**ПРИЛОЖЕНИЕ А**

УЧРЕЖДЕНИЕ ОБРАЗОВАНИЯ

«БРЕСТСКИЙ ГОСУДАРСТВЕННЫЙ ТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ»

# КАФЕДРА ИНТЕЛЛЕКТУАЛЬНЫХ ИНФОРМАЦИОННЫХ ТЕХНОЛОГИЙ

ОБРАБОТКА МАССИВОВ СТРУКТУРИРОВАННЫХ ДАННЫХ

**«**ШКОЛЬНИКИ**»**

**ТЕКСТ ПРОГРАММЫ**

КР.АС59.200046-01 12 00

|  |  |
| --- | --- |
| Руководитель | И.Н. Аверина |
| Выполнил  студент 1 курса  группы АС-59 | Т.А. Быбко |
|  |  |
|  |  |
|  |  |

## 2021

**СОДЕРЖАНИЕ**

Kursovaya.cpp – основной модуль, в котором содержится main и функция меню для работы с программой.

Shcolniki.h – модуль, содержащий структуру данных.

AddTakeOut.h, AddTakeOut.cpp – модуль, в котором находятся функции, работающии с добавлением информации, выводом, и считывании

WorkingWithStruct.h, WorkingWithStruct.cpp – модуль, в котором содержаться функции, непосредственно работающие с массивом структур

Sort.h, Sort.cpp – модуль, в котором находится функция сортировки

**Kursovaya.cpp**

#include <iostream>

#include <fstream>

#include <windows.h> //Для функции Sleep(), которая считает в тысячных долях секунды и для локализации текста при вводе и выводе из файла

#include <iomanip> //Для setw()

#include <conio.h> //Для \_getch()

#include "AddTakeOut.h"

#include "Shcolniki.h"

#include "WorkWithStruct.h"

#include "Sort.h"

// "" кавычки пишутся в директиве препроцесора для поиска файла прямо среди файлов нашего проекте

using namespace std;

void Menu(int, School\*&, char\*);

int main()

{

setlocale(LC\_ALL, "Russian");

SetConsoleCP(1251);

SetConsoleOutputCP(1251);

cout << "\n\n\n\n\n\n\n\n\n\n\n\n\n\t\t\t\t\tДобро пожаловать в электронный дневник!" << endl;

Sleep(3000); //Ожидание 3 секунды

char source[] = "Shcolniki.txt"; // Имя основного рабочего файла

int num = 0;

School\* List = new School[num]; // Массив структур

WorkingWithFile(num, List, source); //Считывает исходную информацию с файла

Menu(num, List, source);

delete[] List;

return 0;

}

//Функция меню

void Menu(int num, School\* &List, char\* source)

{

char act;

bool is\_working = true;

do

{

system("cls"); // Очищение командной строки

cout << "Введите номер действия, которое хотите совершить:" << endl;

cout << "1. Вывести текущий список" << endl;

cout << "2. Добавить нового ученика в список" << endl;

cout << "3. Изменить информацию о выбранном ученике" << endl;

cout << "4. Удалить ученика из списка" << endl;

cout << "5. Отсортировать список" << endl;

cout << "6. Провести поиск по выбранному параметру" << endl;

cout << "7. Средний балл согласно интервалу" << endl;

cout << "0. Выйти из меню" << endl;

act = \_getch(); //Ввод символа в переменную без вывода этого процесса на консоль

switch (act)

{

case '1':

system("cls");

Print\_List(num, List);

system("pause"); //Для приостановки работы программы и при необходимости продолжить нужно будет нажать любую кнопку

break;

case '2':

system("cls");

AddNew(List, num, source);

system("pause");

break;

case '3':

system("cls");

ChangeInfo(List, num, source);

system("pause");

break;

case '4':

system("cls");

Delete(List, num, source);

system("pause");

break;

case '5':

system("cls");

Sort(List, num);

WorkingWithFile(num, List, source); //Чтобы информация вернулась в исходное положение после сортировки

system("pause");

break;

case '6':

system("cls");

Search(List, num);

system("pause");

break;

case '7':

system("cls");

Interval(List, num);

system("pause");

break;

case '0':

is\_working = false;

break;

default:

system("cls");

cout << "Выбран неверный номер действия!" << endl;

system("pause");

break;

}

} while (is\_working);

