Министерство науки и высшего образования Российской Федерации

Пермский национальный исследовательский политехнический университет

Электротехнический факультет

Кафедра Информационные технологии и автоматизированные системы

Лабораторная работа №10

по дисциплине

«Программирование»

Вариант 8

Выполнил: студент группы Асу15-бз

Чиков Дмитрий Александрович

Проверил: \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

г. Пермь – 2020

**Цель работы:**

Написать приложение для работы с простой базой данных, хранящей информацию об объекте на внешнем носителе. Приложение должно выполнять следующие функции:

1. Создание базы данных, содержащей записи указанного формата.
2. Просмотр базы данных.
3. Удаление элементов из базы данных (по ключу/ по номеру).
4. Корректировка элементов в базе данных (по ключу / по номеру).
5. Добавление элементов в базу данных (в начало / в конец/ с заданным номером).
6. Выполнение задания, указанного в варианте.

**Задачи работы:**

Списки студентов нескольких групп 1 курса хранятся в отдельных файлах – один файл на одну группу. Программа должна

* 1. Добавлять, удалять, корректировать, позволять просматривать записи файлов.
  2. Создавать новый файл, в котором будет храниться общий список студентов 1 курса, отсортированный по убыванию номеров групп.

Для выполнения работы были реализованы следующие функции:

struct student

student\* Add\_to\_end(int n) { //функция добавления записей в файлы групп в конец

void Add\_to\_file(int n) { //функция добавления записей в файлы групп в любое место

void Add() – основная функция добавления записей

void Read\_group() – чтение списка групп

void Read\_students() – чтение списка студентов

int search\_line(string filename, string line\_search) – поиск строки

string remove\_line(string filename, int i\_number\_line\_delete) – удаление строки

void modify\_line(string filename, int i\_number\_line) – изменение строки

void remove\_students() – удаление студентов

void modify\_students() – изменение студентов

void Create\_sum() – создание общего файла студентов

int main() – основная функция.

Листинг приведен в приложении А.

Тестирование:

Создание записей о студентах представлено на рисунке 1: сначала пользователь вводит количество добавляемых студентов, выбирает куда будут добавляться записи и затем, следуя подсказкам на экране вводит информацию о них.

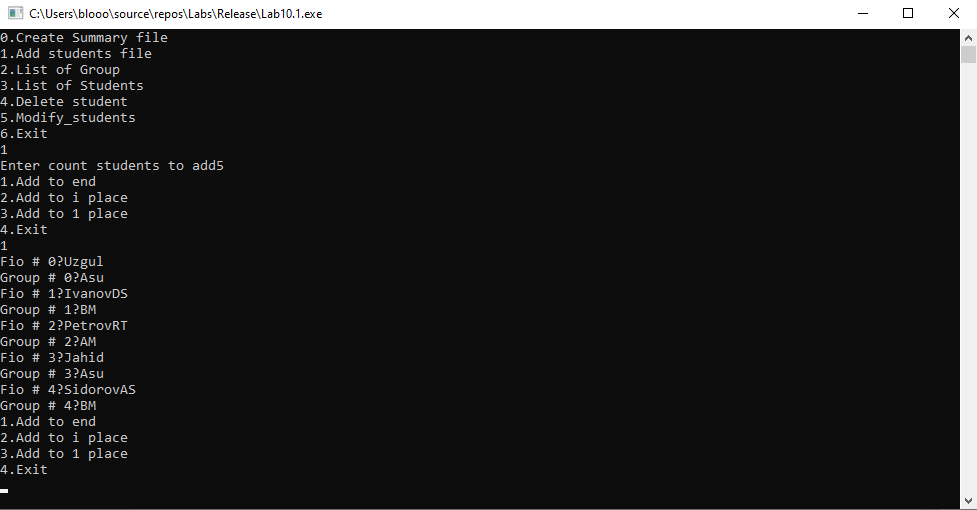


Рисунок 1 Ввод информации о студентах

В результате ввода данных было создано 4 файла: 3 файла по группам и 1 файл(main.txt), содержащий список групп. Результат работы программы представлен на рисунке 2.

