Содержание

[1 Требования к программе 2](#_Toc69058705)

[1.1 Задание 2](#_Toc69058706)

[1.2 Исходные данные 2](#_Toc69058707)

[1.3 Функциональные требования к программе 3](#_Toc69058708)

[1.4 Требования к программной реализации 4](#_Toc69058709)

[2 Конструирование программы 6](#_Toc69058710)

[2.1 Разработка модульной структуры программы 6](#_Toc69058711)

[2.2 Выбор способа организации данных 6](#_Toc69058712)

[2.3 Разработка перечня пользовательских функций программы 7](#_Toc69058712)

[3 Разработка алгоритмов программы 10](#_Toc69058713)

[3.1 Алгоритм функции main 10](#_Toc69058714)

[3.2 Алгоритм функции bool otlichnik(Student a); 10](#_Toc69058715)

[3.3 Алгоритм функции void defaultBenefitChange(vector <Student>& v); 11](#_Toc69058716)

[4 Описание работы программы 12](#_Toc69058713)

[4.1 Авторизация 12](#_Toc69058714)

[4.2 Модуль администратора 12](#_Toc69058715)

[4.3 Модуль пользователя 15](#_Toc69058716)

[4.4 Исключительные ситуации 16](#_Toc69058714)

[ПРИЛОЖЕНИЕ 17](#_Toc69058717)

1 ТРЕБОВАНИЯ К ПРОГРАММЕ

1.1 Задание

Разработка программы расчета стипендии.

Для расчета стипендии необходимо обработать информацию следующего вида: номер группы, Ф. И. О. студента, форма обучения (платная/бюджетная), зачеты по пяти предметам (зачет/незачет), оценки по 4 предметам, признак участия в общественной работе: 1 – активное участие, 0 – неучастие.

Индивидуальное задание: рассчитать стипендию для студентов-бюджетников. При этом: отличники (отметки 9, 10) и общественники получают 50 % надбавку, а просто отличники – 25 %. Студенты со средним баллом, равным или ниже 5, стипендию не получают. Базовый размер стипендии вводится с клавиатуры.

Общее для всех вариантов задание: реализовать авторизацию для входа в систему, функционал администратора и функционал пользователя (см. более подробно в функциональных требованиях к курсовой работе).

1.2 Исходные данные

Тема курсовой работы выбирается из списка, приведенного в

приложении 1.

Язык программирования С++.

Среда разработки Microsoft Visual Studio.

Вид приложения – консольное.

Парадигма программирования – процедурная (по согласованию с преподавателем допускается реализация программы в рамках объектно-ориентированной парадигмы программирования).

Способ организации данных – структуры (struct) (либо поля классов в случае объектно-ориентированного программирования).

Способ хранения данных – файлы (по согласованию с преподавателем допускается подключение баз данных).

Каждая логически завершенная подзадача программы должна быть реализована в виде отдельной функции (метода в случае объектно-ориентированного программирования).

Построение программного кода должно соответствовать соглашению о коде «С++ Code Convention».

К защите курсовой работы представляются: консольное приложение и пояснительная записка.

Текст пояснительной записки оформляется в соответствии со стандартом предприятия СТП 01–2017.

1.3 Функциональные требования к программе

Данные аккаунтов и студентов хранятся в отдельных файлах в форме соответствующих структур.

При каждом вводе пароля, он маскируется звездочками.

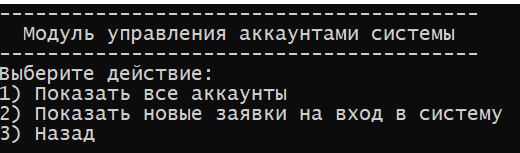
Алгоритм регистрации устроен через подтверждение заявки на создание нового аккаунта в системе администратором. В момент оформления новой заявки запрашиваемый логин проверяется на уникальность и, в случае наличия аккаунта с таким логином в системе, выдает пользователю сообщение и просит ввести новый.

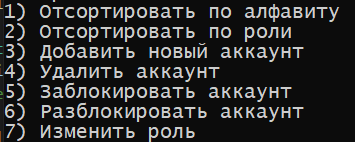
Все пароли хешируются через md5 и в файле хранится только хеш соответствующего пароля.

Работа с данными:

Модуль администратора включает два подмодуля (работа с учетными записями и данными самой системы).

В первом реализованы следующие функции (рис. 1.1):





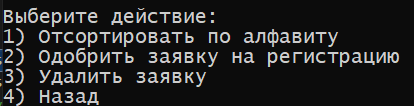
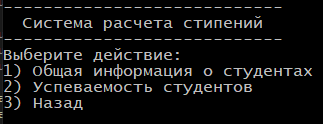


Рисунок 1.1

Во втором реализованы следующие функции (рис. 1.2):



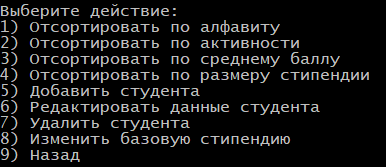


Рисунок 1.2

В модуле пользователя реализованы только функции системы расчета стипендий, кроме добавления/удаления/редактирования данных студентов

В работе также предусмотрена обработка основных исключительных ситуаций, навигация, запрос на выполнение необратимых действий, обратная связь с пользователем.

1.4 Требования к программной реализации

1. Все переменные и константы должны иметь осмысленные имена в рамках тематики варианта курсовой работы. Переменным рекомендуется присваивать имена, состоящие из букв нижнего регистра; для формирования составного имени используется нижнее подчеркивание (например, number\_of\_students) или «верблюжья нотация» (например, flagExit). Константам рекомендуется присваивать имена, состоящие из букв верхнего регистра (например, SIZE\_ARR\_OF\_\_ACCOUNTS, FILE\_OF\_ACCOUNTS).

2. Имена функций должны быть осмысленными, начинаться с буквы нижнего регистра, строится по принципу глагол+существительное (например,

addAccount, findStudentBySurname). Если функция выполняет проверку и возвращает результат типа bool, то ее название должно начинаться с глагола is (например, isNumberNumeric, isLoginUnique).

3. Не допускается использование оператора прерывания goto.

4. Код не должен содержать неименованных числовых констант («магических» чисел), неименованных строковых констант (например, имен файлов и др.). Подобного рода информацию следует представлять как глобальные константы. По правилам качественного стиля программирования тексты всех информационных сообщений, выводимых пользователю в ответ на его действия, также оформляются как константы.

5. Код необходимо комментировать (как минимум в части объявления структур, массивов/векторов, прототипов функций, нетривиальной логики).

6. Код не должен дублироваться – для этого существуют функции!.

7. Одна функция решает только одну задачу (например, не допускается в одной функции считывать данные из файла и выводить их на консоль – это две.

разные функции). При этом внутри функции возможен вызов других функций.

8. Выполнение операций чтения/записи в файл должно быть сведено к минимуму (т. е. после однократной выгрузки данных из файла в массив/вектор дальнейшая работа ведется с этим массивом/вектором, а не происходит многократное считывание данных из файла в каждой функции).

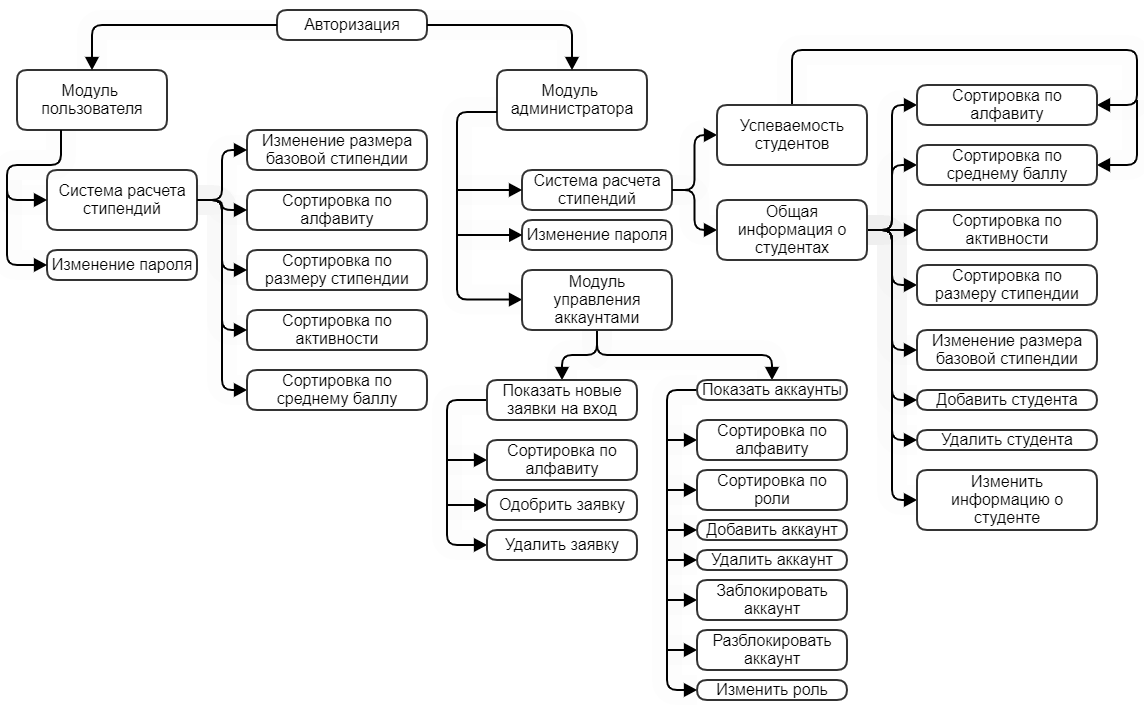
9. Следует избегать глубокой вложенности условных и циклических конструкций: вложенность блоков должна быть не более трѐх.