}

**Shcolniki.h**

#pragma once

struct School

{

char FIO[50] = "-";

char pol[10] = "-";

char class\_name[5] = "-";

char telephone[20] = "-";

double fisic = 0.0;

double math = 0.0;

double rus = 0.0;

double lit = 0.0;

double rat = 0.0;

};

**AddTakeOut.h**

#pragma once

#include "Shcolniki.h"

void Print\_List(int, School\*, int = 0, int = 0); //зададим i по умолчанию, понадобится для функции поиска, а CoutNum нужно будет, чтобы вывод шапки не повторялся

void WorkingWithFile(int&, School\*&, char\*);

void AddNew(School\*&, int&, char\*);

void Rewrite(School\*, int, char\*); //Перезапись информации в файл

void Interval(School\*, int);

void ForSortAfterChange(School\*, int, const char[]);

**AddTakeOut.cpp**

#include <iostream>

#include <fstream>

#include <windows.h>

#include <iomanip>

#include <conio.h>

#include "AddTakeOut.h"

#include "Shcolniki.h"

#include "Sort.h"

using namespace std;

//Функция вывода информации на экран

void Print\_List(int num, School\* List, int i, int CoutNum)

{

if (CoutNum == 0)

{

cout << "|" << setw(4) << "Н-ер" << "| " << setw(27) << "Фамилия Имя Отчество" << " | ";

cout << setw(7) << "Пол" << " | " << setw(5) << "Класс" << " | " << setw(13) << "Телефон" << " | ";

cout << setw(7) << "Физ" << " | " << setw(7) << "Матем" << " | " << setw(7) << "Рус.яз" << " | " << setw(7) << "Рус.лит" << " | ";

cout << setw(7) << "Ср.балл" << " |" << endl;

}

for (i; i < num; i++)

{

cout << "| " << setw(2) << i+1 << " | ";

cout << setw(27) << List[i].FIO << " | ";

cout << setw(7) << List[i].pol << " | ";

cout << setw(5) << List[i].class\_name << " | ";

cout << setw(13) << List[i].telephone << " | ";

cout << setw(7) << List[i].fisic << " | ";

cout << setw(7) << List[i].math << " | ";

cout << setw(7) << List[i].rus << " | ";

cout << setw(7) << List[i].lit << " | ";

cout << setw(7) << List[i].rat << " |" << endl;

}

}

//Функция чтения из файла

void WorkingWithFile(int &num, School\* &List, char\* source)

{

ifstream file;

file.open(source, ios::binary);

num = 0;

file.read((char\*)&num, sizeof(int));

delete[] List;

List = new School[num];

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

file.read((char\*)&List[i], sizeof(School));

file.close();

}

//Функция добавления новой записи

void AddNew(School\* &List, int &num, char\* source)

{

cout << "Текущий список: " << endl;

Print\_List(num, List);

int kuda = 0;

cout << "Укажите номер записи, после которой создать новую: ";

cin >> kuda;

cin.ignore();

system("cls");

if (kuda < 0 || kuda > num)

cout << "Ошибка: такого номера нет" << endl;

else

{

num++;

School\* Buffer = new School[num];

for (int i = 0, j = 0; i < num - 1 && j < num; i++, j++)

{

if (kuda != 0) //Если пользователь хочет добавить запись в самое начало, значит не нужно заполнять начальную запись другим

Buffer[j] = List[i];

else

Buffer[j + 1] = List[i]; //Записываем "утерянный" 0 элемент (ведь в следующей итерации i будет уже равно 1)

if (kuda - 1 == j || kuda == 0)

{

cout << "Информация о новом студенте:" << endl;

cout << "ФИО: ";

cin.getline(Buffer[kuda].FIO, 50);

cout << "Пол: ";

cin.getline(Buffer[kuda].pol, 10);

cout << "Класс: ";

cin.getline(Buffer[kuda].class\_name, 5);

cout << "Номер телефона: ";

cin.getline(Buffer[kuda].telephone, 20);

cout << "Балл по физике: ";

cin >> Buffer[kuda].fisic;

cout << "Балл по математике: ";

cin >> Buffer[kuda].math;

cout << "Балл по русскому языку: ";

cin >> Buffer[kuda].rus;

cout << "Балл по русской литературе: ";

cin >> Buffer[kuda].lit;

cin.ignore(); //Чтобы избежать ошибок при повторном добавлении

if (kuda == 0)

kuda = -1;//Чтобы постоянно не срабатывало условие добавления новой записи

j++; //Чтобы не перезаписать этот элемент в следующее итерации

}

}

delete[] List;