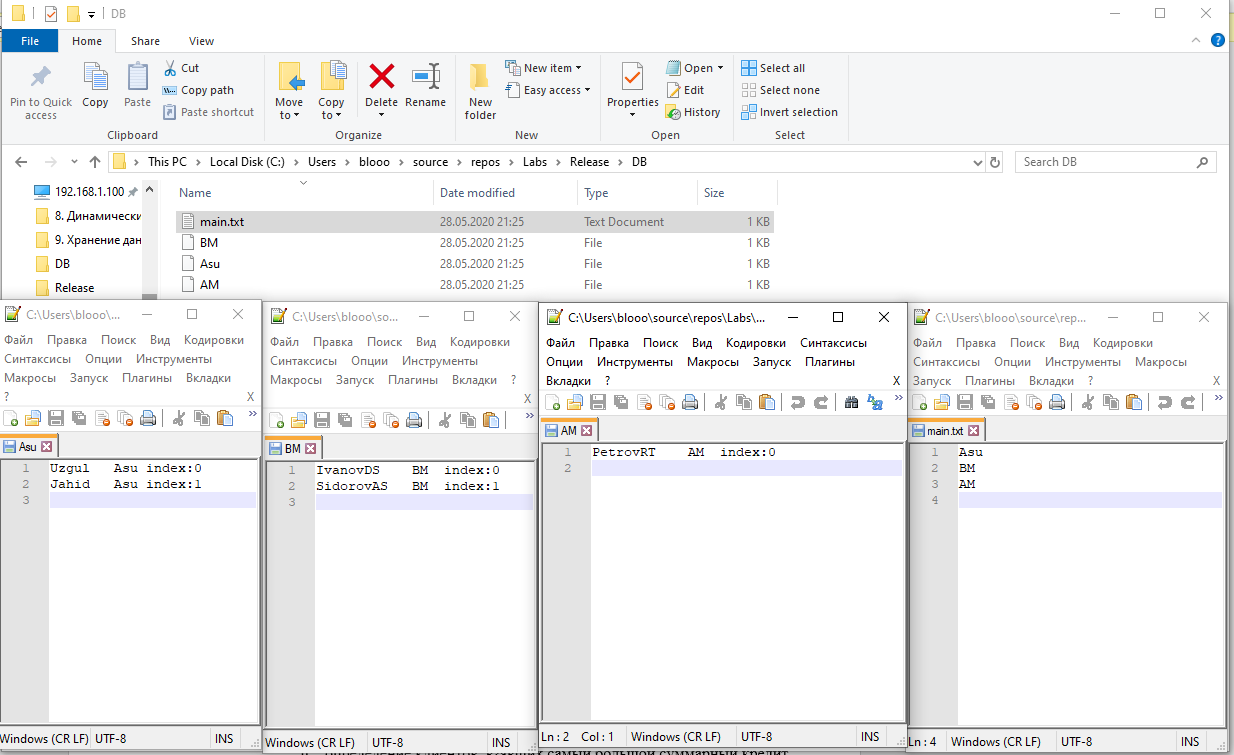


Рисунок 2 Результат работы программы

Функция просмотра записей делится на 2 части: просмотр списка групп и просмотр списка студентов определенной группы. Просмотр списка студентов представлен на рисунке 3. Пользователь должен ввести наименование группы, после чего отобразится список этой группы.

