумма вклада, тип валюты, ставка в % годовых)	Сортировка по сумме вклада	5	2
	Сортировка по названию	7	31
Объект- студент (поля: ФИО, группа, номер зачетной книжки, массив 4-х оценок за сессию)	Сортировка по ФИО	3	10
	Сортировка по среднему баллу	11	12

Код:

Header.h (заголовочный файл)

```
#pragma once
#include <string>
using namespace std;

class Student {

    string name;
    string group;
    string number;
    int grades[4];

public:
    Student();
    Student(const string& name, const string& grp, const string& num, const int
g[4]);
    Student(const Student& copy);
    Student(Student&& cmove) noexcept;
    Student& operator=(const Student& givecopy);
    Student& operator=(Student&& givemove) noexcept;
    double getGrade() const;
    friend ostream& operator<<(ostream& outS, const Student& student);
    bool operator<(const Student& other) const;
};
```

Header.cpp (файл реализации)

```
#include "Header.h"
#include <numeric>
#include <iostream>

Student::Student() : name(""), group(""), number(""), grades{ 0, 0, 0, 0 } {}

Student::Student(const string& name, const string& grp, const string& num, const int
g[4]) : name(name), group(grp), number(num) {
    for (int i = 0; i < 4; ++i) { grades[i] = g[i]; }
}

Student::Student(const Student& copy) : name(copy.name), group(copy.group),
number(copy.number) {
    for (int i = 0; i < 4; ++i) {
        grades[i] = copy.grades[i];
    }
}

Student::Student(Student&& cmove) noexcept : name(move(cmove.name)),
group(move(cmove.group)), number(move(cmove.number)) {
    for (int i = 0; i < 4; ++i) {
        grades[i] = cmove.grades[i];
    }
}

Student& Student::operator=(const Student& givecopy) {
    if (this != &givecopy) {
        name = givecopy.name;
        group = givecopy.group;
        number = givecopy.number;
    }
}
```

```

        for (int i = 0; i < 4; ++i) { grades[i] = givecopy.grades[i]; }
    }
    return *this;
}

Student& Student::operator=(Student&& givemove) noexcept {
    if (this != &givemove) {
        name = move(givemove.name);
        group = move(givemove.group);
        number = move(givemove.number);
        for (int i = 0; i < 4; ++i) { grades[i] = givemove.grades[i]; }
    }
    return *this;
}

double Student::getGrade() const {
    return accumulate(begin(grades), end(grades), 0.0) / 4.0;
}

ostream& operator<<(ostream& outS, const Student& student) {
    outS << "ФИО: " << student.name << ", группа: " << student.group << ", номер  
зачетки: " << student.number << ", оценки: ";
    for (int grade : student.grades) { outS << grade << " "; }
    outS << ", ср. балл: " << student.getGrade();
    return outS;
}

bool Student::operator<(const Student& other) const {
    return getGrade() < other.getGrade();
}

```

zhc.cpp (основной файл)

```

#include <iostream>
#include <fstream>
#include <string>
#include <list>
#include <vector>
#include <algorithm>
#include <numeric>
#include "Header.h"
using namespace std;

int main() {
    list<Student> studentsList;
    ifstream inF("input.txt");
    ofstream outF("output.txt");

    if (inF.is_open()) {
        string name, group, recordNum;
        int grades[4];

        while (inF >> name >> group >> recordNum >> grades[0] >> grades[1] >>
grades[2] >> grades[3]) {
            studentsList.emplace_back(name, group, recordNum, grades);
        }
        inF.close();
    }
    else {
        cerr << "шкибиди файл" << endl;
        return 1;
    }
}

