Министерство науки и высшего образования Российской Федерации

Федеральное государственное автономное учреждение высшего образования

«Пермский национальный исследовательский политехнический университет»

ПНИПУ

**ЛАБОРАТОРНАЯ РАБОТА**

«Лаб. 8 - Стуктуры данных»

**Выполнил:**

студент группы РИС-23-2б

Борисов Никита Андреевич

**Проверила:**

доцент кафедры ИТАС

О.А. Полякова

Пермь, 2024 г.

**Задача:**

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.

По варианту 24 дана структура "Студент":

* фамилия, имя, отчество;
* номер телефона;
* группа;
* оценки по 3 основным предметам.

Удалить все элементы из группы с указанным номером, у которых среднее арифметическое оценок меньше заданного, добавить элемент после элемента с заданной фамилией.

**Анализ:**

В данной задаче требуется работать с файлами и структурами данных.

Программа начинается с открытия файла `Students.txt` и считывания информации о студентах. Для этого используется структура `Student`, которая содержит информацию о студентах: фамилия, имя, отчество, номер телефона, группа и оценки по трём предметам. Информация о студентах считывается построчно из файла, а затем каждая строка анализируется для извлечения отдельных данных о студенте. Для этого используются различные разделители, такие как пробелы. Извлеченные данные затем используются для заполнения структуры `Student`, которая добавляется в вектор `students`.

После того как данные о студентах были считаны из файла и добавлены в вектор `students`, программа выводит текущую информацию о студентах на экран.

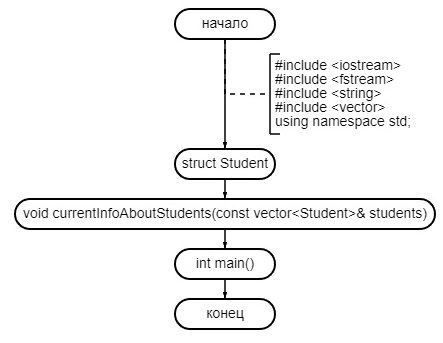
После вывода текущей информации пользователю предлагается ввести номер телефона студента, которого нужно удалить из списка. Программа ищет студента с соответствующим номером телефона в векторе `students` и удаляет его, если он найден.

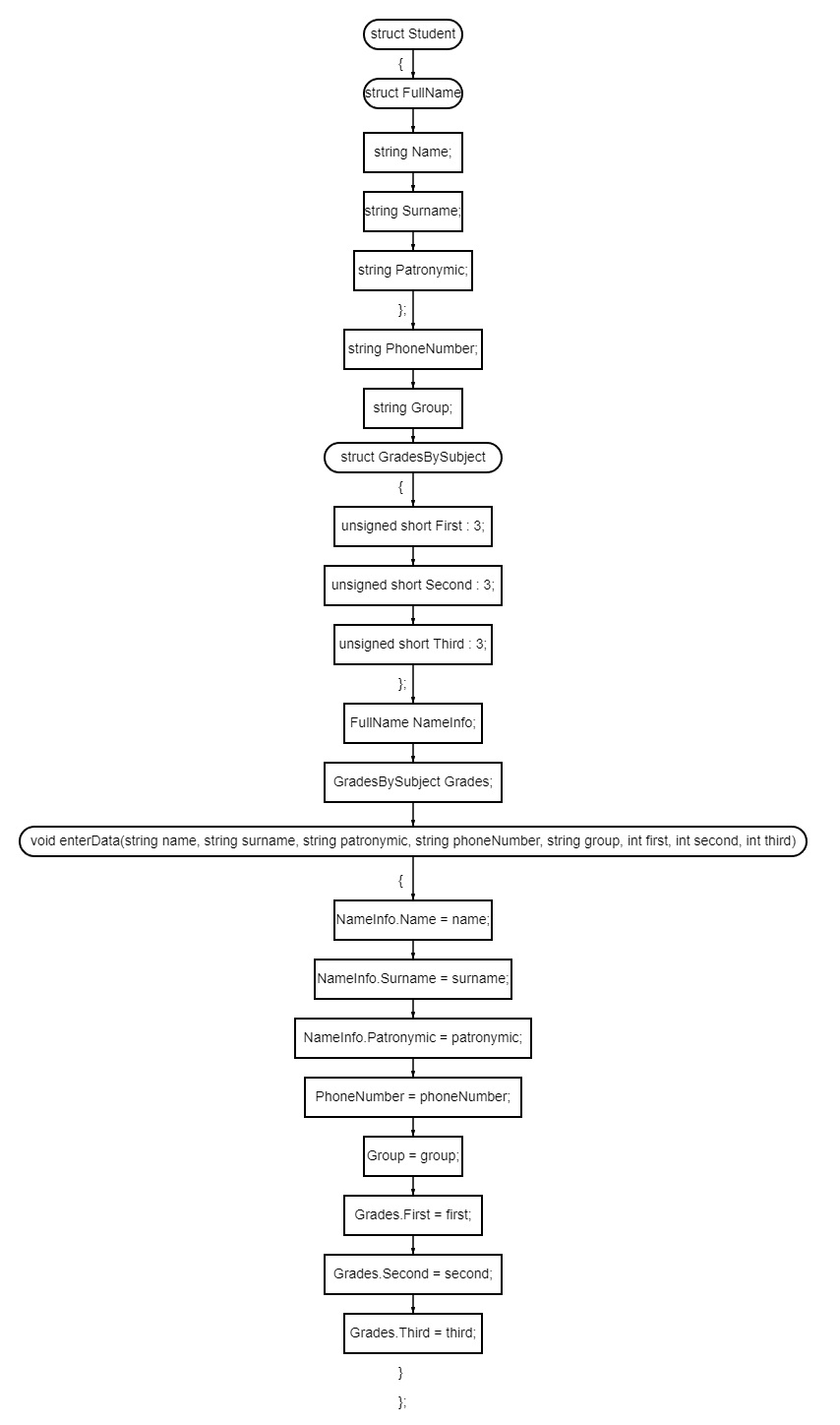
Затем программа предлагает пользователю ввести порог для среднего арифметического оценок студентов. Все студенты, у которых среднее арифметическое оценок по трём предметам меньше заданного порога, удаляются из списка.