10. Следует избегать длинных функций: текст функции должен умещаться на один экран (в эквиваленте строк – не превышать 25-50 строчек).

11. Следует выносить код логически независимых модулей в отдельные .cpp файлы и подключать их с помощью заголовочных .h файлов.

2 КОНСТРУИРОВАНИЕ ПРОГРАММЫ

2.1 Разработка модульной структуры программы



2.2 Выбор способа организации данных

// Структура для аккаунта.

struct Account.

{

string login; // Логин аккаунта.

string password; // Пароль хешированный через md5.

int role; // Роль, где 1 – admin, 2 – user.

int access; // Доступ. Аккаунты без доступа к системе имеют значение – 0 (присваивается после заявки на регистрацию нового аккаунта), с доступом – 1.

int block; // Блокировка. Заблокированные аккаунты имеют значение – 1 в этом поле, остальные – 0 (по умолчанию).

};

// Стурктура для студента.

struct Student.

{

string surname; // Фамилия.

int group\_number; // Номер группы.

int active; // Показатель активности (1 – активен, 0 – неактивен).

double benefits; // Размер стипендии.

Marks marks; // Отдельная структура для фиксации оценок по каждому предмету.

};

Способ объединения входных данных – вектор.

2.3 Разработка перечня пользовательских функций программы

validation.h.

// Ввод чисел.

int inputNumber (int left\_range, int right\_range);

// Проверка на число.

bool isNumberNumeric ();

// Проверка на ввод числа из корректного промежутка.

bool isNumberRangeCorrect (int number, int left\_range, int right\_range);

md5.h.

// Хеширование (используется для пароля).

string md5 (string str);

interfaces.h.

// Меню пользователя.

void UserInterface (vector <Student>& vec\_of\_students, vector <Account>& vec\_of\_accounts, Account& current\_account, bool& similar);

// Меню администратора.

void AdminInterface (vector <Student>& vec\_of\_students, vector <Account>& vec\_of\_accounts, Account& current\_account, bool& similar);

// Запуск системы.

void System (vector <Student>& vec\_of\_students, vector <Account>& vec\_of\_accounts);

accs.h.

// Изменение пароля пользователем.

void userPassChange (vector <Account>& v, Account& current\_acc);

// Ввод пароля с маскировкой (возвращает пароль).

string inputPass ();

// Удаление новой заявкм.

void deleteApplication (vector <Account>& v, int counter);

// Поиск индекса заявки.

int searchApplicationIndex (vector <Account>& v, int counter);

// Подтверждение заявки.

void ApproveApplication (vector <Account>& v, int counter);

// Отображение заявок.

void showNewApplications (vector <Account>& v);

// Удалить аккаунт.

void deleteAccount (vector <Account>& v, int counter, Account current\_acc);

// Изменить роль аккаунта.

void roleChange (vector <Account>& v, int counter);

// Поиск индекса нужного аккаунта.

int searchAccIndex (vector <Account>& v, int counter);

// Блокировка аккаунта.

void blockAccount (vector <Account>& v, int counter);

// Разблокировка аккаунта.

void unblockAccount (vector <Account>& v, int counter);

// Отображение всех аккаунтов.

void showAccounts (vector <Account>& v, Account current\_acc);

// Создание нового аккаунта администратором.

void newAccountCreation (vector <Account>& v);

// Запись всех аккаунтов в файл.

void writeFileAccounts (vector <Account>& v);

// Создание нового акккаунта.

void newAccountApplication (vector <Account>& v);

// Подсчет кол-ва строк в файле.

int getCountOfStucturesInFile (string file\_path);

//Чтение аккаунтов из файла.

void readFileAccounts (vector <Account>& v);

// Сортировки аккаунтов.

// Сортировка по алфавиту (по полю с логинами).

bool mySortByLogin (Account a, Account b);

void sortByLogin (vector <Account>& v);

// Сортировка по роли (администраторы > пользователи).

bool mySortByRole (Account a, Account b);

void sortByRole (vector <Account>& v);

students.h.

// Отображение списка студентов с основной информацией.

void showStudents (vector <Student>& v);

// Чтение информации о студентах из файла в вектор.

void readFileStudents (vector <Student>& v);

// Запись вектора со студентами в файл.

void writeFileStudents (vector <Student>& v);

// Отображение списка студентов с их успеваемостью.

void showStudentsPerfomance (vector <Student>& v);

// Удаление конкретного студента.

void deleteStudent (vector <Student>& v);

// Флаг для отличников.

bool otlichnik (Student a);

// Изменение размера базовой стипендии.

void defaultBenefitChange (vector <Student>& v);

// Добавление нового студента в систему.

void addStudent (vector <Student>& v);

// Изменение информации о студенте.

void editStudent (vector <Student>& v);

// Возможность изменения информации о только что добаленном студенте.

void editAddedStudent (Student a);

// Пользовательский вариант меню со студентами.

void showStudentsForUser (vector <Student>& v);

// Функции сортировки.

// Сортировка по размеру стипендии.

bool mySortByBenefit (Student a, Student b);

void sortByBenefit (vector <Student>& v);

// Сортировка по активности.

bool mySortByActive (Student a, Student b);

void sortByActive (vector <Student>& v);

// Сортировка по среднему баллу.

bool mySortByAverageMark (Student a, Student b);

void sortByAverageMark (vector <Student>& v);

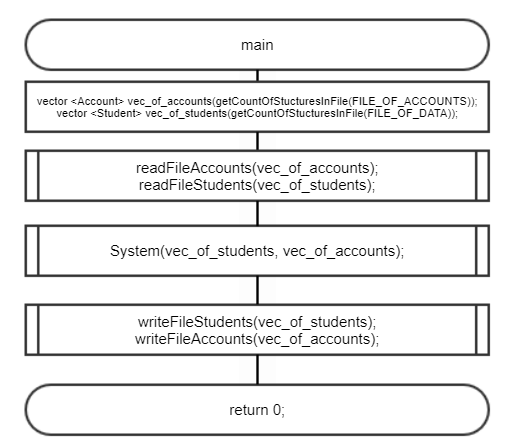
// Сортировка по алфавиту (по полю с фамилиями).

bool mySortBySurname (Student a, Student b);

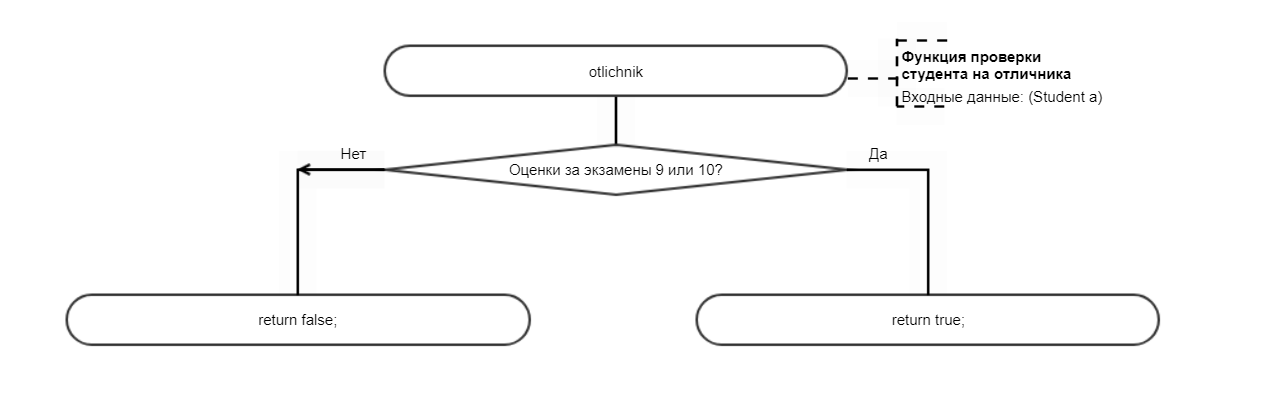
void sortBySurname (vector <Student>& v);

3 РАЗРАБОТКА АЛГОРИТМОВ ПРОГРАММЫ

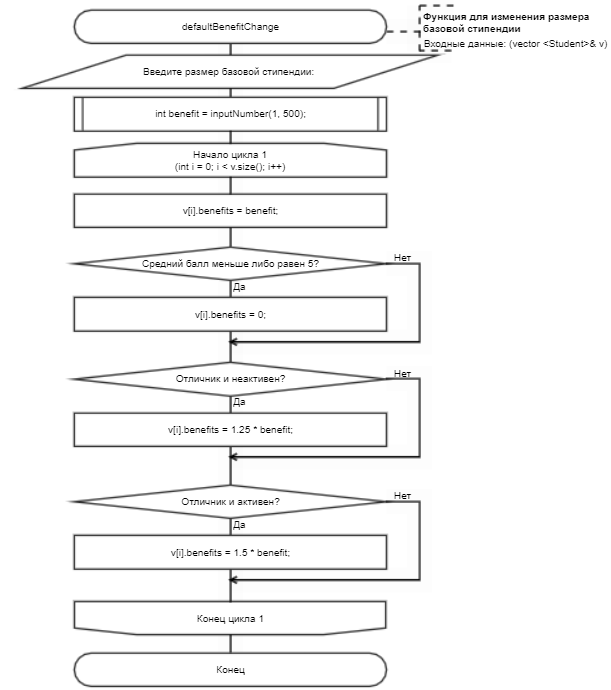
3.1 Алгоритм функции main



3.2 Алгоритм функции bool otlichnik (Student a)



3.3 Алгоритм функции void defaultBenefitChange(vector <Student>& v)



4 ОПИСАНИЕ РАБОТЫ ПРОГРАММЫ

4.1 Авторизация

В начале авторизации программа предлагает пользователю войти, зарегистрировать новый аккаунт или выйти из системы (рис. 4.1).