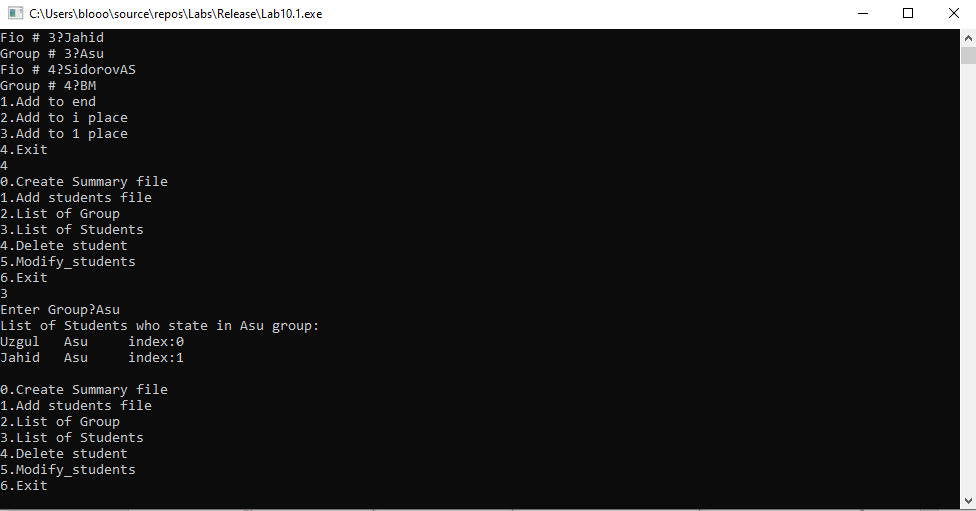


Рисунок 3 Просмотр списка студентов

Изменение информации о студентах выполняется либо по ФИО студента либо по его номеру в списке. Изменение студента по ФИО представлено на рисунке 4, по номеру на рисунке 5.