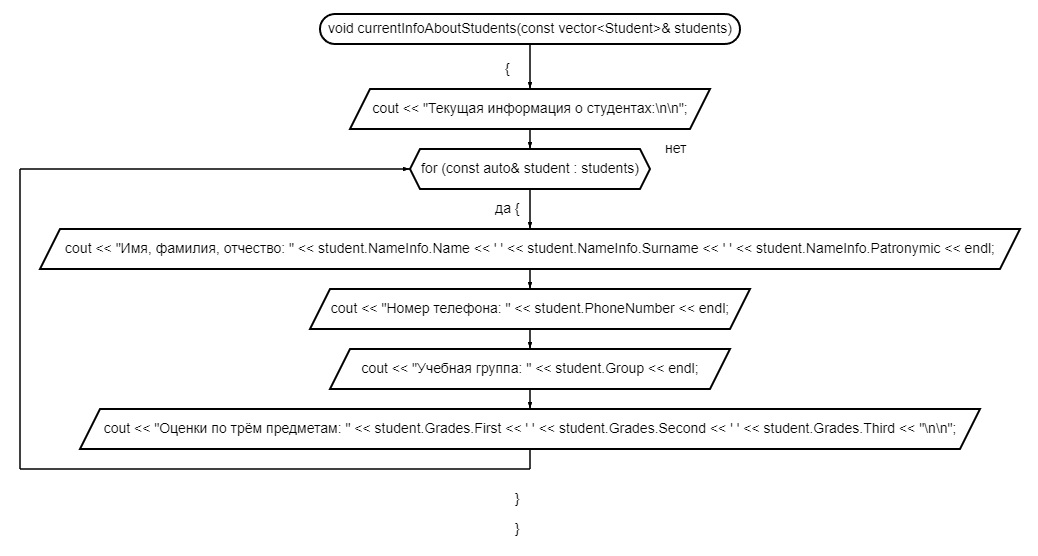
После этого программа предлагает пользователю ввести фамилию студента, после которого нужно добавить нового студента в список. Если студент с указанной фамилией найден в списке, программа запрашивает данные о новом студенте и вставляет его в список после найденного студента.