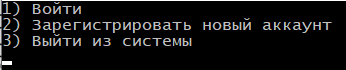


Рисунок 4.1

В случае регистрации будет создана новая учетная запись без доступа к системе. Доступ будет открыт как только администратор одобрит заявку на регистрацию нового аккаунта. Если же пользователь уже имеет учетную запись с доступом к системе он может авторизоваться используя свой логин и пароль. Хеш введенного пароля сверяется с тем, что хранится в файле, и, в случае их совпадения, авторизация завершается успешно.

4.2 Модуль администратора

Модуль администратора включает в себя саму систему расчета стипендий, модуль управления аккаунтами, возможность изменить свой пароль и выйти из аккаунта обратно в меню авторизации (рис. 4.2).

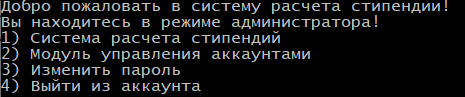


Рисунок 4.2

Все, предложенные администратору функции, показаны в разделе 4.3. Стоит отметить, что работа с данными студентов разбита на 2 подмодуля: модуль с общей информацией и модуль с успеваемостью студентов (рис. 4.3).

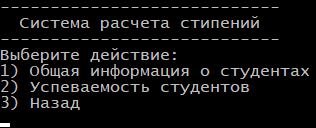


Рисунок 4.3

Сами подмодули реализованы в удобном для просмотра ключе (рис. 4.4 и рис. 4.5).

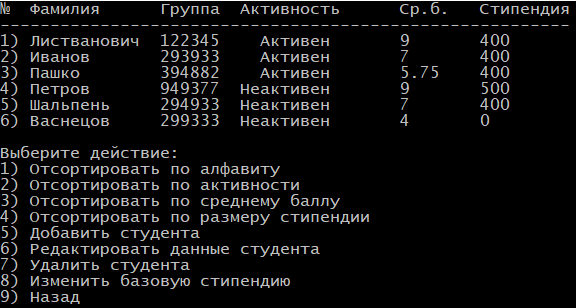


Рисунок 4.4

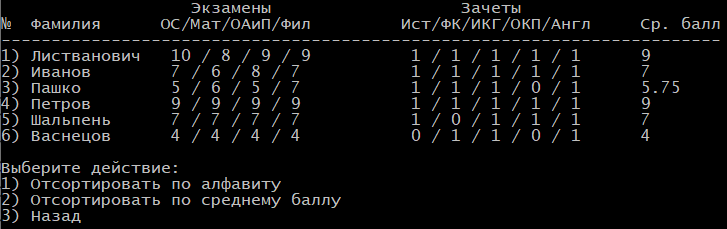


Рисунок 4.5

Модуль управления аккаунтами также разбит на два подмодуля: основной и меню для работы с заявками на регистрацию новых аккаунтов (рис. 4.6). Сами же подмодули, по аналогии с системой, оформлены в удобном для просмотра виде (рис. 4.7 и рис. 4.8).

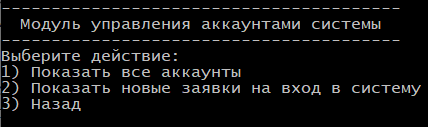


Рисунок 4.6

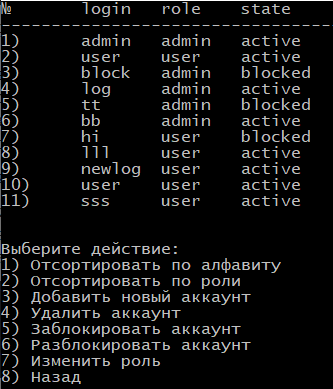


Рисунок 4.7

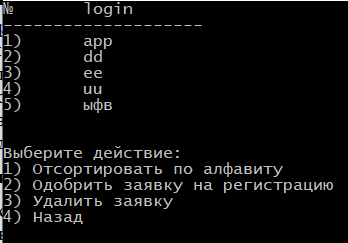


Рисунок 4.8

4.3 Модуль пользователя

В модуле пользователя реализовано изменение пароля и функции системы расчета стипендий, кроме добавления/удаления/редактирования данных студентов (рис. 4.9 и рис. 4.10).

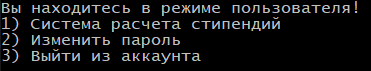


Рисунок 4.9

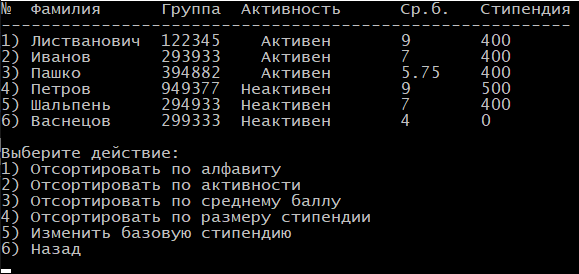


Рисунок 4.10

4.4 Исключительные ситуации

На этапе авторизации предусмотрены ситуации некорректного ввода данных аккаунта, попытка регистрации нового аккаунта с логином, который уже используется.

Корректный ввод числовых данных реализован в каждой функции программы, учитывая все логичные значения.

Для изменения пароля учетной записи нужно ввести свой актуальный пароль, где после трёх неудачных попыток функция будет прервана.

Запрос на выполнение необратимых действий исключает кейсы случайных ошибочных действий.

Предусмотрен случай отсутствия текстовых файлов и вывод соответствующего сообщения на экран.

ПРИЛОЖЕНИЕ

**Source.cpp**

#include <windows.h>

#include <vector>

#include "accs.h"

#include "students.h"

#include "interfaces.h"

int main() {

SetConsoleCP(1251);

SetConsoleOutputCP(1251);

vector <Account> vec\_of\_accounts(getCountOfStucturesInFile(FILE\_OF\_ACCOUNTS));

vector <Student> vec\_of\_students(getCountOfStucturesInFile(FILE\_OF\_DATA));

readFileAccounts(vec\_of\_accounts);

readFileStudents(vec\_of\_students);

System(vec\_of\_students, vec\_of\_accounts);

writeFileStudents(vec\_of\_students);

writeFileAccounts(vec\_of\_accounts);

system("pause");

return 0;

}

interfaces.h

#pragma once

#include <iostream>

#include <string>

#include <fstream>

#include <conio.h>

#include <windows.h>

#include <vector>

#include <algorithm>

#include "validation.h"

#include "accs.h"

#include "students.h"

#include "md5.h"

using namespace std;

// Меню пользователя

void UserInterface(vector <Student>& vec\_of\_students, vector <Account>& vec\_of\_accounts, Account& current\_account, bool& similar);

// Меню администратора

void AdminInterface(vector <Student>& vec\_of\_students, vector <Account>& vec\_of\_accounts, Account& current\_account, bool& similar);

// Запуск системы

void System(vector <Student>& vec\_of\_students, vector <Account>& vec\_of\_accounts);

interfaces.cpp

#include "interfaces.h"

void UserInterface(vector <Student>& vec\_of\_students, vector <Account>& vec\_of\_accounts, Account& current\_account, bool& similar) {

if (current\_account.access == 0) {

cout << endl << "Ваша заявка на регистрацию еще не одобрена администратором!" << endl << endl;

system("pause");

similar = false;

}

else {

cout << "Добро пожаловать в систему расчета стипендии! " << endl;

bool user\_menu\_flag = true;

while (user\_menu\_flag) {

system("cls");

cout << "Вы находитесь в режиме пользователя!" << endl << "1) Система расчета стипендий" << endl << "2) Изменить пароль" << endl << "3) Выйти из аккаунта" << endl;

int user\_menu\_choice = inputNumber(1, 3);

if (user\_menu\_choice == 1) { showStudentsForUser(vec\_of\_students); }

else if (user\_menu\_choice == 3) {

user\_menu\_flag = false;

similar = false;

}

else userPassChange(vec\_of\_accounts, current\_account);

}

}

}

void AdminInterface(vector <Student>& vec\_of\_students, vector <Account>& vec\_of\_accounts, Account& current\_account, bool& similar) {

int admin\_acc\_menu\_choice, admin\_main\_menu\_choice;

bool admin\_main\_menu\_flag = true;

while (admin\_main\_menu\_flag) {

system("cls");

cout << "Добро пожаловать в систему расчета стипендии! " << endl;

cout << "Вы находитесь в режиме администратора!" << endl;

cout << "1) Система расчета стипендий" << endl << "2) Модуль управления аккаунтами" << endl << "3) Изменить пароль" << endl << "4) Выйти из аккаунта" << endl;

admin\_main\_menu\_choice = inputNumber(1, 4);

system("cls");

bool system\_menu\_flag = true, acc\_menu = true;

switch (admin\_main\_menu\_choice)

{

case 1:

while (system\_menu\_flag) {

cout << "----------------------------" << endl << " Система расчета стипений " << endl << "----------------------------" << endl << "Выберите действие: "

<< endl << "1) Общая информация о студентах"

