Министерство образования и науки Российской Федерации

Федеральное государственное бюджетное образовательное учреждение высшего образования

**«Пермский национальный исследовательский политехнический университет»**

Кафедра **«**Информационные технологии и автоматизированные системы**»**

**ОТЧЕТ**

**ПО ЛАБОРАТОРНОЙ РАБОТЕ №8**

Дисциплина: «Основы алгоритмизации и программирования»

Семестр 2

Тема: Блоковый ввод-вывод.

Вариант 25.

Выполнил работу

Студент группы РИС-22-1Б

Поважный В. Е.

Проверил

Доцент кафедры ИТАС

Полякова О.А.

г. Пермь-2023

**Введение**

Для выполнения лабораторной работы требуется сделать отчет программы. Создание программ – отличный способ практики программирования.

**Постановка задачи**

Сформировать двоичный файл из элементов, заданной в варианте структуры, распечатать его содержимое, выполнить удаление и добавление элементов в соответствии со своим вариантом, используя для поиска удаляемых или добавляемых элементов функцию. Формирование, печать, добавление и удаление элементов оформить в виде функций. Предусмотреть сообщения об ошибках при открытии файла и выполнении операций ввода/вывода.

**Код программы**

#include <iostream>

#include <fstream>

#include <string>

using namespace std;

struct Student

{

void setLFM();

void setDMY();

void setAddress();

void setRtg();

string last\_name, first\_name, middle\_name;

string day, month, year;

string country, city, street, number\_house, apartment;

string rating;

};

void Student::setLFM()

{

cout << "Your name: ";

cin >> last\_name >> first\_name >> middle\_name;

}

void Student::setDMY()

{

cout << "Your date: ";

cin >> day >> month >> year;

}

void Student::setAddress()

{

cout << "Your country: ";

cin >> country;

cout << "Your city: ";

cin >> city;

cout << "Your street: ";

cin >> street;

cout << "Your house: ";

cin >> number\_house;

cout << "Your apartment: ";

cin >> apartment;

}

void Student::setRtg()

{

cout << "Your rating: ";

cin >> rating;

}

void fillFile(Student\*& Human, ofstream& fout, const string& path, const int& count)

{

fout.open(path);

for (int i = 0; i < count; i++)

{

fout << "Name: " << Human[i].last\_name << " " << Human[i].first\_name << " " << Human[i].middle\_name << '\n';

fout << "Date: " << Human[i].day << "." << Human[i].month << "." << Human[i].year << '\n';

fout << "Adress: " << Human[i].country << ", city." << Human[i].city << ", st." << Human[i].street << ", " << Human[i].number\_house << ", " << Human[i].apartment << '\n';

fout << "Rating: " << Human[i].rating;

if (i != count - 1)

fout << "\n\n";

}

fout.close();

}

void removeStudents(Student\*& Human, ofstream& fout, const string& path, int& count)

{

int\* indexStudent = new int[count];

int index = 0;

bool f = 0;

for (int i = 0; i < count; ++i)

{

f = 0;

for (int j = 0; j < count && f == 0; ++j)

if (i != j)

if (Human[i].day == Human[j].day && Human[i].month == Human[j].month && Human[i].year == Human[j].year)

f = 1;

if (f == 0)

{

indexStudent[index] = i;

++index;

}

}

if (index != count)

{

count = index;

Student\* Human2 = new Student[index];

for (int i = 0; i < index; ++i)

Human2[i] = Human[indexStudent[i]];

delete[] Human;

Human = Human2;

fillFile(Human, fout, path, index);

}

delete[] indexStudent;

}

void fillStruct(Student\*& Human, const int& with, const int& to)

{

for (int i = with; i < to; ++i)

{

cout << "Student " << i + 1 << ":\n";

Human[i].setLFM();

Human[i].setDMY();

Human[i].setAddress();

Human[i].setRtg();

if (i != to - 1)

cout << "\n";

}

}

void addStudentBefore(Student\*& Human, ofstream& fout, const string& path, int& count)

{

if (count != 0)

{

string last\_name;

cin >> last\_name;

cout << "\n";

const int size = count + 1;

