МИНИСТЕРСТВО ОБРАЗОВАНИЯ И НАУКИ РОССИЙСКОЙ ФЕДЕРАЦИИ

ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ АВТОНОМНОЕ

ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ ВЫСШЕГО ОБРАЗОВАНИЯ

**«ПЕРМСКИЙ НАЦИОНАЛЬНЫЙ ИССЛЕДОВАТЕЛЬСКИЙ ПОЛИТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ»**

ЭЛЕКТРОТЕХНИЧЕСКИЙ ФАКУЛЬТЕТ

КАФЕДРА «ИНФОРМАЦИОННЫХ ТЕХНОЛОГИЙ И АВТОМАТИЗИРОВАННЫХ СИСТЕМ»

**ОТЧЁТ**

**«ЛАБОРАТОРНАЯ №8: ОБРАБОТКА ФАЙЛА»**

Дисциплина: «Программирование»

Выполнил:

Студент группы ИВТ-21-2б

Безух Владимир Сергеевич

Проверил:

Доцент кафедры ИТАС

Полякова Ольга Андреевна

Пермь, 2022

Содержание

[1. Постановка задачи 3](#_Toc97822557)

[2. Анализ задачи 4](#_Toc97822558)

[3. Описание переменных 5](#_Toc97822559)

[4. Исходный код 6](#_Toc97822560)

[5. Анализ результатов 9](#_Toc97822561)

# Постановка задачи

Дана структура «Студент»:

— фамилия, имя, отчество;

— дата рождения;

— домашний адрес;

— рейтинг.

Нужно удалить элементы, у которых даты рождения совпадают. Затем добавить элемент перед элементом с заданной фамилией.

# 

# Анализ задачи

Для удаления одинаковых элементов по дате рождения нужно:

— отсортировать элементы по полю даты рождения;

— в отсортированном виде найти элементы с датой рождения в единственном экземпляре.

Добавить новый элемент по ключу — тривиальная задача.

# 

# Описание переменных

**std::vector<Student> students;** — хранение записей о студентах.

**std::ifstream f\_input("FileInput.txt");** — для использования файла с входными данными.

**std::ofstream f\_output("FileOutput.txt");** — для использования файла с выходными данными.

**bool flag\_new\_group = true;** — проверка начала новой группы элементов по дате рождения.

**std::vector<Student> result\_students;** — выходные данные.

# Исходный код

#include <vector>

#include <string>

#include <fstream>

#include <iostream>

#include <iterator>

#include <algorithm>

struct FullName {

std::string forename;

std::string surname;

std::string patronym;

bool operator==(const FullName& right) const

{

return (forename == right.forename) &&

(surname == right.surname) &&

(patronym == right.patronym);

}

};

struct Date {

unsigned int day;

unsigned int month;

unsigned int year;

bool operator==(const Date& right) const

{

return (day == right.day) &&

(month == right.month) &&

(year == right.year);

}

bool operator!=(const Date& right) const {

return !(\*this == right);

}

bool operator<(const Date& right) const {

if (year < right.year) return true;

else if (year == right.year)

{

if (month < right.month) return true;

else if (month == right.month) return day < right.day;

else return false;

}

else return false;

}

};

struct Address {

std::string city;

std::string street\_name;

std::string house\_number;

bool operator==(const Address& right) const

{

return (city == right.city) &&

(street\_name == right.street\_name) &&

(house\_number == right.house\_number);

}

};

struct Student {

FullName personal\_name;

Date date\_of\_birth;

Address personal\_address;

int personal\_rating;

bool operator==(const Student& right) const

{

return (personal\_name == right.personal\_name) &&

(date\_of\_birth == right.date\_of\_birth) &&

(personal\_address == right.personal\_address) &&

(personal\_rating == right.personal\_rating);

}

friend std::istream& operator>>(std::istream& input, Student& student)