<< endl << "2) Успеваемость студентов"

<< endl << "3) Назад" << endl;

int system\_menu\_choice = inputNumber(1, 3);

switch (system\_menu\_choice) {

case 1: showStudents(vec\_of\_students); break;

case 2: showStudentsPerfomance(vec\_of\_students); break;

case 3: system("cls"); system\_menu\_flag = false; break;

}

}

break;

case 2:

while (acc\_menu) {

cout << "----------------------------------------" << endl << " Модуль управления аккаунтами системы " << endl << "----------------------------------------" << endl << "Выберите действие: " << endl << "1) Показать все аккаунты" << endl << "2) Показать новые заявки на вход в систему" << endl << "3) Назад" << endl;

admin\_acc\_menu\_choice = inputNumber(1, 3);

system("cls");

switch (admin\_acc\_menu\_choice) {

case 1: showAccounts(vec\_of\_accounts, current\_account); break;

case 2: showNewApplications(vec\_of\_accounts); break;

case 3: acc\_menu = false; break;

}

}

break;

case 3:

userPassChange(vec\_of\_accounts, current\_account);

break;

case 4:

admin\_main\_menu\_flag = false;

similar = false;

break;

}

}

}

void System(vector <Student>& vec\_of\_students, vector <Account>& vec\_of\_accounts) {

string written\_login, written\_password;

Account current\_account;

bool similar = false;

bool system\_flag = true;

while (system\_flag) {

system("cls");

cout << "1) Войти" << endl << "2) Зарегистрировать новый аккаунт" << endl << "3) Выйти из системы" << endl;

int user\_choice = inputNumber(1, 3);

system("cls");

switch (user\_choice) {

case 1:

cout << "Авторизируйтесь в системе" << endl;

while (!similar) {

cout << "Введите логин: "; cin >> written\_login; cout << endl;

cout << "Введите пароль: ";

written\_password = inputPass();

cout << endl;

for (int i = 0; i < vec\_of\_accounts.size(); i++) {

if (vec\_of\_accounts[i].login == written\_login && vec\_of\_accounts[i].password == md5(written\_password)) {

current\_account = vec\_of\_accounts[i];

similar = true;

}

}

if (!similar) cout << "Данные введены некорректно! Повторите попытку. " << endl;

}

system("cls");

if (current\_account.block == 1) {

cout << "Ваш аккаунт заблокирован!" << endl;

system("pause");

similar = false;

}

else {

// MAIN ---------------------

switch (current\_account.role) {

case 1:

AdminInterface(vec\_of\_students, vec\_of\_accounts, current\_account, similar);

break;

case 0:

UserInterface(vec\_of\_students, vec\_of\_accounts, current\_account, similar);

break;

}

}

break;

case 2:

newAccountApplication(vec\_of\_accounts); break;

case 3: system\_flag = false; break;

}

}

}

validation.h

#pragma once

#include <iostream>

using namespace std;

// ПРОВЕРКА НА ВВОД

// Ввод чисел

int inputNumber(int left\_range, int right\_range);

// Проверка на число

bool isNumberNumeric();

// Проверка на ввод числа из корректного промежутка

bool isNumberRangeCorrect(int number, int left\_range, int right\_range);

validation.cpp

#include "validation.h"

bool isNumberNumeric() {

if (cin.get() == '\n') return true;

else

{

cin.clear();

cin.ignore(256, '\n');

return false;

}

}

bool isNumberRangeCorrect(int number, int left\_range, int right\_range) {

if ((number >= left\_range) && (number <= right\_range))return true;

else return false;

}

int inputNumber(int left\_range, int right\_range) {

int number;

while (true) {

cin >> number;

if (isNumberNumeric() && isNumberRangeCorrect(number, left\_range, right\_range)) {

return number;

}

else {

cout << "Некорректный ввод!" << endl;

cout << "Введите корректные данные: ";

}

}

}

**md5.h**

#ifndef BZF\_MD5\_H

#define BZF\_MD5\_H

#include <cstring>

#include <iostream>

// a small class for calculating MD5 hashes of strings or byte arrays

// it is not meant to be fast or secure

//

// usage: 1) feed it blocks of uchars with update()

// 2) finalize()

// 3) get hexdigest() string

// or

// MD5(std::string).hexdigest()

//

// assumes that char is 8 bit and int is 32 bit

class MD5

{

public:

typedef unsigned int size\_type; // must be 32bit

MD5();

MD5(const std::string& text);

void update(const unsigned char\* buf, size\_type length);

void update(const char\* buf, size\_type length);

MD5& finalize();

std::string hexdigest() const;

friend std::ostream& operator<<(std::ostream&, MD5 md5);

private:

void init();

typedef unsigned char uint1; // 8bit

typedef unsigned int uint4; // 32bit

enum { blocksize = 64 }; // VC6 won't eat a const static int here

void transform(const uint1 block[blocksize]);

static void decode(uint4 output[], const uint1 input[], size\_type len);

static void encode(uint1 output[], const uint4 input[], size\_type len);

bool finalized;

uint1 buffer[blocksize]; // bytes that didn't fit in last 64 byte chunk

uint4 count[2]; // 64bit counter for number of bits (lo, hi)

uint4 state[4]; // digest so far

uint1 digest[16]; // the result

// low level logic operations

static inline uint4 F(uint4 x, uint4 y, uint4 z);

static inline uint4 G(uint4 x, uint4 y, uint4 z);

static inline uint4 H(uint4 x, uint4 y, uint4 z);

static inline uint4 I(uint4 x, uint4 y, uint4 z);

static inline uint4 rotate\_left(uint4 x, int n);

static inline void FF(uint4& a, uint4 b, uint4 c, uint4 d, uint4 x, uint4 s, uint4 ac);

static inline void GG(uint4& a, uint4 b, uint4 c, uint4 d, uint4 x, uint4 s, uint4 ac);

static inline void HH(uint4& a, uint4 b, uint4 c, uint4 d, uint4 x, uint4 s, uint4 ac);

static inline void II(uint4& a, uint4 b, uint4 c, uint4 d, uint4 x, uint4 s, uint4 ac);

};

std::string md5(const std::string str);

#endif

md5.mpp

#include "md5.h"

/\* system implementation headers \*/

#include <cstdio>

// Constants for MD5Transform routine.

#define S11 7

#define S12 12

#define S13 17

#define S14 22

#define S21 5

#define S22 9

#define S23 14

#define S24 20

#define S31 4

#define S32 11

#define S33 16

#define S34 23

#define S41 6

#define S42 10

#define S43 15

#define S44 21

///////////////////////////////////////////////

// F, G, H and I are basic MD5 functions.

inline MD5::uint4 MD5::F(uint4 x, uint4 y, uint4 z) {

return x & y | ~x & z;

}

inline MD5::uint4 MD5::G(uint4 x, uint4 y, uint4 z) {

return x & z | y & ~z;

}

inline MD5::uint4 MD5::H(uint4 x, uint4 y, uint4 z) {

return x ^ y ^ z;

}

inline MD5::uint4 MD5::I(uint4 x, uint4 y, uint4 z) {

return y ^ (x | ~z);

}

// rotate\_left rotates x left n bits.

inline MD5::uint4 MD5::rotate\_left(uint4 x, int n) {

return (x << n) | (x >> (32 - n));

}

// FF, GG, HH, and II transformations for rounds 1, 2, 3, and 4.

// Rotation is separate from addition to prevent recomputation.

inline void MD5::FF(uint4& a, uint4 b, uint4 c, uint4 d, uint4 x, uint4 s, uint4 ac) {

a = rotate\_left(a + F(b, c, d) + x + ac, s) + b;

}

inline void MD5::GG(uint4& a, uint4 b, uint4 c, uint4 d, uint4 x, uint4 s, uint4 ac) {

a = rotate\_left(a + G(b, c, d) + x + ac, s) + b;

}

inline void MD5::HH(uint4& a, uint4 b, uint4 c, uint4 d, uint4 x, uint4 s, uint4 ac) {

a = rotate\_left(a + H(b, c, d) + x + ac, s) + b;

}

inline void MD5::II(uint4& a, uint4 b, uint4 c, uint4 d, uint4 x, uint4 s, uint4 ac) {

a = rotate\_left(a + I(b, c, d) + x + ac, s) + b;

}

//////////////////////////////////////////////

// default ctor, just initailize

MD5::MD5()

{

init();

}

//////////////////////////////////////////////

// nifty shortcut ctor, compute MD5 for string and finalize it right away

MD5::MD5(const std::string& text)

{

init();

update(text.c\_str(), text.length());

finalize();

}

//////////////////////////////

void MD5::init()

{

finalized = false;

count[0] = 0;

count[1] = 0;

// load magic initialization constants.

state[0] = 0x67452301;

state[1] = 0xefcdab89;

state[2] = 0x98badcfe;

state[3] = 0x10325476;

}

//////////////////////////////

// decodes input (unsigned char) into output (uint4). Assumes len is a multiple of 4.

void MD5::decode(uint4 output[], const uint1 input[], size\_type len)

{

for (unsigned int i = 0, j = 0; j < len; i++, j += 4)

output[i] = ((uint4)input[j]) | (((uint4)input[j + 1]) << 8) |

(((uint4)input[j + 2]) << 16) | (((uint4)input[j + 3]) << 24);

}

//////////////////////////////

// encodes input (uint4) into output (unsigned char). Assumes len is

// a multiple of 4.

void MD5::encode(uint1 output[], const uint4 input[], size\_type len)