List = Buffer;

Rewrite(List, num, source);

}

}

//Функция перезаписи в файл

void Rewrite(School\* List, int num, char\* source)

{

ofstream file;

file.open("Repository.txt");

if (file.is\_open())

{

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

{

if (i == 0)

file << List[i].FIO << " ";

else

file << "\n" << List[i].FIO << " ";

file << List[i].pol << " ";

file << List[i].class\_name << " ";

file << List[i].telephone << " ";

file << List[i].fisic << " ";

file << List[i].math << " ";

file << List[i].rus << " ";

file << List[i].lit << " ";

List[i].rat = (List[i].fisic + List[i].math + List[i].rus + List[i].lit) / 4.0;

file << List[i].rat;

}

file.close();

}

else

cout << "Ошибка открытия файла!" << endl;

file.open(source, ios::binary);

file.write((char\*)&num, sizeof(int));

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

file.write((char\*)&List[i], sizeof(School));

file.close();

}

//Функция вывода информации согласно интервалу

void Interval(School\* List, int num)

{

double FirstPoint = 0.0, SecondPoint = 0.0; //Определяет первую и вторую точку интервала

int CoutNum = 0;//Подсчитывает не только количество выводов, но исколько всего записей удовлетворяют данному интервалу

cout << "Определите нужный интервал среднего балла" << endl;

cout << "Введите начальное значение в интервале: ";

cin >> FirstPoint;

cout << "Введите крайнее значение в интервале: ";

cin >> SecondPoint;

cin.ignore();

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

{

if (List[i].rat >= FirstPoint && List[i].rat <= SecondPoint)

{

Print\_List(i + 1, List, i, CoutNum);

CoutNum++;

}

}

cout << "Общее количество записей, удовлетворяющих данному интервалу: " << CoutNum << endl;

}

//Функция изменения индексных бинарных файлов после изменения информации в базе данных

void ForSortAfterChange(School\* List, int num, const char IndexFile[]) //Заполняет индексный бинарный файл неверной информацией после изменения, чтобы функция сортировки не брала оттуда индексы

{

ofstream file;

file.open(IndexFile, ios::binary);

num--;

file.write((char\*)&num, sizeof(int));

file.close();

}

**WorkWithStruct. h**

#pragma once

#include "Shcolniki.h"

void Search(School\*, int&, int = 1);

void Delete(School\*&, int&, char\*);

void ChangeInfo(School\*, int, char\*);

**WorkWithStruct. Cpp**

#include <iostream>

#include <fstream>

#include <windows.h>

#include <iomanip>

#include <conio.h>

#include "WorkWithStruct.h"

#include "Shcolniki.h"

#include "AddTakeOut.h"

#include "Sort.h"

using namespace std;

//Функция поиска

void Search(School\* List, int &num, int mode) //mode = 1 это поиск с последущим выводом на экран, mode = 2 нужен исключительно для удаления

{

char Searching[50] = { '\0' };

int SearchRate = 0;

bool Similarity = true; //Проверяет схожесть введёной информации с уже имеющейся в структуре

int CoutNum = 0; //Подсчитывает количество результатов поиска

cout << "Укажите номер нужного поля: " << endl;

cout << "1. ФИО\n2. Пол\n3. Класс\n4. Номер телефона\n5. Оценки по физике\n6. Математике\n7. Русскому\n8. Литературе" << endl;

char SearchAct = \_getch();

system("cls");

cout << "Введите информацию: " << endl;

switch (SearchAct)

{

case '1':

cin.getline(Searching, 20);

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

{

Similarity = true;

int j = 0;

while (Similarity == true && j != 20)

{

if (List[i].FIO[j] != Searching[j])

Similarity = false;

j++;

}

if (Similarity == true)

{

if (mode == 1)

{

Print\_List(i + 1, List, i, CoutNum);

CoutNum++;

}

else if (mode == 2)

{

for (j = i; j < num - 1; j++)

swap(List[j], List[j + 1]);

num--;

i--; //Для того, чтобы не перескочить через нужный элемент, при условии, что несколько искомых элементов идут друг за другом