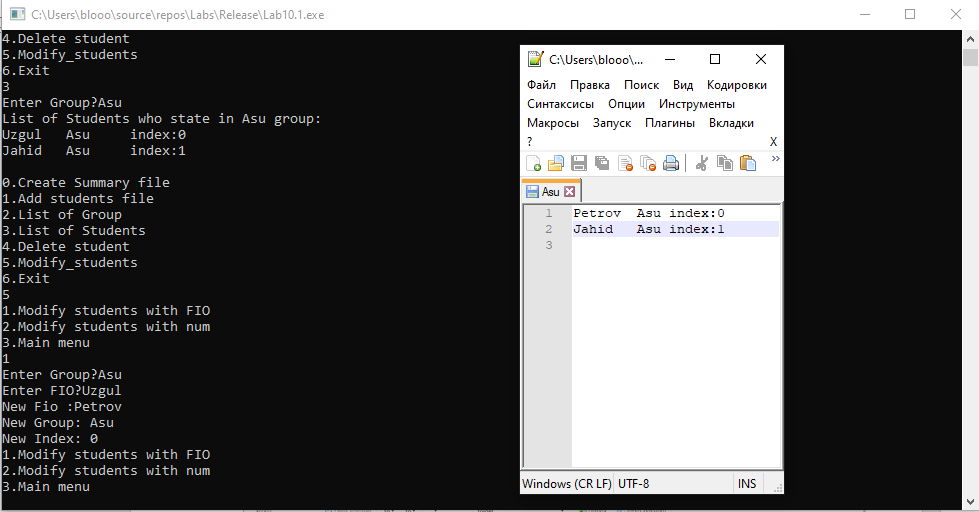


Рисунок 4 Изменение записи о студенте по ФИО

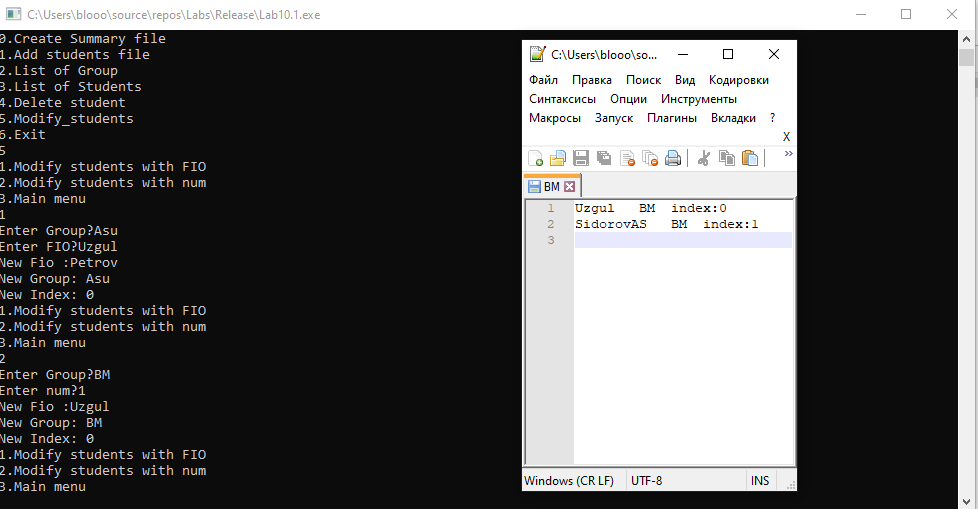


Рисунок 5 Изменение записи о студенте по номеру

Удаление студентов так же осуществляется по ФИО или по номеру. Результат работы программы представлен на рисунках 6 и 7.