{

for (size\_type i = 0, j = 0; j < len; i++, j += 4) {

output[j] = input[i] & 0xff;

output[j + 1] = (input[i] >> 8) & 0xff;

output[j + 2] = (input[i] >> 16) & 0xff;

output[j + 3] = (input[i] >> 24) & 0xff;

}

}

//////////////////////////////

// apply MD5 algo on a block

void MD5::transform(const uint1 block[blocksize])

{

uint4 a = state[0], b = state[1], c = state[2], d = state[3], x[16];

decode(x, block, blocksize);

/\* Round 1 \*/

FF(a, b, c, d, x[0], S11, 0xd76aa478); /\* 1 \*/

FF(d, a, b, c, x[1], S12, 0xe8c7b756); /\* 2 \*/

FF(c, d, a, b, x[2], S13, 0x242070db); /\* 3 \*/

FF(b, c, d, a, x[3], S14, 0xc1bdceee); /\* 4 \*/

FF(a, b, c, d, x[4], S11, 0xf57c0faf); /\* 5 \*/

FF(d, a, b, c, x[5], S12, 0x4787c62a); /\* 6 \*/

FF(c, d, a, b, x[6], S13, 0xa8304613); /\* 7 \*/

FF(b, c, d, a, x[7], S14, 0xfd469501); /\* 8 \*/

FF(a, b, c, d, x[8], S11, 0x698098d8); /\* 9 \*/

FF(d, a, b, c, x[9], S12, 0x8b44f7af); /\* 10 \*/

FF(c, d, a, b, x[10], S13, 0xffff5bb1); /\* 11 \*/

FF(b, c, d, a, x[11], S14, 0x895cd7be); /\* 12 \*/

FF(a, b, c, d, x[12], S11, 0x6b901122); /\* 13 \*/

FF(d, a, b, c, x[13], S12, 0xfd987193); /\* 14 \*/

FF(c, d, a, b, x[14], S13, 0xa679438e); /\* 15 \*/

FF(b, c, d, a, x[15], S14, 0x49b40821); /\* 16 \*/

/\* Round 2 \*/

GG(a, b, c, d, x[1], S21, 0xf61e2562); /\* 17 \*/

GG(d, a, b, c, x[6], S22, 0xc040b340); /\* 18 \*/

GG(c, d, a, b, x[11], S23, 0x265e5a51); /\* 19 \*/

GG(b, c, d, a, x[0], S24, 0xe9b6c7aa); /\* 20 \*/

GG(a, b, c, d, x[5], S21, 0xd62f105d); /\* 21 \*/

GG(d, a, b, c, x[10], S22, 0x2441453); /\* 22 \*/

GG(c, d, a, b, x[15], S23, 0xd8a1e681); /\* 23 \*/

GG(b, c, d, a, x[4], S24, 0xe7d3fbc8); /\* 24 \*/

GG(a, b, c, d, x[9], S21, 0x21e1cde6); /\* 25 \*/

GG(d, a, b, c, x[14], S22, 0xc33707d6); /\* 26 \*/

GG(c, d, a, b, x[3], S23, 0xf4d50d87); /\* 27 \*/

GG(b, c, d, a, x[8], S24, 0x455a14ed); /\* 28 \*/

GG(a, b, c, d, x[13], S21, 0xa9e3e905); /\* 29 \*/

GG(d, a, b, c, x[2], S22, 0xfcefa3f8); /\* 30 \*/

GG(c, d, a, b, x[7], S23, 0x676f02d9); /\* 31 \*/

GG(b, c, d, a, x[12], S24, 0x8d2a4c8a); /\* 32 \*/

/\* Round 3 \*/

HH(a, b, c, d, x[5], S31, 0xfffa3942); /\* 33 \*/

HH(d, a, b, c, x[8], S32, 0x8771f681); /\* 34 \*/

HH(c, d, a, b, x[11], S33, 0x6d9d6122); /\* 35 \*/

HH(b, c, d, a, x[14], S34, 0xfde5380c); /\* 36 \*/

HH(a, b, c, d, x[1], S31, 0xa4beea44); /\* 37 \*/

HH(d, a, b, c, x[4], S32, 0x4bdecfa9); /\* 38 \*/

HH(c, d, a, b, x[7], S33, 0xf6bb4b60); /\* 39 \*/

HH(b, c, d, a, x[10], S34, 0xbebfbc70); /\* 40 \*/

HH(a, b, c, d, x[13], S31, 0x289b7ec6); /\* 41 \*/

HH(d, a, b, c, x[0], S32, 0xeaa127fa); /\* 42 \*/

HH(c, d, a, b, x[3], S33, 0xd4ef3085); /\* 43 \*/

HH(b, c, d, a, x[6], S34, 0x4881d05); /\* 44 \*/

HH(a, b, c, d, x[9], S31, 0xd9d4d039); /\* 45 \*/

HH(d, a, b, c, x[12], S32, 0xe6db99e5); /\* 46 \*/

HH(c, d, a, b, x[15], S33, 0x1fa27cf8); /\* 47 \*/

HH(b, c, d, a, x[2], S34, 0xc4ac5665); /\* 48 \*/

/\* Round 4 \*/

II(a, b, c, d, x[0], S41, 0xf4292244); /\* 49 \*/

II(d, a, b, c, x[7], S42, 0x432aff97); /\* 50 \*/

II(c, d, a, b, x[14], S43, 0xab9423a7); /\* 51 \*/

II(b, c, d, a, x[5], S44, 0xfc93a039); /\* 52 \*/

II(a, b, c, d, x[12], S41, 0x655b59c3); /\* 53 \*/

II(d, a, b, c, x[3], S42, 0x8f0ccc92); /\* 54 \*/

II(c, d, a, b, x[10], S43, 0xffeff47d); /\* 55 \*/

II(b, c, d, a, x[1], S44, 0x85845dd1); /\* 56 \*/

II(a, b, c, d, x[8], S41, 0x6fa87e4f); /\* 57 \*/

II(d, a, b, c, x[15], S42, 0xfe2ce6e0); /\* 58 \*/

II(c, d, a, b, x[6], S43, 0xa3014314); /\* 59 \*/

II(b, c, d, a, x[13], S44, 0x4e0811a1); /\* 60 \*/

II(a, b, c, d, x[4], S41, 0xf7537e82); /\* 61 \*/

II(d, a, b, c, x[11], S42, 0xbd3af235); /\* 62 \*/

II(c, d, a, b, x[2], S43, 0x2ad7d2bb); /\* 63 \*/

II(b, c, d, a, x[9], S44, 0xeb86d391); /\* 64 \*/

state[0] += a;

state[1] += b;

state[2] += c;

state[3] += d;

// Zeroize sensitive information.

memset(x, 0, sizeof x);

}

//////////////////////////////

// MD5 block update operation. Continues an MD5 message-digest

// operation, processing another message block

void MD5::update(const unsigned char input[], size\_type length)

{

// compute number of bytes mod 64

size\_type index = count[0] / 8 % blocksize;

// Update number of bits

if ((count[0] += (length << 3)) < (length << 3))

count[1]++;

count[1] += (length >> 29);

// number of bytes we need to fill in buffer

size\_type firstpart = 64 - index;

size\_type i;

// transform as many times as possible.

if (length >= firstpart)

{

// fill buffer first, transform

memcpy(&buffer[index], input, firstpart);

transform(buffer);

// transform chunks of blocksize (64 bytes)

for (i = firstpart; i + blocksize <= length; i += blocksize)

transform(&input[i]);

index = 0;

}

else

i = 0;

// buffer remaining input

memcpy(&buffer[index], &input[i], length - i);

}

//////////////////////////////

// for convenience provide a verson with signed char

void MD5::update(const char input[], size\_type length)

{

update((const unsigned char\*)input, length);

}

//////////////////////////////

// MD5 finalization. Ends an MD5 message-digest operation, writing the

// the message digest and zeroizing the context.

MD5& MD5::finalize()

{

static unsigned char padding[64] = {

0x80, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,

0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,

0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0

};

if (!finalized) {

// Save number of bits

unsigned char bits[8];

encode(bits, count, 8);

// pad out to 56 mod 64.

size\_type index = count[0] / 8 % 64;

size\_type padLen = (index < 56) ? (56 - index) : (120 - index);

update(padding, padLen);

// Append length (before padding)

update(bits, 8);

// Store state in digest

encode(digest, state, 16);

// Zeroize sensitive information.

memset(buffer, 0, sizeof buffer);

memset(count, 0, sizeof count);

finalized = true;

}

return \*this;

}

//////////////////////////////

// return hex representation of digest as string

std::string MD5::hexdigest() const

{

if (!finalized)

return "";

char buf[33];

for (int i = 0; i < 16; i++)

sprintf(buf + i \* 2, "%02x", digest[i]);

buf[32] = 0;

return std::string(buf);

}

//////////////////////////////

std::ostream& operator<<(std::ostream& out, MD5 md5)

{

return out << md5.hexdigest();

}

//////////////////////////////

std::string md5(const std::string str)

{

MD5 md5 = MD5(str);

return md5.hexdigest();

}

accs.h

#pragma once

#include <iostream>

#include <string>

#include <fstream>

#include <conio.h>

#include <vector>

#include <algorithm>

#include "md5.h"

#include "validation.h"

using namespace std;

const string FILE\_OF\_ACCOUNTS = "accounts.txt";