}

}

}

break;

case '2':

cin.getline(Searching, 10);

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

{

Similarity = true;

int j = 0;

while (Similarity == true && j != 10)

{

if (List[i].pol[j] != Searching[j])

Similarity = false;

j++;

}

if (Similarity == true)

{

if (mode == 1)

{

Print\_List(i + 1, List, i, CoutNum);

CoutNum++;

}

else if (mode == 2)

{

for (int j = i; j < num - 1; j++)

swap(List[j], List[j + 1]);

num--;

i--;

}

}

}

break;

case '3':

cin.getline(Searching, 5);

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

{

Similarity = true;

int j = 0;

while (Similarity == true && j != 5)

{

if (List[i].class\_name[j] != Searching[j])

Similarity = false;

j++;

}

if (Similarity == true)

{

if (mode == 1)

{

Print\_List(i + 1, List, i, CoutNum);

CoutNum++;

}

else if (mode == 2)

{

for (int j = i; j < num - 1; j++)

swap(List[j], List[j + 1]);

num--;

i--;

}

}

}

break;

case '4':

cin.getline(Searching, 20);

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

{

Similarity = true;

int j = 0;

while (Similarity == true && j != 20)

{

if (List[i].telephone[j] != Searching[j])

Similarity = false;

j++;

}

if (Similarity == true)

{

if (mode == 1)

{

Print\_List(i + 1, List, i, CoutNum);

CoutNum++;

}

else if (mode == 2)

{

for (int j = i; j < num - 1; j++)

swap(List[j], List[j + 1]);

num--;

i--;

}

}

}

break;

case '5':

cin >> SearchRate;

cin.ignore();

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

{

if (List[i].fisic == SearchRate)

{

if (mode == 1)

{

Print\_List(i + 1, List, i, CoutNum);

CoutNum++;

}

else if (mode == 2)

{

for (int j = i; j < num - 1; j++)

swap(List[j], List[j + 1]);

num--;

i--;

}

}

}

break;

case '6':

cin >> SearchRate;

cin.ignore();

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

{

if (List[i].math == SearchRate)

{

if (mode == 1)

{

Print\_List(i + 1, List, i, CoutNum);

CoutNum++;

}

else if (mode == 2)

{

for (int j = i; j < num - 1; j++)

swap(List[j], List[j + 1]);

num--;

i--;

}

}

}

break;

case '7':

cin >> SearchRate;

cin.ignore();

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

{

if (List[i].rus == SearchRate)

{

if (mode == 1)

{

Print\_List(i + 1, List, i, CoutNum);

CoutNum++;

}

else if (mode == 2)

{

for (int j = i; j < num - 1; j++)

swap(List[j], List[j + 1]);

num--;

i--;

}

}

}

break;

case '8':

cin >> SearchRate;

cin.ignore();

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

{

if (List[i].lit == SearchRate)

{

if (mode == 1)

{

Print\_List(i + 1, List, i, CoutNum);

CoutNum++;

}

else if (mode == 2)

{

for (int j = i; j < num - 1; j++)

swap(List[j], List[j + 1]);

num--;

i--;

}

}

}

break;

default:

system("cls");

cout << "Выбран неверный номер поля!" << endl;

break;

}

if (mode == 1)

{

if (CoutNum == 0)

cout << "По вашему запросу ничего не найдено..." << endl;

}

}

//Функция удаления

void Delete(School\* &List, int &num, char\* source)

{

char ActDelete;

cout << "Удалить ученика по значению поля или по номеру?" << endl;

cout << "1. Ввести номер ученика, которого необходимо удалить" << endl;

cout << "2. Удалить по полю" << endl;

ActDelete = \_getch();

int DeleteID = 0; //Номер записи для удаления

system("cls");

if (ActDelete == '1' || ActDelete == '2')

Print\_List(num, List);

School\* Buffer = new School[num];

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

Buffer[i] = List[i];

delete[] List;

switch (ActDelete)

{

case '1':

cout << "Введите номер ученика, информацию о котором вы хотите удалить: ";

cin >> DeleteID;

cin.ignore(); //Чтобы избежать ошибок в других пунктах меню

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

{

if (i == DeleteID - 1)