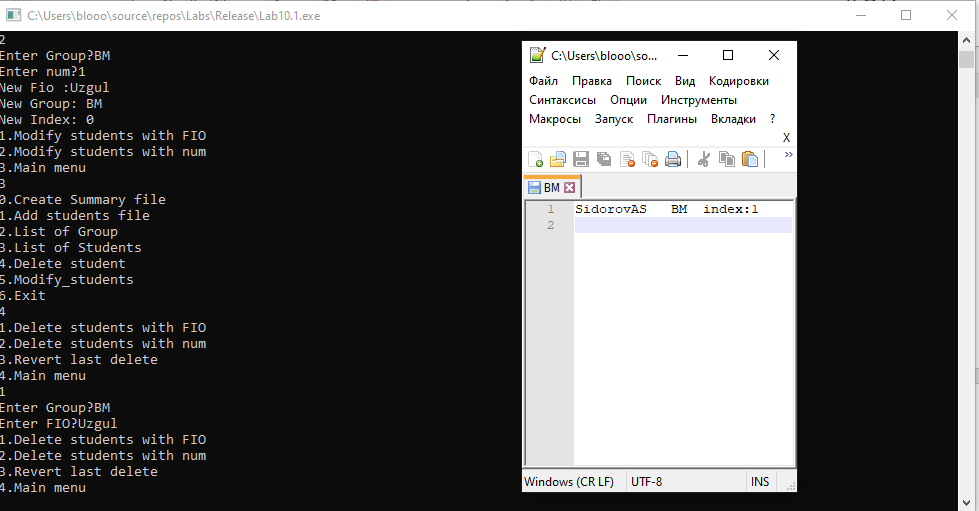


Рисунок 6 Удаление студента по ФИО

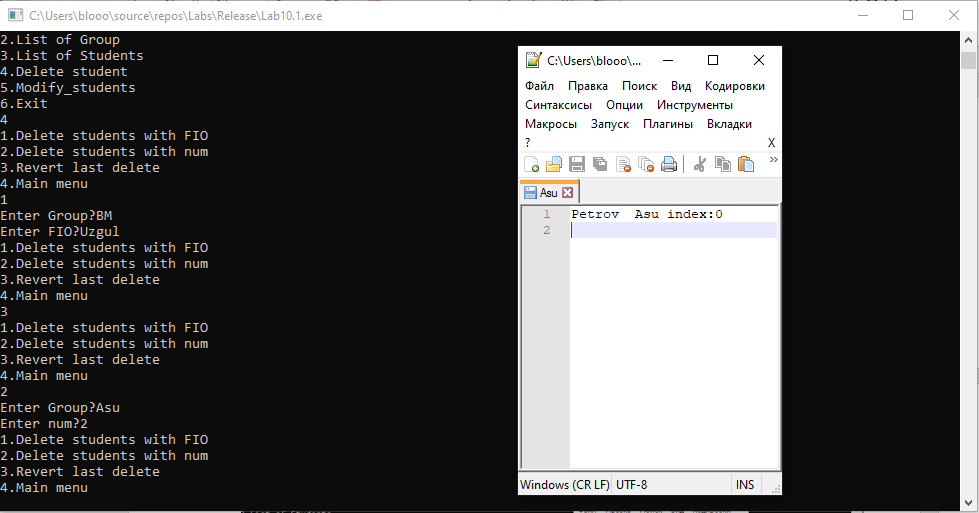


Рисунок 7 Удаление студента по номеру

Создание общего списка студентов происходит по списку групп в файле main.txt. Результат работы программы представлен на рисунке 8.

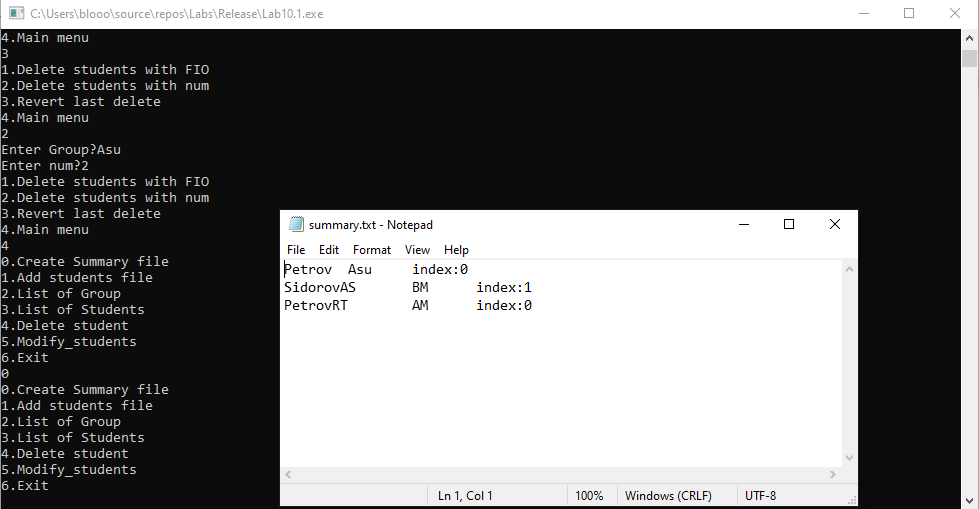


Рисунок 8 Создание общего списка студентов

**Приложение А**

#include <stdio.h>

#include <fstream>

#include <algorithm>

using namespace std;

#include <iostream>

#include <list>

#include <sstream>

#include <string>

#include <iterator>

#include <vector>

struct student

{

char group[255];

char fio[255];

int index;

};

std::string filepath("C:/Users/blooo/source/repos/Labs/Release/DB/");

student\* Add\_to\_end(int n) { //функция создания файлов групп

student\* a;

list<const char\*>::iterator it;

a = new student[n];//создаем динамический массив

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

{

cout << "Fio # " << i << "?";

cin >> a[i].fio;

cout << "Group # " << i << "?";

cin >> a[i].group;

std::string filename = filepath + a[i].group;

std::ifstream file\_in(filename);

int i\_number\_line\_now = 0; //счётчик строк

std::string line; //для хранения строки

std::string line\_file\_text; //для хранения текста файла

while (getline(file\_in, line))

{

i\_number\_line\_now++;

line\_file\_text.insert(line\_file\_text.size(), line); /\*дабавить строку\*/

/\*добавить перенос на слудующую строку\*/

line\_file\_text.insert(line\_file\_text.size(), "\r\n");