{

input >> student.personal\_name.forename;

input >> student.personal\_name.surname;

input >> student.personal\_name.patronym;

input >> student.date\_of\_birth.day;

input >> student.date\_of\_birth.month;

input >> student.date\_of\_birth.year;

input >> student.personal\_address.city;

input >> student.personal\_address.street\_name;

input >> student.personal\_address.house\_number;

input >> student.personal\_rating;

return input;

}

friend std::ostream& operator<<(std::ostream& output, const Student& student)

{

output << student.personal\_name.forename << ' ';

output << student.personal\_name.surname << ' ';

output << student.personal\_name.patronym << ' ';

output << student.date\_of\_birth.day << ' ';

output << student.date\_of\_birth.month << ' ';

output << student.date\_of\_birth.year << ' ';

output << student.personal\_address.city << ' ';

output << student.personal\_address.street\_name << ' ';

output << student.personal\_address.house\_number << ' ';

output << student.personal\_rating << '\n';

return output;

}

};

int main()

{

std::vector<Student> students;

std::ifstream f\_input("FileInput.txt");

std::ofstream f\_output("FileOutput.txt");

while (!f\_input.eof()) {

Student student;

f\_input >> student;

students.emplace\_back(student);

}

sort(students.begin(), students.end(),

[](const Student& left, const Student& right)

{ return left.date\_of\_birth < right.date\_of\_birth; });

bool flag\_new\_group = true;

std::vector<Student> result\_students;

for (auto prev = students.begin(); prev != students.end(); ++prev) {

auto curr = prev; ++curr;

if (curr == students.end()) {

if (flag\_new\_group)

result\_students.emplace\_back(\*prev);

break;

}

if (prev->date\_of\_birth != curr->date\_of\_birth) {

if (flag\_new\_group)

result\_students.emplace\_back(\*prev);

flag\_new\_group = true;

} else

flag\_new\_group = false;

}

std::string surname; std::cin >> surname;

Student student; std::cin >> student;

auto iterator = result\_students.begin();

for (; iterator->personal\_name.surname != surname; ++iterator);

result\_students.emplace(iterator, student);

for (auto& result\_student : result\_students)

f\_output << result\_student;

f\_input.close();

f\_output.close();

}

# Анализ результатов

Исходный файл (рис. 1).

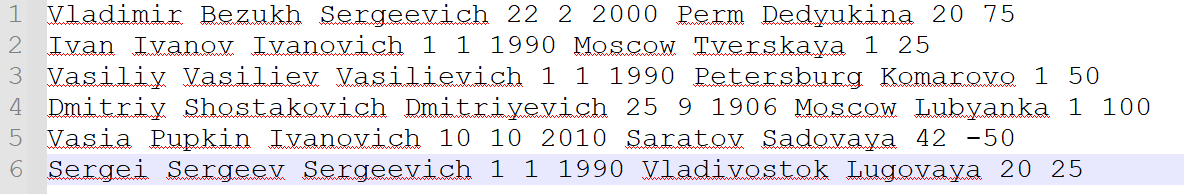


Рисунок 1 — Студенты

Вставка элемента по ключу (рис. 2).

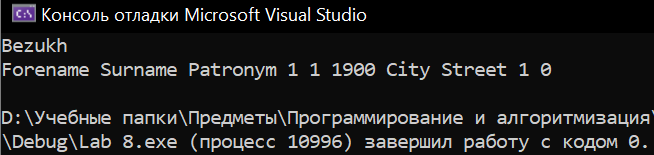


Рисунок 2 — Вставка нового студента

Результирующий файл (рис. 3).

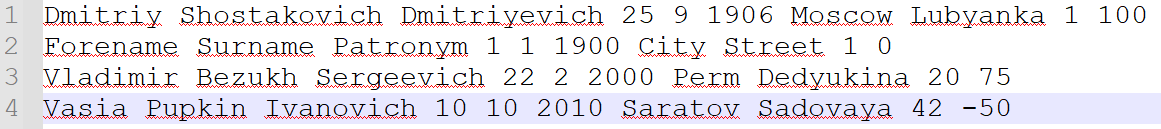


Рисунок 3 — Результат после удаления трёх студентов и вставки нового