Student\* Human2 = new Student[size];

int index1 = 0;

bool f = 0;

for (int i = 0; i < count; ++i)

if (Human[index1].last\_name == last\_name && f != 1)

{

fillStruct(Human2, i, i + 1);

f = 1;

++count;

}

else

{

Human2[i] = Human[index1];

++index1;

}

if (f == 1)

{

delete[] Human;

Human = Human2;

fillFile(Human, fout, path, size);

}

else

delete[] Human2;

}

}

void addStudentToEnd(Student\*& Human, ofstream& fout, const string& path, int& count)

{

const int size = count + 1;

Student\* Human2 = new Student[size];

for (int i = 0; i < count; ++i)

Human2[i] = Human[i];

fillStruct(Human2, count, size);

++count;

delete[] Human;

Human = Human2;

fillFile(Human, fout, path, size);

}

void showFile(ifstream& fin, const string& path, int& count)

{

if (count == 0)

cout << "File is empty!\n\n";

else

{

fin.open(path);

if (!fin.is\_open())

return;

char ch;

cout << "Imported students:\n\n";

while (fin.get(ch))

cout << ch;

cout << "\n\n";

fin.close();

}

}

int checkEnterPosInt(string& str\_input, string str\_show = "")

{

int numeric;

while (true)

{

try

{

cout << str\_show; cin >> str\_input;

numeric = stoi(str\_input);

int i, temp = numeric;

for (i = 0; temp > 0; ++i)

temp /= 10;

if (str\_input.length() == i)

break;

}

catch (...) { system("cls"); cout << "ERROR.\n\n"; }

}

return numeric;

}

int main()

{

system("chcp 1251 >> null");

ofstream fout;

ifstream fin;

string path = "fileStudent.txt";

int count = 0;

Student\* Human = new Student[count];

bool f = 1;

while (f)

{

string choice\_enter,

choice\_show = "1 - Add student\n2 - Add student before\n3 - Remove student with the same date\n4 - Import data\n5 - Close\n";

int choice = checkEnterPosInt(choice\_enter, choice\_show);

cout << "\n";

switch (choice)

{

case 1:

system("cls");

addStudentToEnd(Human, fout, path, count);

break;

case 2:

system("cls");

showFile(fin, path, count);

addStudentBefore(Human, fout, path, count);

break;

case 3:

system("cls");

removeStudents(Human, fout, path, count);

break;

case 4:

system("cls");

showFile(fin, path, count);

break;

case 5:

f = 0;

delete[] Human;

break;

default:

system("cls"); cout << "Ошибка ввода. Повторите попытку.\n";

break;