}

file\_in.close();

fstream f(filename, ios::app);//двунаправленный файловый поток с установкой указателя в конец файла

f << a[i].fio; f << "\t";

f << a[i].group; f << "\t";

f << "index:"; f << i\_number\_line\_now; f << "\n";

f.close();//закрытие потока

std::string filename2 = filepath + "main.txt";

ifstream file(filename2);

istream\_iterator<string> eof;

bool found = find(istream\_iterator<string>(file), eof, a[i].group) != eof;

if (!found)

{

fstream f(filename2, ios::app);//двунаправленный файловый поток с установкой указателя в конец файла

f << a[i].group; f << "\n";

f.close();//закрытие потока

}

}

return a;

}

void Add\_to\_file(int n) { //функция создания файлов групп

std::vector<std::string> vec;

int index, i\_number\_line\_now = 0;

std::string add, fio, group; //для хранения текста изменения

cout << "New Fio :";

cin >> fio;

cout << "New Group: ";

cin >> group;

cout << "New Index: ";

cin >> index;

std::string filename = filepath + group;

std::ifstream file\_in(filename);

if (!file\_in)

{

std::cerr << "Error. Can't open file" << std::endl;

}

std::string line; //для хранения строки

std::string line\_file\_text; //для хранения текста файла

while (getline(file\_in, line))

{

add = fio + "\t" + group + "\t" + "index:" + to\_string(i\_number\_line\_now);

if (!(i\_number\_line\_now == n))

{

line\_file\_text.insert(line\_file\_text.size(), line); /\*дабавить строку\*/

/\*добавить перенос на слудующую строку\*/

line\_file\_text.insert(line\_file\_text.size(), "\r\n");

}

else {

line\_file\_text.insert(line\_file\_text.size(), add);

line\_file\_text.insert(line\_file\_text.size(), "\r\n");

line\_file\_text.insert(line\_file\_text.size(), line); /\*дабавить строку\*/

/\*добавить перенос на слудующую строку\*/

line\_file\_text.insert(line\_file\_text.size(), "\r\n");

}

i\_number\_line\_now++;

}

file\_in.close();

//теперь в line\_file\_text будет содержаться измененный текст файла, теперь можно перезаписать файл

std::ofstream file\_out;

file\_out.open(filename, std::ios::trunc | std::ios::binary); //открыть и обрезать

//записать

file\_out.write(line\_file\_text.c\_str(), line\_file\_text.size());

file\_out.clear();

std::string filename2 = filepath + "main.txt";

ifstream file(filename2);

istream\_iterator<string> eof;

bool found = find(istream\_iterator<string>(file), eof, group) != eof;

if (!found)

{

fstream f(filename2, ios::app);//двунаправленный файловый поток с установкой указателя в конец файла

f << group; f << "\n";

f.close();//закрытие потока

}

}

void Add()

{

int n, k, j;

cout << "Enter count students to add";

cin >> n;

student\* a = new student[n];

do

{

std::cout << "1.Add to end\n";

std::cout << "2.Add to i place\n";

std::cout << "3.Add to 1 place\n";

std::cout << "4.Exit\n";

std::cin >> k;

switch (k)

{

case 1: a = Add\_to\_end(n); break;

case 2: for (int i = 0; i < n; i++)

{

cout << "Enter place";

cin >> j;

Add\_to\_file(j);

}

break;

case 3: for (int i = 0; i < n; i++) Add\_to\_file(0); break;

}

} while (k != 4);//выход

}

void Read\_group()

{

std::string filename = filepath + "main.txt";

std::string line;

ifstream f;

f.open(filename, ios::in);//двунаправленный файловый поток

if (f.is\_open())

cout << "List of Group:" << std::endl;

{

while (getline(f, line))

{

cout << line << std::endl;

}

}

cout << std::endl;

f.close();//закрытие потока

}

void Read\_students()

{

std::string line, students;

cout << "Enter Group?";

cin >> line;

std::string filename = filepath + line;

ifstream f;

f.open(filename, ios::in);//двунаправленный файловый поток

if (f.is\_open()) {

cout << "List of Students who state in " << line << " group:" << std::endl;