{

for (int j = i; j < num - 1; j++)

swap(Buffer[j], Buffer[j + 1]);

num--;

}

}

List = new School[num];

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

List[i] = Buffer[i];

break;

case '2':

Search(Buffer, num, 2);

List = new School[num];

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

List[i] = Buffer[i];

break;

default:

cout << "Выбран несуществующий номер действия!" << endl;

List = new School[num];

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

List[i] = Buffer[i];

break;

}

delete[] Buffer;

Rewrite(List, num, source);

}

//Функция изменения информации

void ChangeInfo(School\* List, int num, char\* source)

{

cout << "Вы хотите изменить всю запись или только конкретное поле?" << endl;

cout << "1. Полностью всю запись\n2. Только конкретное поле" << endl;

char Change = \_getch();

int ChangeInfo = 0;

char WhatChange; //Нужно будет для изменения конкретного поля

system("cls");

switch (Change)

{

case '1':

cout << "Текущий список: " << endl;

Print\_List(num, List);

cout << "Введите номер строки, которую хотите изменить: ";

cin >> ChangeInfo;

cin.ignore();

if (ChangeInfo < 0 || ChangeInfo > num)

{

system("cls");

cout << "Ошибка: такой строки нет" << endl;

break;

}

system("cls");

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

{

if (i == ChangeInfo - 1)

{

cout << "Измённая информация о данном ученике:" << endl;

cout << "ФИО: ";

cin.getline(List[i].FIO, 50);

cout << "Пол: ";

cin.getline(List[i].pol, 10);

cout << "Класс: ";

cin.getline(List[i].class\_name, 5);

cout << "Номер телефона: ";

cin.getline(List[i].telephone, 20);

cout << "Балл по физике: ";

cin >> List[i].fisic;

cout << "Балл по математике: ";

cin >> List[i].math;

cout << "Балл по русскому языку: ";

cin >> List[i].rus;

cout << "Балл по русской литературе: ";

cin >> List[i].lit;

cin.ignore();

}

}

ForSortAfterChange(List, num, "Sort1.txt");

ForSortAfterChange(List, num, "Sort2.txt");

ForSortAfterChange(List, num, "Sort3.txt");

ForSortAfterChange(List, num, "Sort4.txt");

ForSortAfterChange(List, num, "Sort51.txt");

ForSortAfterChange(List, num, "Sort52.txt");

ForSortAfterChange(List, num, "Sort53.txt");

ForSortAfterChange(List, num, "Sort54.txt");

ForSortAfterChange(List, num, "Sort55.txt");

break;

case '2':

cout << "Текущий список: " << endl;

Print\_List(num, List);

cout << "Сначала выберете номер студента, часть информации о котором нужно изменить: ";

cin >> ChangeInfo;

cin.ignore();

if (ChangeInfo < 0 || ChangeInfo > num)

{

cout << "Ошибка: такой строки нет" << endl;

break;

}

system("cls");

cout << "Что именно вы хотите изменить?" << endl;

cout << "1. ФИО\n2. Пол\n3. Класс\n4. Номер телефона\n5. Оценки" << endl;

WhatChange = \_getch();

system("cls");

switch (WhatChange)

{

case '1':

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

{

if (i == ChangeInfo - 1)

{

cout << "Введите новую информацию об ученике" << endl;

cout << "ФИО: ";

cin.getline(List[i].FIO, 50);

}

}

ForSortAfterChange(List, num, "Sort1.txt");

break;

case '2':

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

{

if (i == ChangeInfo - 1)

{

cout << "Введите новую информацию об ученике" << endl;

cout << "Пол: ";

cin.getline(List[i].pol, 10);

}

}

ForSortAfterChange(List, num, "Sort2.txt");

break;

case '3':

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

{

if (i == ChangeInfo - 1)

{

cout << "Введите новую информацию об ученике" << endl;

cout << "Класс: ";

cin.getline(List[i].class\_name, 5);

}

}

ForSortAfterChange(List, num, "Sort3.txt");

break;

case '4':

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

{

if (i == ChangeInfo - 1)

{

cout << "Введите новую информацию об ученике" << endl;

cout << "Номер телефона: ";

cin.getline(List[i].telephone, 20);

}

}

ForSortAfterChange(List, num, "Sort4.txt");

break;

case '5':

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

{

if (i == ChangeInfo - 1)