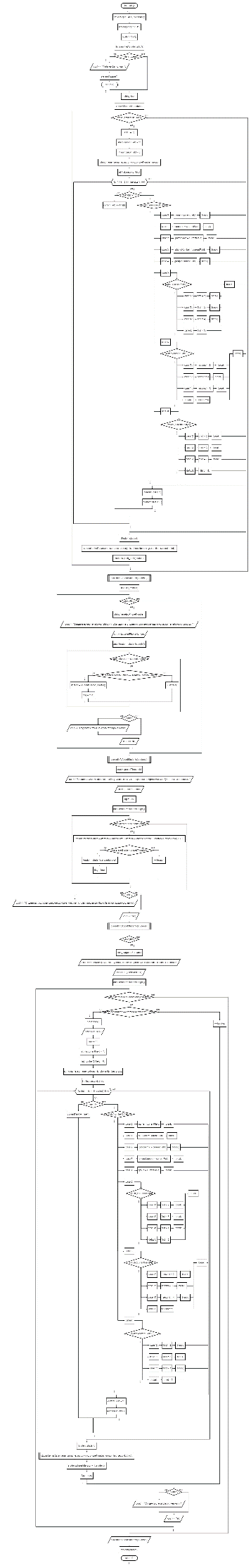
По завершении всех операций программа выводит на экран текущую информацию о студентах после каждой операции.