{

while (getline(f, students))

{

cout << students << std::endl;

}

}

cout << std::endl;

f.close();//закрытие потока

}

else

cout << "Error! Same group not exist" << std::endl;

}

int search\_line(string filename, string line\_search) //процедура для окончательного удаления строк

{

std::vector<std::string> vec;

std::ifstream file\_in(filename);

if (!file\_in)

{

std::cerr << "Error. Can't open file" << std::endl;

}

int i\_number\_line\_now = 0; //счётчик строк

int searched\_line = 0;

std::string line; //для хранения строки

std::string line\_file\_text; //для хранения текста файла

while (getline(file\_in, line))

{

i\_number\_line\_now++;

int check = line.find(line\_search);

if (check == -1)

{

line\_file\_text.insert(line\_file\_text.size(), line); /\*дабавить строку\*/

/\*добавить перенос на слудующую строку\*/

line\_file\_text.insert(line\_file\_text.size(), "\r\n");

}

else

{

line\_file\_text.insert(line\_file\_text.size(), line); /\*дабавить строку\*/

/\*добавить перенос на слудующую строку\*/

line\_file\_text.insert(line\_file\_text.size(), "\r\n");

searched\_line = i\_number\_line\_now;

}

}

file\_in.close();

//теперь в line\_file\_text будет содержаться измененный текст файла, теперь можно перезаписать файл

std::ofstream file\_out;

file\_out.open(filename, std::ios::trunc | std::ios::binary); //открыть и обрезать

//записать

file\_out.write(line\_file\_text.c\_str(), line\_file\_text.size());

file\_out.clear();

return searched\_line;

}

string remove\_line(string filename, int i\_number\_line\_delete) //процедура для окончательного удаления строк

{

std::vector<std::string> vec;

std::ifstream file\_in(filename);

if (!file\_in)

{

std::cerr << "Error. Can't open file" << std::endl;

}

int i\_number\_line\_now = 0; //счётчик строк

std::string line; //для хранения строки

std::string line\_file\_text; //для хранения текста файла

std::string last; //для хранения удаляемой строки

while (getline(file\_in, line))

{

i\_number\_line\_now++;

if (!(i\_number\_line\_now == i\_number\_line\_delete))

{

line\_file\_text.insert(line\_file\_text.size(), line); /\*дабавить строку\*/

/\*добавить перенос на слудующую строку\*/

line\_file\_text.insert(line\_file\_text.size(), "\r\n");

}

else last = line;

}

file\_in.close();

//теперь в line\_file\_text будет содержаться измененный текст файла, теперь можно перезаписать файл

std::ofstream file\_out;

file\_out.open(filename, std::ios::trunc | std::ios::binary); //открыть и обрезать

//записать

file\_out.write(line\_file\_text.c\_str(), line\_file\_text.size());

file\_out.clear();

return last;

}

void modify\_line(string filename, int i\_number\_line) //процедура для окончательного удаления строк

{

std::vector<std::string> vec;

std::ifstream file\_in(filename);

std::string change, fio, group; //для хранения текста изменения

int index;

if (!file\_in)

{

std::cerr << "Error. Can't open file" << std::endl;

}

cout << "New Fio :";

cin >> fio;

cout << "New Group: ";

cin >> group;

cout << "New Index: ";

cin >> index;

change = fio + "\t" + group + "\t" + "index:" + to\_string(index);

int i\_number\_line\_now = 0; //счётчик строк

std::string line; //для хранения строки

std::string line\_file\_text; //для хранения текста файла

while (getline(file\_in, line))

{

i\_number\_line\_now++;

if (!(i\_number\_line\_now == i\_number\_line))

{

line\_file\_text.insert(line\_file\_text.size(), line); /\*дабавить строку\*/

/\*добавить перенос на слудующую строку\*/

line\_file\_text.insert(line\_file\_text.size(), "\r\n");