```

```

    }

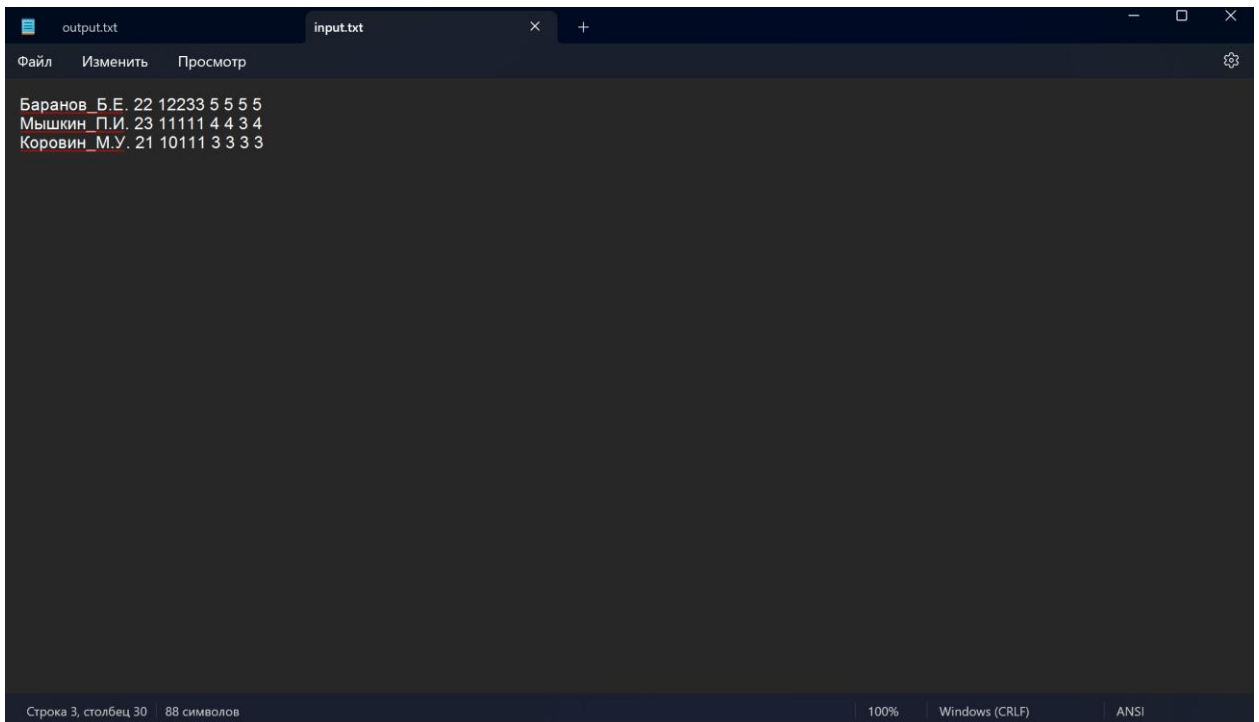
    outF << "Список студентов (list):" << endl;
    for (const auto& student : studentsList) {
        outF << student << endl;
    }
    outF << endl;

    vector<Student> studentsVector;
    copy(studentsList.begin(), studentsList.end(), back_inserter(studentsVector));
    outF << "Копированный список студентов (vector):" << endl;
    for (const auto& student : studentsVector) {
        outF << student << endl;
    }

    studentsList.sort();
    outF << "\nСписок после сортировки (list):" << endl;
    for (const auto& student : studentsList) {
        outF << student << endl;
    }
    outF << endl;
    outF.close();
    return 0;
}

```

Файл для ввода:



The screenshot shows a text editor window with two tabs: 'output.txt' and 'input.txt'. The 'input.txt' tab is active, displaying the following text:

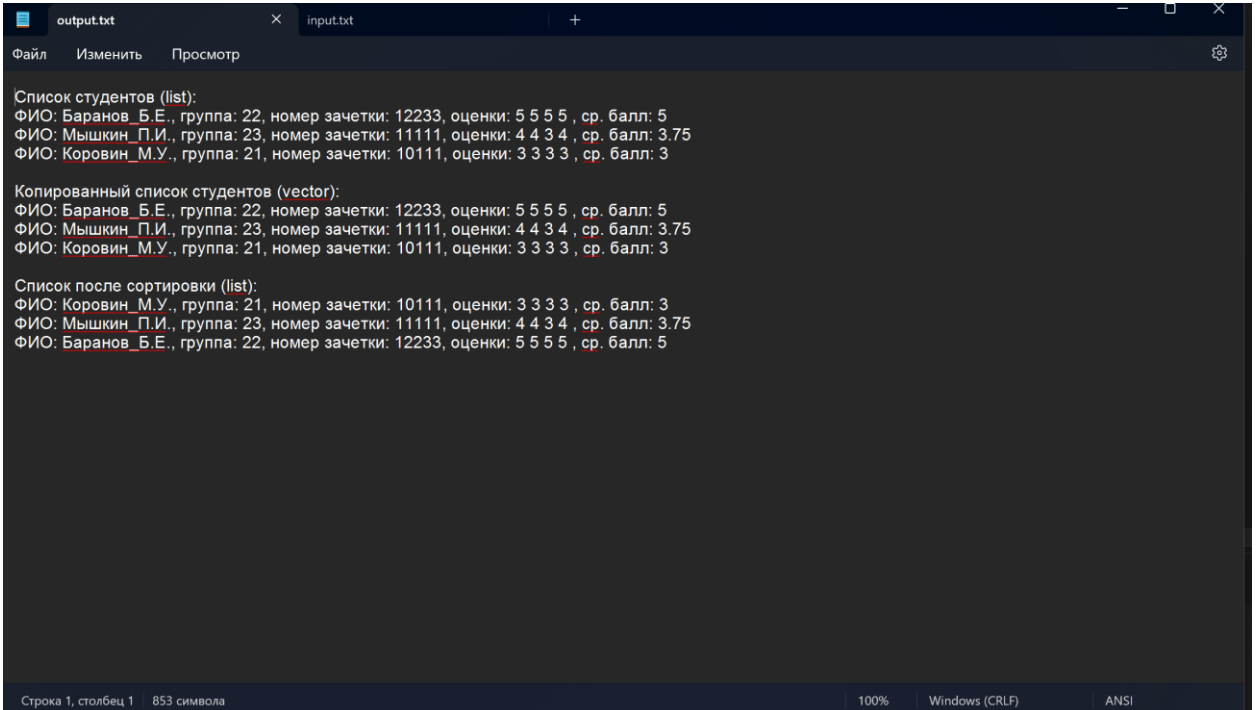
```

Баранов_Б.Е. 22 12233 5 5 5 5
Мышкин_П.И. 23 11111 4 4 3 4
Коровин_М.У. 21 10111 3 3 3 3

```

The status bar at the bottom indicates 'Строка 3, столбец 30 | 88 символов', '100%', 'Windows (CRLF)', and 'ANSI'.

Результаты вывода программы:

A screenshot of a code editor window with two tabs: 'output.txt' and 'input.txt'. The 'output.txt' tab is active and displays the following text:

```
Список студентов (list):  
ФИО: Баранов_Б.Е., группа: 22, номер зачетки: 12233, оценки: 5 5 5 5 , ср. балл: 5  
ФИО: Мышкин_П.И., группа: 23, номер зачетки: 11111, оценки: 4 4 3 4 , ср. балл: 3.75  
ФИО: Коровин_М.У., группа: 21, номер зачетки: 10111, оценки: 3 3 3 3 , ср. балл: 3  
  
Копированный список студентов (vector):  
ФИО: Баранов_Б.Е., группа: 22, номер зачетки: 12233, оценки: 5 5 5 5 , ср. балл: 5  
ФИО: Мышкин_П.И., группа: 23, номер зачетки: 11111, оценки: 4 4 3 4 , ср. балл: 3.75  
ФИО: Коровин_М.У., группа: 21, номер зачетки: 10111, оценки: 3 3 3 3 , ср. балл: 3  
  
Список после сортировки (list):  
ФИО: Коровин_М.У., группа: 21, номер зачетки: 10111, оценки: 3 3 3 3 , ср. балл: 3  
ФИО: Мышкин_П.И., группа: 23, номер зачетки: 11111, оценки: 4 4 3 4 , ср. балл: 3.75  
ФИО: Баранов_Б.Е., группа: 22, номер зачетки: 12233, оценки: 5 5 5 5 , ср. балл: 5
```

The editor interface includes a menu bar with 'Файл', 'Изменить', and 'Просмотр'. The status bar at the bottom shows 'Строка 1, столбец 1', '853 символа', '100%', 'Windows (CRLF)', and 'ANSI'.

Вывод:

Научился использовать объекты класса в различных контейнерах.
Написал работоспособную программу.