{

cout << "Введите новую информацию об ученике" << endl;

cout << "Оценка по физике: ";

cin >> List[i].fisic;

cout << "Оценка по математике: ";

cin >> List[i].math;

cout << "Оценка по русскому: ";

cin >> List[i].rus;

cout << "Оценка по литературе: ";

cin >> List[i].lit;

cin.ignore();

}

}

ForSortAfterChange(List, num, "Sort51.txt");

ForSortAfterChange(List, num, "Sort52.txt");

ForSortAfterChange(List, num, "Sort53.txt");

ForSortAfterChange(List, num, "Sort54.txt");

ForSortAfterChange(List, num, "Sort55.txt");

break;

default:

cout << "Выбран неверный номер действия!" << endl;

break;

}

break;

default:

cout << "Выбран неверный номер действия!" << endl;

break;

}

Rewrite(List, num, source);

}

**Sort.h**

#pragma once

#include "Shcolniki.h"

void Sort(School\*, int, int = 0);

**Sort.cpp**

#include <iostream>

#include <fstream>

#include <windows.h>

#include <iomanip>

#include <conio.h>

#include "WorkWithStruct.h"

#include "Shcolniki.h"

#include "AddTakeOut.h"

#include "Sort.h"

using namespace std;

//Функция сортировки

void Sort(School\* List, int num, int mode) //Здесь тоже 2 режима, один сортирует в индексный бинарный файл, а второй просто выводит информацию из готового индексного файла

{

int ind = 0; // ind - количество индексов в файле

int SortAct = 0;

fstream file; //Для записи и чтения одновременно

int\* Index = new int[num]; //Создаём массив с индексами (для индексного файла)

for (int i = 0; i < num; i++) //Заполняем его индексами

Index[i] = i + 1;

Print\_List(num, List);

cout << "По какому полю сортировать?" << endl;

cout << "1. По ФИО\n2. По полу\n3. По классу\n4. По номеру телефона\n5. По оценкам" << endl;

SortAct = \_getch();

int i, j, m;

switch (SortAct)

{

case '1':

system("cls");

file.open("Sort1.txt", ios::binary);

file.read((char\*)&ind, sizeof(int));

if (ind == num)

mode = 2;

else

mode = 1;

if (mode == 1)

{

//Используем сортировку выбором

for (i = 0; i < num; i++)

{

for (j = i, m = i; j < num; j++)

{

if (strcmp(List[j].FIO, List[m].FIO) < 0)

m = j;

}

swap(List[i], List[m]);

swap(Index[i], Index[m]);

}

Print\_List(num, List);

}

if (mode == 2)

{

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

{

file.read((char\*)&Index[i], sizeof(int));

Print\_List(Index[i] + 1, List, Index[i], i);

}

}

break;

case '2':

system("cls");

file.open("Sort2.txt", ios::binary);

file.read((char\*)&ind, sizeof(int));

if (ind == num)

mode = 2;

else

mode = 1;

if (mode == 1)

{

for (i = 0; i < num; i++)

{

for (j = i, m = i; j < num; j++)

{

if (strcmp(List[j].pol, List[m].pol) > 0)

m = j;

}

swap(List[i], List[m]);

swap(Index[i], Index[m]);

}

Print\_List(num, List);

}

if (mode == 2)

{

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

{

file.read((char\*)&Index[i], sizeof(int));

Print\_List(Index[i] + 1, List, Index[i], i);

}

}

break;

case '3':

system("cls");

file.open("Sort3.txt", ios::binary);

file.read((char\*)&ind, sizeof(int));

if (ind == num)

mode = 2;

else

mode = 1;

if (mode == 1)

{

for (i = 0; i < num; i++)

{

for (j = i, m = i; j < num; j++)

{

if (strcmp(List[j].class\_name, List[m].class\_name) < 0)

m = j;

}

swap(List[i], List[m]);

swap(Index[i], Index[m]);

}

Print\_List(num, List);

}

if (mode == 2)

{

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

{

file.read((char\*)&Index[i], sizeof(int));

Print\_List(Index[i] + 1, List, Index[i], i);

}

}

break;

case '4':

system("cls");

file.open("Sort4.txt", ios::binary);

file.read((char\*)&ind, sizeof(int));

if (ind == num)

mode = 2;

else

mode = 1;

if (mode == 1)