}

else {

line\_file\_text.insert(line\_file\_text.size(), change); /\*дабавить строку\*/

/\*добавить перенос на слудующую строку\*/

line\_file\_text.insert(line\_file\_text.size(), "\r\n");

}

}

file\_in.close();

//теперь в line\_file\_text будет содержаться измененный текст файла, теперь можно перезаписать файл

std::ofstream file\_out;

file\_out.open(filename, std::ios::trunc | std::ios::binary); //открыть и обрезать

//записать

file\_out.write(line\_file\_text.c\_str(), line\_file\_text.size());

file\_out.clear();

}

void remove\_students() { //процедура для поиска строк для удаления

int k, n;

std::string line, students, fio, last;

do

{

std::cout << "1.Delete students with FIO\n";

std::cout << "2.Delete students with num\n";

std::cout << "3.Revert last delete\n";

std::cout << "4.Main menu\n";

std::cin >> k;

switch (k)

{

case 1:

cout << "Enter Group?";

cin >> line;

cout << "Enter FIO?";

cin >> fio;

students = filepath + line;

n = search\_line(students, fio);

last = remove\_line(students, n);

break;

case 2:

cout << "Enter Group?";

cin >> line;

cout << "Enter num?";

cin >> n;

students = filepath + line;

last = remove\_line(students, n);

break;

}

} while (k != 4);//выход

}

void modify\_students() { //процедура для поиска строк для удаления

int k, n;

std::string line, students, fio, last;

do

{

std::cout << "1.Modify students with FIO\n";

std::cout << "2.Modify students with num\n";

std::cout << "3.Main menu\n";

std::cin >> k;

switch (k)

{

case 1:

cout << "Enter Group?";

cin >> line;

cout << "Enter FIO?";

cin >> fio;

students = filepath + line;

n = search\_line(students, fio);

modify\_line(students, n);

break;

case 2:

cout << "Enter Group?";

cin >> line;

cout << "Enter num?";

cin >> n;

students = filepath + line;

modify\_line(students, n);

break;

}

} while (k != 3);//выход

}

void Create\_sum() {

std::string filename = filepath + "main.txt";

std::string filename3 = filepath + "summary.txt";

int index, i\_number\_line\_now = 0;

std::string group, line, line\_file\_text; //для хранения текста изменения

ifstream f(filename, ios::in);

istream\_iterator<string> eof;

//открыли файл со списком групп для чтения

while (getline(f, group))

{

std::string filename2 = filepath + group;

ifstream file\_in(filename2, ios::in);

//открыли файл со списком студентов для чтения

while (getline(file\_in, line))

{

line\_file\_text.insert(line\_file\_text.size(), line); /\*дабавить строку\*/

/\*добавить перенос на слудующую строку\*/

line\_file\_text.insert(line\_file\_text.size(), "\r\n");

}

file\_in.close();

//закрыли файл со студентами, в переменной line\_file\_text теперь весь файл

std::ofstream file\_out;

file\_out.open(filename3, ios::app | ios::binary); //открыть

file\_out.write(line\_file\_text.c\_str(), line\_file\_text.size());//записать

file\_out.clear();

line\_file\_text.clear();

}

f.close();//закрытие потока

}

int main() {

int k;

do

{

std::cout << "0.Create Summary file\n";

std::cout << "1.Add students file\n";

std::cout << "2.List of Group\n";

std::cout << "3.List of Students\n";

std::cout << "4.Delete student\n";

std::cout << "5.Modify\_students\n";

std::cout << "6.Exit\n";

std::cin >> k;

switch (k)

{

case 0: Create\_sum(); break;//Добавление студентов в файлы

case 1: Add(); break;//Добавление студентов в файлы

case 2: Read\_group(); break;//Просмотр групп

case 3: Read\_students(); break;//Просмотр студентов

case 4: remove\_students(); break;//Удаление студентов

case 5: modify\_students(); break;//Изменение студентов

}

} while (k != 6);//выход

}