**Моделирование:**

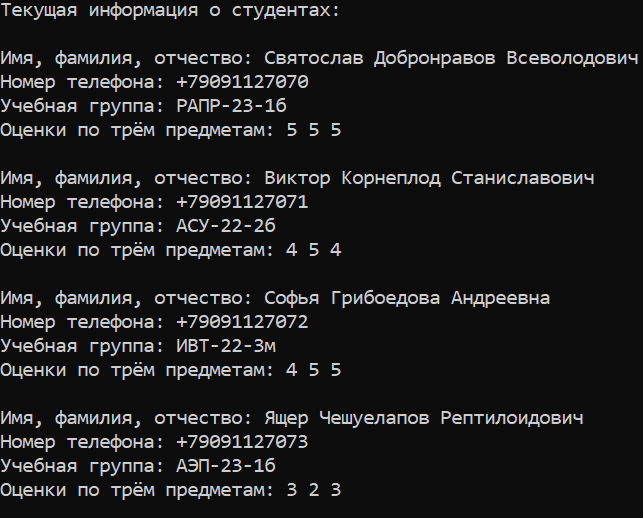


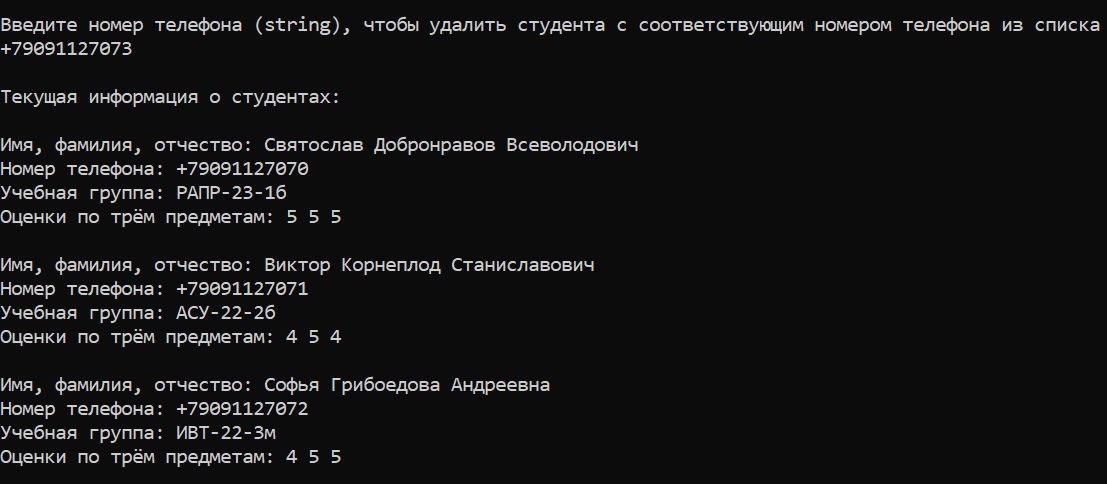


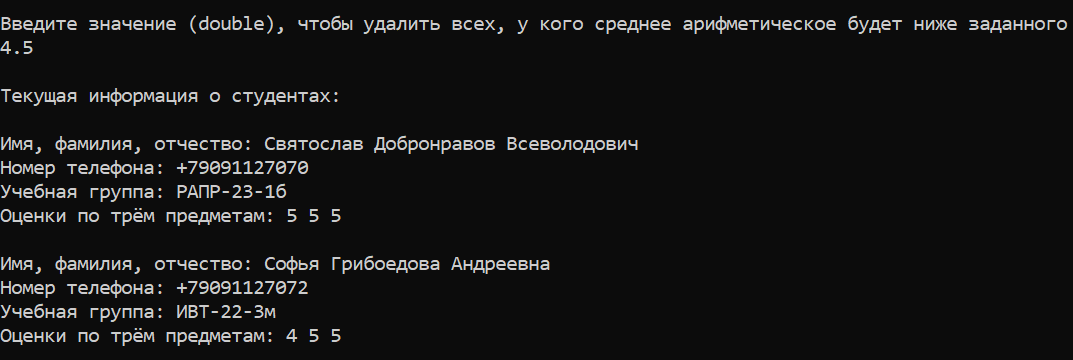


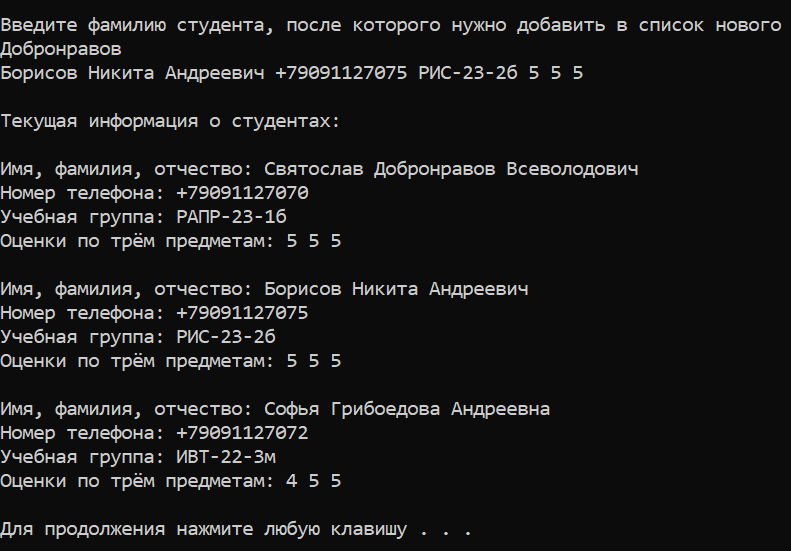


**Результаты работы программы:**









**Код программы:**

#include <iostream>

#include <fstream>

#include <string>

#include <vector>

using namespace std;

struct Student

{

struct FullName

{

string Name;

string Surname;

string Patronymic;

};

string PhoneNumber;

string Group;

struct GradesBySubject

{

unsigned short First : 3;

unsigned short Second : 3;

unsigned short Third : 3;

};

FullName NameInfo;

GradesBySubject Grades;

void enterData(string name, string surname, string patronymic, string phoneNumber, string group, int first, int second, int third)

{

NameInfo.Name = name;

NameInfo.Surname = surname;

NameInfo.Patronymic = patronymic;

PhoneNumber = phoneNumber;

Group = group;

Grades.First = first;

Grades.Second = second;

Grades.Third = third;

}

};

void currentInfoAboutStudents(const vector<Student>& students)

{

cout << "Текущая информация о студентах:\n\n";

for (const auto& student : students)

{

cout << "Имя, фамилия, отчество: " << student.NameInfo.Name << ' ' << student.NameInfo.Surname << ' ' << student.NameInfo.Patronymic << endl;

cout << "Номер телефона: " << student.PhoneNumber << endl;

cout << "Учебная группа: " << student.Group << endl;

cout << "Оценки по трём предметам: " << student.Grades.First << ' ' << student.Grades.Second << ' ' << student.Grades.Third << "\n\n";

}

}

int main()

{

setlocale(LC\_ALL, "Russian");

system("chcp 1251");

system("cls");

// Перенос данных из файла в программу

ifstream file("Students.txt");

if (!file.is\_open())

{

cout << "Файл не был открыт";

system("pause");

return 0;

}

string line;

vector<Student> students;

while (getline(file, line))

{

line += ' ';

string currentField = "";

int numberOfField = 0;

string name, surname, patronymic, phoneNumber, group;

int first, second, third;

for (int i = 0; i < line.size(); i++)