{

for (i = 0; i < num; i++)

{

for (j = i, m = i; j < num; j++)

{

if (strcmp(List[j].telephone, List[m].telephone) < 0)

m = j;

}

swap(List[i], List[m]);

swap(Index[i], Index[m]);

}

Print\_List(num, List);

}

if (mode == 2)

{

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

{

file.read((char\*)&Index[i], sizeof(int));

Print\_List(Index[i] + 1, List, Index[i], i);

}

}

break;

case '5':

system("cls");

cout << "1. Физика\n2. Математика\n3. Русский язык\n4. Русская литература\n5. Средний балл" << endl;

SortAct = '\0';

SortAct = \_getch();

system("cls");

switch (SortAct)

{

case '1':

file.open("Sort51.txt", ios::binary);

file.read((char\*)&ind, sizeof(int));

if (ind == num)

mode = 2;

else

mode = 1;

if (mode == 1)

{

for (i = 0; i < num; i++)

{

for (j = i, m = i; j < num; j++)

{

if (List[j].fisic > List[m].fisic)

m = j;

}

swap(List[i], List[m]);

swap(Index[i], Index[m]);

}

Print\_List(num, List);

}

if (mode == 2)

{

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

{

file.read((char\*)&Index[i], sizeof(int));

Print\_List(Index[i] + 1, List, Index[i], i);

}

}

break;

case '2':

file.open("Sort52.txt", ios::binary);

file.read((char\*)&ind, sizeof(int));

if (ind == num)

mode = 2;

else

mode = 1;

if (mode == 1)

{

for (i = 0; i < num; i++)

{

for (j = i, m = i; j < num; j++)

{

if (List[j].math > List[m].math)

m = j;

}

swap(List[i], List[m]);

swap(Index[i], Index[m]);

}

Print\_List(num, List);

}

if (mode == 2)

{

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

{

file.read((char\*)&Index[i], sizeof(int));

Print\_List(Index[i] + 1, List, Index[i], i);

}

}

break;

case '3':

file.open("Sort53.txt", ios::binary);

file.read((char\*)&ind, sizeof(int));

if (ind == num)

mode = 2;

else

mode = 1;

if (mode == 1)

{

for (i = 0; i < num; i++)

{

for (j = i, m = i; j < num; j++)

{

if (List[j].rus > List[m].rus)

m = j;

}

swap(List[i], List[m]);

swap(Index[i], Index[m]);

}

Print\_List(num, List);

}

if (mode == 2)

{

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

{

file.read((char\*)&Index[i], sizeof(int));

Print\_List(Index[i] + 1, List, Index[i], i);

}

}

break;

case '4':

file.open("Sort54.txt", ios::binary);

file.read((char\*)&ind, sizeof(int));

if (ind == num)

mode = 2;

else

mode = 1;

if (mode == 1)

{

for (i = 0; i < num; i++)

{

for (j = i, m = i; j < num; j++)

{

if (List[j].lit > List[m].lit)

m = j;

}

swap(List[i], List[m]);

swap(Index[i], Index[m]);

}

Print\_List(num, List);

}

if (mode == 2)

{

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

{

file.read((char\*)&Index[i], sizeof(int));

Print\_List(Index[i] + 1, List, Index[i], i);

}

}

break;

case '5':

file.open("Sort55.txt", ios::binary);

file.read((char\*)&ind, sizeof(int));

if (ind == num)

mode = 2;

else

mode = 1;

if (mode == 1)

{

for (i = 0; i < num; i++)

{

for (j = i, m = i; j < num; j++)

{

if (List[j].rat > List[m].rat)

m = j;

}

swap(List[i], List[m]);

swap(Index[i], Index[m]);

}

Print\_List(num, List);

}

if (mode == 2)

{

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

{

file.read((char\*)&Index[i], sizeof(int));

Print\_List(Index[i] + 1, List, Index[i], i);

}

}

break;

default:

cout << "Выбран неверный номер действия!" << endl;

break;

}

break;

default:

cout << "Выбран неверный номер действия!" << endl;

break;

}

if (mode == 1)

{

file.write((char\*)& num, sizeof(int));

for (int i = 0; i < num; i++) //Записываем индексы в созданный бинарный файл

file.write((char\*)&Index[i], sizeof(int));

}

file.close();

delete[] Index;

}