// Структура аккаунта

struct Account

{

string login; // Логин аккаунта

string password; // Пароль хешированный через md5

int role; // Роль, где 1 - admin, 2 - user

int access; // Доступ. Аккаунты без доступа к системе имеют значение - 0 (присваивается после заявки на регистрацию нового аккаунта), с доступом - 1

int block; // Блокировка. Заблокированные аккаунты имеют значение - 1 в этом поле, остальные - 0 (по умолчанию)

};

// ФУНКЦИИ ДЛЯ РАБОТЫ С АККАУНТАМИ

// Изменение пароля пользователем

void userPassChange(vector <Account>& v, Account& current\_acc);

// Ввод пароля с маскировкой (возвращает пароль)

string inputPass();

// Удаление новой заявкм

void deleteApplication(vector <Account>& v, int counter);

// Поиск индекса заявки

int searchApplicationIndex(vector <Account>& v, int counter);

// Подтверждение заявки

void ApproveApplication(vector <Account>& v, int counter);

// Отображение заявок

void showNewApplications(vector <Account>& v);

// Удалить аккаунт

void deleteAccount(vector <Account>& v, int counter, Account current\_acc);

// Изменить роль аккаунта

void roleChange(vector <Account>& v, int counter);

// Поиск индекса нужного аккаунта

int searchAccIndex(vector <Account>& v, int counter);

// Блокировка аккаунта

void blockAccount(vector <Account>& v, int counter);

// Разблокировка аккаунта

void unblockAccount(vector <Account>& v, int counter);

// Отображение всех аккаунтов

void showAccounts(vector <Account>& v, Account current\_acc);

// Создание нового аккаунта администратором

void newAccountCreation(vector <Account>& v);

// Запись всех аккаунтов в файл

void writeFileAccounts(vector <Account>& v);

// Создание нового акккаунта

void newAccountApplication(vector <Account>& v);

// Подсчет кол-ва строк в файле

int getCountOfStucturesInFile(string file\_path);

//Чтение аккаунтов из файла

void readFileAccounts(vector <Account>& v);

// Сортировки аккаунтов

// Сортировка по алфавиту (по полю с логинами)

bool mySortByLogin(Account a, Account b);

void sortByLogin(vector <Account>& v);

// Сортировка по роли (администраторы > пользователи)

bool mySortByRole(Account a, Account b);

void sortByRole(vector <Account>& v);

accs.cpp

#include "accs.h"

// Аккаунты

string inputPass() {

string pass;

char input[30], c;

int counter = 0;

while ((c = \_getch()) != '\r')

{

input[counter++] = c;

\_putch('\*');

}

pass.assign(input, 0, counter);

cout << endl;

return pass;

}

void showAccounts(vector <Account>& v, Account current\_acc) {

int counter = 1;

bool acc\_table\_flag = true;

while (acc\_table\_flag) {

cout << "№\tlogin\trole\tstate" << endl;

cout << "---------------------------------------" << endl;

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

if (v[i].access == 1) {

cout << counter++ << ")\t"

<< v[i].login << "\t";

if (v[i].role == 0) {

cout << "user" << "\t";

}

else { cout << "admin" << "\t"; }

if (v[i].block == 0) {

cout << "active" << endl;

}

else { cout << "blocked" << endl; }

}

}

cout << endl << endl;

cout << "Выберите действие: " << endl << "1) Отсортировать по алфавиту" << endl << "2) Отсортировать по роли" << endl << "3) Добавить новый аккаунт" << endl << "4) Удалить аккаунт" << endl << "5) Заблокировать аккаунт" << endl << "6) Разблокировать аккаунт" << endl << "7) Изменить роль" << endl << "8) Назад" << endl;

int user\_choice = inputNumber(1, 8);

switch (user\_choice) {

case 1: sortByLogin(v); break;

case 2: sortByRole(v); break;

case 3: newAccountCreation(v); break;

case 4: deleteAccount(v, counter, current\_acc); break;

case 5: blockAccount(v, counter); break;

case 6: unblockAccount(v, counter); break;

case 7: roleChange(v, counter); break;

case 8: acc\_table\_flag = false; break;

}

system("cls");

counter = 1;

}

}

void userPassChange(vector <Account>& v, Account& current\_acc) {

system("cls");

string pass\_comparison;

bool first\_attempt = true;

int fail\_counter = 0;

while ((current\_acc.password != md5(pass\_comparison)) && (fail\_counter != 3)) {

if (first\_attempt) cout << "Для изменения пароля введите свой текущий пароль: ";

else cout << "Некорректный ввод пароля. Повторите попытку: ";

pass\_comparison = inputPass();

fail\_counter++;

first\_attempt = false;

}

if (fail\_counter != 3) {

string new\_pass, new\_pass\_copy;

cout << "Введите новый пароль: ";

new\_pass = inputPass();

cout << endl;

bool first\_attempt = true, match = false;

int fails = 0;

while (!match && fails != 3) {

if (first\_attempt) cout << "Повторите новый пароль: ";

else cout << "Некорректный ввод! Повторите новый пароль: ";

new\_pass\_copy = inputPass();

if (new\_pass == new\_pass\_copy) match = true;

fails++;

first\_attempt = false;

}

if (fails != 3) {

current\_acc.password = new\_pass;

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

if (v[i].login == current\_acc.login) {

v[i].password = md5(new\_pass); break;

}

}

cout << "Ваш пароль изменен успешно!" << endl;

system("pause");

}

else {

cout << "Повторный ввод пароля не совпал трижды. Операция изменения пароля завершена. Пароль не изменен!" << endl;

system("pause");

}

}

else {

cout << "Неправильный пароль был введен трижды. Изменение пароля недоступно!" << endl;

system("pause");

system("cls");

}

}

void showNewApplications(vector <Account>& v) {

int counter = 1;

bool acc\_table\_flag = true;

while (acc\_table\_flag) {

cout << "№\tlogin" << endl;

cout << "--------------------" << endl;

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

if (v[i].access == 0) {

cout << counter++ << ")\t"

<< v[i].login << endl;

}

}

cout << endl << endl;

cout << "Выберите действие: " << endl << "1) Отсортировать по алфавиту" << endl << "2) Одобрить заявку на регистрацию" << endl << "3) Удалить заявку" << endl << "4) Назад" << endl;

int user\_choice = inputNumber(1, 4);

switch (user\_choice) {

case 1: sortByLogin(v); break;

case 2: ApproveApplication(v, counter); break;

case 3: deleteApplication(v, counter); break;

case 4: acc\_table\_flag = false; break;

}

system("cls");

counter = 1;

}

}

void deleteApplication(vector <Account>& v, int counter) {

cout << "Введите номер заявки: ";

int index = searchApplicationIndex(v, counter);

system("cls");

cout << "Вы действительно хотите удалить заявку на регистрацию пользователя " << v[index].login << " из системы?" << endl << "1) Да" << endl << "2) Отменить" << endl;

int choice = inputNumber(1, 2);

if (choice == 1) v.erase(v.begin() + index);

system("cls");

}

void ApproveApplication(vector <Account>& v, int counter) {

cout << "Введите номер заявки: ";

int index = searchApplicationIndex(v, counter);

system("cls");

cout << "Вы действительно хотите одобрить заявку на регистрацию пользователя " << v[index].login << " в системе?" << endl << "1) Да" << endl << "2) Отменить" << endl;

int choice = inputNumber(1, 2);

if (choice == 1) v[index].access = 1;

system("cls");

}

void deleteAccount(vector <Account>& v, int counter, Account current\_acc) {

int index;

while (true) {

cout << "Введите номер удаляемого аккаунта: ";

index = searchAccIndex(v, counter);

if (v[index].login == current\_acc.login) cout << "Вы не можете удалить свой аккаунт!" << endl;

else break;

}

system("cls");

cout << "Вы действительно хотите удалить акккаунт " << v[index].login << " из системы?" << endl << "1) Да" << endl << "2) Отменить" << endl;

int choice = inputNumber(1, 2);

if (choice == 1) v.erase(v.begin() + index);

system("cls");

}

void roleChange(vector <Account>& v, int counter) {

cout << "Введите номер аккаунта: ";

int index = searchAccIndex(v, counter);

if (v[index].role == 1) { v[index].role = 0; }

else { v[index].role = 1; }

}

void blockAccount(vector <Account>& v, int counter) {

cout << "Введите номер блокируемого аккаунта: ";

int index = searchAccIndex(v, counter);

v[index].block = 1;

}

void unblockAccount(vector <Account>& v, int counter) {

cout << "Введите номер аккаунта: ";

int index = searchAccIndex(v, counter);

v[index].block = 0;

}

int searchApplicationIndex(vector <Account>& v, int counter) {

int index, number = inputNumber(1, counter - 1);

int k = 0;

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

if (v[i].access == 0) {

k++;

}

if (k == number) {

index = i;

break;

}

}

return index;

}

int searchAccIndex(vector <Account>& v, int counter) {

int index, number = inputNumber(1, counter - 1);

int k = 0;

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

if (v[i].access == 1) {

k++;

}

if (k == number) {

index = i;

break;

}

}

return index;

}

void newAccountCreation(vector <Account>& v) {

system("cls");

Account new\_acc;

cout << "Создание нового аккаунта в системе" << endl;

bool flag = false;

while (!flag) {

flag = true;

cout << "Введите логин: ";

cin >> new\_acc.login;

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

if (v[i].login == new\_acc.login) {

flag = false;

cout << "Аккаунт с таким логином уже зарегистрирован! Пожалуйста, придумайте другой." << endl;