{

if (line[i] != ' ')

currentField += line[i];

else

{

switch (numberOfField)

{

case 0:

name = currentField;

break;

case 1:

surname = currentField;

break;

case 2:

patronymic = currentField;

break;

case 3:

phoneNumber = currentField;

break;

case 4:

group = currentField;

break;

case 5:

switch (currentField[0])

{

case '2':

first = 2;

break;

case '3':

first = 3;

break;

case '4':

first = 4;

break;

default:

first = 5;

}

break;

case 6:

switch (currentField[0])

{

case '2':

second = 2;

break;

case '3':

second = 3;

break;

case '4':

second = 4;

break;

default:

second = 5;

}

break;

default:

switch (currentField[0])

{

case '2':

third = 2;

break;

case '3':

third = 3;

break;

case '4':

third = 4;

break;

default:

third = 5;

}

}

currentField = "";

numberOfField++;

}

}

Student student;

student.enterData(name, surname, patronymic, phoneNumber, group, first, second, third);

students.push\_back(student);

}

currentInfoAboutStudents(students);

// Удаление студента с соответствующим номером телефона

bool flag = false;

while (!flag) {

string targetedPhoneNumber;

cout << "Введите номер телефона (string), чтобы удалить студента с соответствующим номером телефона из списка\n";

cin >> targetedPhoneNumber;

auto iterator = students.begin();

while (iterator != students.end())

if (targetedPhoneNumber == iterator->PhoneNumber)

{

iterator = students.erase(iterator);

flag = true;

}

else

++iterator;

if (!flag)

cout << "Студента с этим номером телефона нет\n";

cout << "\n";

}

currentInfoAboutStudents(students);

// Удаление студентов, у которых среднее арифметическое по трём предметам меньше заданного

double presetThreshold;

cout << "Введите значение (double), чтобы удалить всех, у кого среднее арифметическое будет ниже заданного\n";

cin >> presetThreshold;

flag = false;

auto iterator = students.begin();

while (iterator != students.end())

{

double averageGrade = (iterator->Grades.First + iterator->Grades.Second + iterator->Grades.Third) / 3.0;

if (averageGrade < presetThreshold)

{

iterator = students.erase(iterator);

flag = true;

}

else

++iterator;

}

if (!flag)

cout << "Студентов, у которых среднее арифметическое по трём предметам было бы ниже указанного, нет\n";

cout << "\n";

currentInfoAboutStudents(students);

// Добавление элемента в список студентов после студента с заданной фамилией

flag = false;

while (!flag) {

string targetedSurname;

cout << "Введите фамилию студента, после которого нужно добавить в список нового\n";

cin >> targetedSurname;

auto iterator = students.begin();

while (iterator != students.end() && !flag)

if (targetedSurname == iterator->NameInfo.Surname)

{

cin.ignore();

getline(cin, line);

line += ' ';

string currentField = "";

int numberOfField = 0;

string name, surname, patronymic, phoneNumber, group;

int first, second, third;

for (int i = 0; i < line.size(); i++)

{

if (line[i] != ' ')

currentField += line[i];

else

{

switch (numberOfField)

{

case 0:

name = currentField;

break;

case 1:

surname = currentField;

break;

case 2:

patronymic = currentField;

break;

case 3:

phoneNumber = currentField;

break;

case 4:

group = currentField;

break;

case 5:

switch (currentField[0])

{

case '2':

first = 2;

break;

case '3':

first = 3;

break;

case '4':

first = 4;

break;

default:

first = 5;

}

break;

case 6:

switch (currentField[0])

{

case '2':

second = 2;

break;

case '3':

second = 3;

break;

case '4':

second = 4;

break;

default:

second = 5;

}

break;

default:

switch (currentField[0])

{

case '2':

third = 2;

break;

case '3':

third = 3;

break;

case '4':

third = 4;

break;

default:

third = 5;

}

}

currentField = "";

numberOfField++;

}

}

Student student;

student.enterData(name, surname, patronymic, phoneNumber, group, first, second, third);

students.insert(iterator + 1, student);

flag = true;

}

else

++iterator;

if (!flag)

cout << "Студента с этой фамилией нет\n";

cout << "\n";

}

currentInfoAboutStudents(students);

system("pause");

return 1;

}

**Ссылка на Git:** <https://github.com/Exateym/Study>