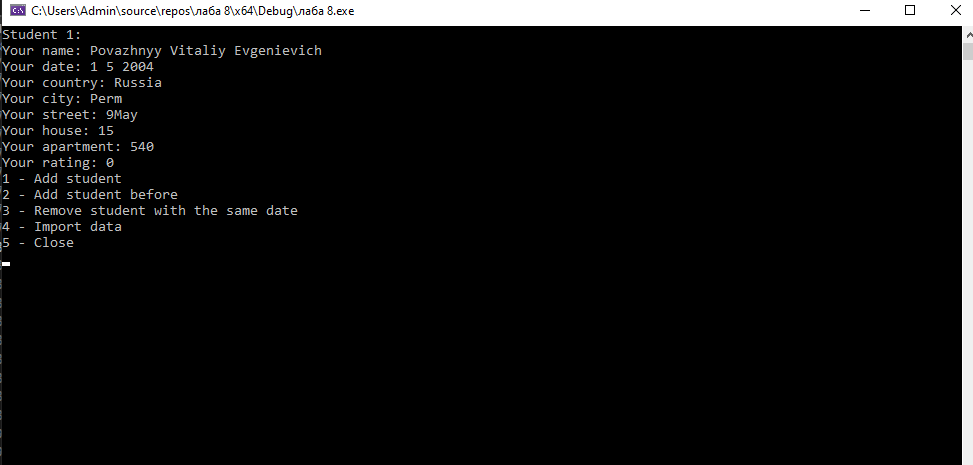
}

}

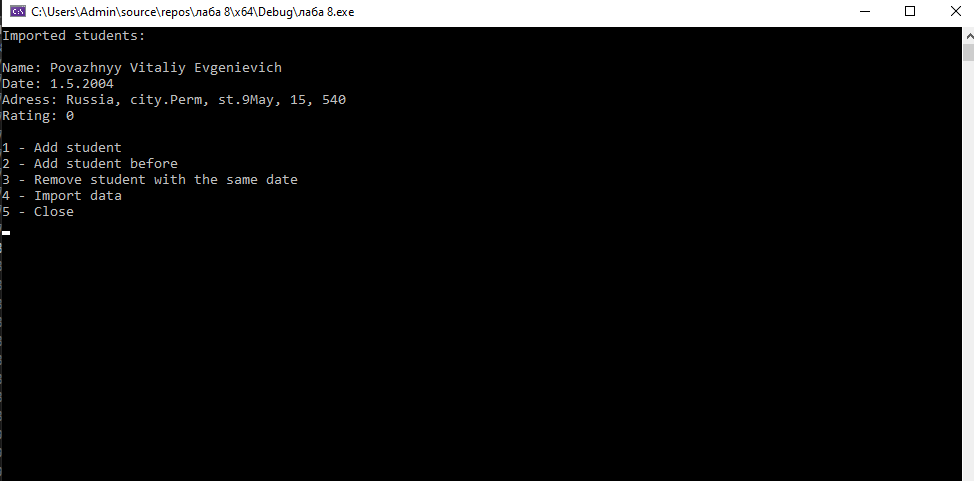
return 0;

}

**Вывод программы**

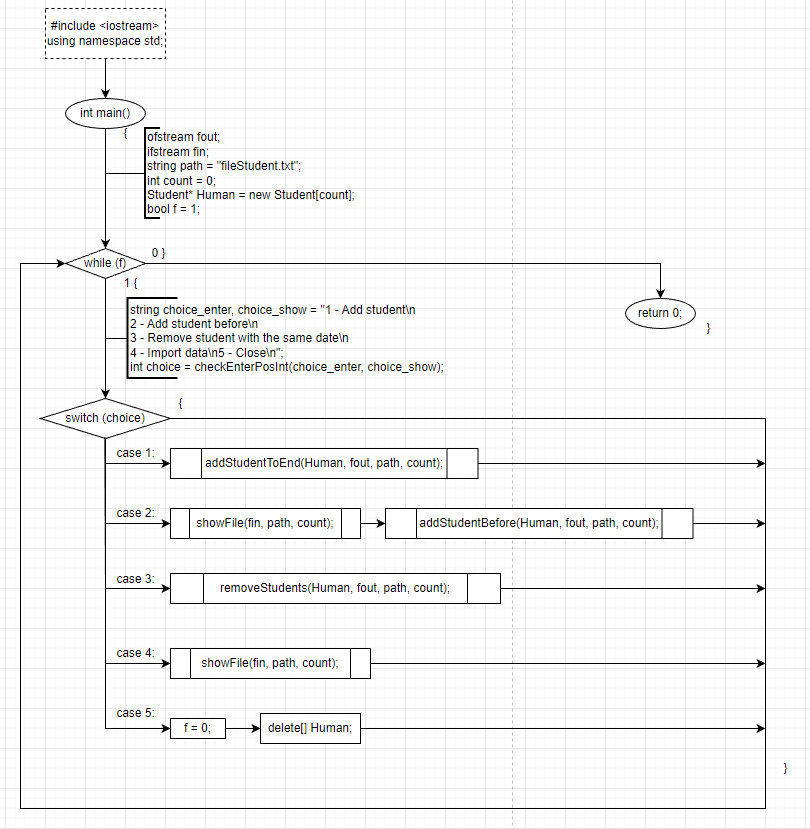
****

***Рисунок 1 – вывод программы с вводом студента***

****

***Рисунок 2 – вывод программы с загрузкой студентов из файла***

**Блок-схема**

****

***Рисунок 3 – Схема алгоритма программы***

**Вывод**

Программа выполняет свою задачу.

**Заключение**

Для решения задачи потребовались знания языка программирования, а конкретнее C++. Программа выполняет те условия, что были указаны в постановке задачи и работает без проблем.