}

}

}

cout << "Введите пароль: ";

cin >> new\_acc.password;

cout << "1) Аккаунт администратора" << endl << "2) Пользовательский аккаунт" << endl;

int user\_choice = inputNumber(1, 2);

if (user\_choice == 1) {

new\_acc.role = 1;

}

else

{

new\_acc.role = 0;

cout << "1) С доступом к системе" << endl << "2) Без доступа к системе" << endl;

user\_choice = inputNumber(1, 2);

if (user\_choice == 1) {

new\_acc.access = 1;

}

else

{

new\_acc.access = 0;

}

}

new\_acc.block = 0;

v.push\_back(new\_acc);

system("cls");

cout << "Новый аккаунт успешно создан!" << endl;

}

void newAccountApplication(vector <Account>& v) {

Account new\_acc;

cout << "Создание нового аккаунта в системе" << endl;

bool flag = false;

while (!flag) {

flag = true;

cout << "Введите логин: ";

cin >> new\_acc.login;

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

if (v[i].login == new\_acc.login) {

flag = false;

cout << "Аккаунт с таким логином уже зарегистрирован! Пожалуйста, придумайте другой." << endl;

}

}

}

cout << "Введите пароль: ";

cin >> new\_acc.password;

new\_acc.password = md5(new\_acc.password);

new\_acc.access = 0;

new\_acc.block = 0;

new\_acc.role = 0;

v.push\_back(new\_acc);

system("cls");

cout << "Ваша заявка на регистрацию принята!" << endl << "Система будет доступна после одобрения вашей заявки администратором." << endl;

system("pause");

}

void writeFileAccounts(vector <Account>& v)

{

ofstream fout(FILE\_OF\_ACCOUNTS, ios::out); // Открыли файл для записи

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

{

fout << v[i].login << " "

<< v[i].password << " "

<< v[i].role << " "

<< v[i].access << " "

<< v[i].block;

if (i < v.size() - 1)

{

fout << endl;

}

}

fout.close();

}

void readFileAccounts(vector <Account>& v)

{

ifstream fin(FILE\_OF\_ACCOUNTS, ios::in); // Открыли файл для чтения

if (!fin.is\_open()) {

cout << "На данный момент не существует ни одного аккаунта." << endl << "Пожалуйста, зарегистрируйтесь в системе!" << endl;

newAccountCreation(v);

}

else

{

int i = 0;

while (!fin.eof())

{

if (i < v.size())

{

fin >> v[i].login

>> v[i].password

>> v[i].role

>> v[i].access

>> v[i].block;

i++;

}

else

{

break;

}

}

}

fin.close(); //Закрыли файл

}

int getCountOfStucturesInFile(string file\_path)

{

ifstream file(file\_path, ios::in); // Открыли текстовый файл для чтения

int number\_of\_strings = 0;

if (file.is\_open())

{

string buffer;

while (getline(file, buffer))

number\_of\_strings++;

}

file.close();

return number\_of\_strings;

}

// Аккаунты

bool mySortByLogin(Account a, Account b) {

return a.login < b.login;

}

void sortByLogin(vector <Account>& v) {

system("cls");

sort(v.begin(), v.end(), mySortByLogin);

}

bool mySortByRole(Account a, Account b) {

return a.role > b.role;

}

void sortByRole(vector <Account>& v) {

system("cls");

sort(v.begin(), v.end(), mySortByRole);

}

students.h

#pragma once

#include <iostream>

#include <string>

#include <fstream>

#include <vector>

#include <algorithm>

#include "validation.h"

using namespace std;

const string FILE\_OF\_DATA = "students.txt";

// Структура для оценок

struct Marks {

//экзамены

int os;

int math;

int oaip;

int fil;

//зачеты

int hist;

int pe;

int ikg;

int okp;

int english;

double average;

};

// Стурктура для студента

struct Student

{

string surname; // Фамилия

int group\_number; // Номер группы

int active; // Показатель активности (1 - активен, 0 - неактивен)

double benefits; // Размер стипендии

Marks marks; // Отдельная структура для фиксации оценок по каждому предмету

};

// ФУНКЦИИ ДЛЯ РАБОТЫ СО СТУДЕНТАМИ

// Отображение списка студентов с основной информацией

void showStudents(vector <Student>& v);

// Чтение информации о студентах из файла в вектор

void readFileStudents(vector <Student>& v);

// Запись вектора со студентами в файл

void writeFileStudents(vector <Student>& v);

// Отображение списка студентов с их успеваемостью

void showStudentsPerfomance(vector <Student>& v);

// Удаление конкретного студента

void deleteStudent(vector <Student>& v);

// Флаг для отличников

bool otlichnik(Student a);

// Изменение размера базовой стипендии

void defaultBenefitChange(vector <Student>& v);

// Добавление нового студента в систему

void addStudent(vector <Student>& v);

// Изменение информации о студенте

void editStudent(vector <Student>& v);

// Возможность изменения информации о только что добаленном студенте

void editAddedStudent(Student a);

// Пользовательский вариант меню со студентами

void showStudentsForUser(vector <Student>& v);

// Функции сортировки

// Сортировка по размеру стипендии

bool mySortByBenefit(Student a, Student b);

void sortByBenefit(vector <Student>& v);

// Сортировка по активности

bool mySortByActive(Student a, Student b);

void sortByActive(vector <Student>& v);

// Сортировка по среднему баллу

bool mySortByAverageMark(Student a, Student b);

void sortByAverageMark(vector <Student>& v);

// Сортировка по алфавиту (по полю с фамилиями)

bool mySortBySurname(Student a, Student b);

void sortBySurname(vector <Student>& v);

students.cpp

#include "students.h"

void showStudentsPerfomance(vector <Student>& v) {

system("cls");

bool flag = true;

int number = 1;

while (flag) {

cout << " Экзамены Зачеты" << endl;

cout << "№ Фамилия\tОС/Мат/ОАиП/Фил\t\tИст/ФК/ИКГ/ОКП/Англ\tСр. балл" << endl;

cout << "------------------------------------------------------------------------" << endl;

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

cout << number++ << ") "

<< v[i].surname << "\t"

<< " " << v[i].marks.os << " / "

<< v[i].marks.math << " / "

<< v[i].marks.oaip << " / "

<< v[i].marks.fil << "\t\t"

<< " " << v[i].marks.hist << " / "

<< v[i].marks.pe << " / "

<< v[i].marks.ikg << " / "

<< v[i].marks.okp << " / "

<< v[i].marks.english << "\t"

<< v[i].marks.average << endl;

}

cout << endl;

cout << "Выберите действие: " << endl << "1) Отсортировать по алфавиту" << endl << "2) Отсортировать по среднему баллу" << endl << "3) Назад" << endl;

int user\_choice = inputNumber(1, 3);

switch (user\_choice)

{

case 1: sortBySurname(v); break;

case 2: sortByAverageMark(v); break;

case 3: flag = false; system("cls"); break;

}

number = 1;

system("cls");

}

}

void showStudentsForUser(vector <Student>& v) {

system("cls");

bool system\_menu\_flag = true;

int number = 1;

while (system\_menu\_flag) {

cout << "№ Фамилия\tГруппа\tАктивность\tСр.б.\tСтипендия" << endl;

cout << "---------------------------------------------------------" << endl;

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

cout << number++ << ") "

<< v[i].surname << "\t"

<< v[i].group\_number << "\t";

if (v[i].active == 0) {

cout << "Неактивен" << "\t";

}

else { cout << " Активен" << "\t"; }

cout << v[i].marks.average << "\t"

<< v[i].benefits << endl;

}

cout << endl;

cout << "Выберите действие: " << endl << "1) Отсортировать по алфавиту" << endl << "2) Отсортировать по активности" << endl << "3) Отсортировать по среднему баллу" << endl << "4) Отсортировать по размеру стипендии" << endl << "5) Изменить базовую стипендию" << endl << "6) Назад" << endl;

int user\_choice = inputNumber(1, 6);

switch (user\_choice) {

case 1: sortBySurname(v); break;

case 2: sortByActive(v); break;

case 3: sortByAverageMark(v); break;

case 4: sortByBenefit(v); break;

case 5: defaultBenefitChange(v); break;

case 6: system\_menu\_flag = false; break;

}

number = 1;

system("cls");

}

}

void showStudents(vector <Student>& v) {

system("cls");

bool system\_menu\_flag = true;

int number = 1;

while (system\_menu\_flag) {

cout << "№ Фамилия\tГруппа\tАктивность\tСр.б.\tСтипендия" << endl;

cout << "---------------------------------------------------------" << endl;

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

cout << number++ << ") "

<< v[i].surname << "\t"

<< v[i].group\_number << "\t";

if (v[i].active == 0) {

cout << "Неактивен" << "\t";

}

else { cout << " Активен" << "\t"; }

cout << v[i].marks.average << "\t"

<< v[i].benefits << endl;

}

cout << endl;

cout << "Выберите действие: " << endl << "1) Отсортировать по алфавиту" << endl << "2) Отсортировать по активности" << endl << "3) Отсортировать по среднему баллу" << endl << "4) Отсортировать по размеру стипендии" << endl << "5) Добавить студента" << endl << "6) Редактировать данные студента" << endl << "7) Удалить студента" << endl << "8) Изменить базовую стипендию" << endl << "9) Назад" << endl;

int user\_choice = inputNumber(1, 9);

switch (user\_choice) {

case 1: sortBySurname(v); break;

case 2: sortByActive(v); break;

case 3: sortByAverageMark(v); break;

case 4: sortByBenefit(v); break;

case 5: addStudent(v); break;

case 6: editStudent(v); break;

case 7: deleteStudent(v); break;

case 8: defaultBenefitChange(v); break;

case 9: system\_menu\_flag = false; break;

}

number = 1;

system("cls");

}

}

void editStudent(vector <Student>& v) {

cout << "Введите номер изменяемой записи: ";

int index = inputNumber(1, v.size()) - 1;

system("cls");

bool flag = true;

while (flag) {

system("cls");

cout << "Какую информацию вы хотите изменить?" << endl

<< "1) Фамилию (" << v[index].surname << ")" << endl

<< "2) Номер группы (" << v[index].group\_number << ")" << endl

<< "3) Активность (";

if (v[index].active == 1) { cout << "Активен"; }

else { cout << "Неактивен"; }

cout << ")" << endl

<< "4) Оценку по Математике (" << v[index].marks.math << ")" << endl

<< "5) Оценку по ОС (" << v[index].marks.os << ")" << endl

<< "6) Оценку по Философии (" << v[index].marks.fil << ")" << endl

<< "7) Оценку по ОАиП (" << v[index].marks.oaip << ")" << endl

<< "8) Зачет по ИКГ (";

if (v[index].marks.ikg == 1) { cout << "Зачет"; }

else { cout << "Незачет"; }

cout << ")" << endl

<< "9) Зачет по Английскому языку (";

if (v[index].marks.english == 1) { cout << "Зачет"; }

else { cout << "Незачет"; }

cout << ")" << endl

<< "10) Зачет по ОКП (";

if (v[index].marks.okp == 1) { cout << "Зачет"; }

else { cout << "Незачет"; }

cout << ")" << endl

<< "11) Зачет по Истории (";

if (v[index].marks.hist == 1) { cout << "Зачет"; }

else { cout << "Незачет"; }

cout << ")" << endl

<< "12) Зачет по Физической культуре (";

if (v[index].marks.pe == 1) { cout << "Зачет"; }

else { cout << "Незачет"; }

cout << ")" << endl

<< "13) Завершить редактирование" << endl;

int choice = inputNumber(1, 13);

switch (choice)

{

case 1:

cout << "Введите новую фамилию: ";

cin >> v[index].surname;

break;

case 2:

cout << "Введите новый номер группы: ";

cin >> v[index].group\_number;

break;

case 3:

if (v[index].active == 1) { v[index].active = 0; }

else { v[index].active = 1; }

break;

case 4:

cout << "Новая оценка по Математике: ";

cin >> v[index].marks.math;

break;

case 5:

cout << "Новая оценка по ОС: ";

cin >> v[index].marks.os;

break;

case 6:

cout << "Новая оценка по Философии: ";

cin >> v[index].marks.fil;

break;

case 7:

cout << "Новая оценка по ОАиП: ";

cin >> v[index].marks.oaip;

break;

case 8:

if (v[index].marks.ikg == 1) { v[index].marks.ikg = 0; }

else { v[index].marks.ikg = 1; }

break;

case 9:

if (v[index].marks.english == 1) { v[index].marks.english = 0; }

else { v[index].marks.english = 1; }

break;

case 10:

if (v[index].marks.okp == 1) { v[index].marks.okp = 0; }

else { v[index].marks.okp = 1; }

break;

case 11:

if (v[index].marks.hist == 1) { v[index].marks.hist = 0; }

else { v[index].marks.hist = 1; }

break;

case 12:

if (v[index].marks.pe == 1) { v[index].marks.pe = 0; }

else { v[index].marks.pe = 1; }

break;

case 13: flag = false; break;

}

v[index].marks.average = (v[index].marks.os + v[index].marks.oaip + v[index].marks.math + v[index].marks.fil) / 4.;

}

}

void editAddedStudent(Student a) {

system("cls");

bool flag = true;

while (flag) {

system("cls");

cout << "Какую информацию вы хотите изменить?" << endl

<< "1) Фамилию (" << a.surname << ")" << endl

<< "2) Номер группы (" << a.group\_number << ")" << endl

<< "3) Активность (";

if (a.active == 1) { cout << "Активен"; }

else { cout << "Неактивен"; }

cout << ")" << endl

<< "4) Оценку по Математике (" << a.marks.math << ")" << endl

<< "5) Оценку по ОС (" << a.marks.os << ")" << endl

<< "6) Оценку по Философии (" << a.marks.fil << ")" << endl

<< "7) Оценку по ОАиП (" << a.marks.oaip << ")" << endl

<< "8) Зачет по ИКГ (";

if (a.marks.ikg == 1) { cout << "Зачет"; }

else { cout << "Незачет"; }

cout << ")" << endl

<< "9) Зачет по Английскому языку (";

if (a.marks.english == 1) { cout << "Зачет"; }

else { cout << "Незачет"; }

cout << ")" << endl

<< "10) Зачет по ОКП (";

if (a.marks.okp == 1) { cout << "Зачет"; }

else { cout << "Незачет"; }

cout << ")" << endl

<< "11) Зачет по Истории (";

if (a.marks.hist == 1) { cout << "Зачет"; }

else { cout << "Незачет"; }

cout << ")" << endl

<< "12) Зачет по Физической культуре (";

if (a.marks.pe == 1) { cout << "Зачет"; }

else { cout << "Незачет"; }

cout << ")" << endl

<< "13) Завершить редактирование и сохранить студента" << endl;

int choice = inputNumber(1, 13);

switch (choice)

{

case 1:

cout << "Введите новую фамилию: ";

cin >> a.surname;

break;

case 2:

cout << "Введите новый номер группы: ";

cin >> a.group\_number;

break;

case 3:

if (a.active == 1) { a.active = 0; }

else { a.active = 1; }

break;

case 4:

cout << "Новая оценка по Математике: ";

cin >> a.marks.math;

break;

case 5:

cout << "Новая оценка по ОС: ";

cin >> a.marks.os;

break;

case 6:

cout << "Новая оценка по Философии: ";

cin >> a.marks.fil;

break;

case 7:

cout << "Новая оценка по ОАиП: ";

cin >> a.marks.oaip;

break;

case 8:

if (a.marks.ikg == 1) { a.marks.ikg = 0; }

else { a.marks.ikg = 1; }

break;

case 9:

if (a.marks.english == 1) { a.marks.english = 0; }

else { a.marks.english = 1; }

break;

case 10:

if (a.marks.okp == 1) { a.marks.okp = 0; }

else { a.marks.okp = 1; }

break;

case 11:

if (a.marks.hist == 1) { a.marks.hist = 0; }

else { a.marks.hist = 1; }

break;

case 12:

if (a.marks.pe == 1) { a.marks.pe = 0; }

else { a.marks.pe = 1; }

break;

case 13: flag = false; break;

}

a.marks.average = (a.marks.os + a.marks.oaip + a.marks.math + a.marks.fil) / 4.;

}

}

void addStudent(vector <Student>& v) {

system("cls");

Student new\_student;

cout << "Добавление нового студента в систему" << endl;

cout << "Введите фамилию: ";

cin >> new\_student.surname;

cout << "Введите номер группы: ";

cin >> new\_student.group\_number;

cout << "Студент ведет активную общественную жизнь?" << endl << "1) Да" << endl << "2) Нет" << endl;

int active = inputNumber(1, 2);

if (active == 1) { new\_student.active = 1; }

else { new\_student.active = 0; }

cout << "Оценка студента за экзамен по ОС:";

new\_student.marks.os = inputNumber(4, 10);

cout << "Оценка студента за экзамен по ОАиП:";

new\_student.marks.oaip = inputNumber(4, 10);

cout << "Оценка студента за экзамен по Математике:";

new\_student.marks.math = inputNumber(4, 10);

cout << "Оценка студента за экзамен по Философии:";

new\_student.marks.fil = inputNumber(4, 10);

new\_student.marks.average = (new\_student.marks.os + new\_student.marks.oaip + new\_student.marks.math + new\_student.marks.fil) / 4.;

new\_student.benefits = 0;

cout << "Студент сдал зачет по дисциплине 'ИКГ'?" << endl << "1) Да" << endl << "2) Нет" << endl;

int choice = inputNumber(1, 2);

if (choice == 1) { new\_student.marks.ikg = 1; }

else { new\_student.marks.ikg = 0; }

cout << "Студент сдал зачет по дисциплине 'Физическая культура'?" << endl << "1) Да" << endl << "2) Нет" << endl;

choice = inputNumber(1, 2);

if (choice == 1) { new\_student.marks.pe = 1; }

else { new\_student.marks.pe = 0; }

cout << "Студент сдал зачет по дисциплине 'История'?" << endl << "1) Да" << endl << "2) Нет" << endl;

choice = inputNumber(1, 2);

if (choice == 1) { new\_student.marks.hist = 1; }

else { new\_student.marks.hist = 0; }

cout << "Студент сдал зачет по дисциплине 'Английский язык'?" << endl << "1) Да" << endl << "2) Нет" << endl;

choice = inputNumber(1, 2);

if (choice == 1) { new\_student.marks.english = 1; }

else { new\_student.marks.english = 0; }

cout << "Студент сдал зачет по дисциплине 'ОКП'?" << endl << "1) Да" << endl << "2) Не смей сюда нажимать! Должно быть стыдно... " << endl << endl;

choice = inputNumber(1, 2);

if (choice == 1) { new\_student.marks.okp = 1; }

else {

new\_student.marks.okp = 0;

cout << "Это никчемный студент, довольно бесполезной информации!" << endl;

system("pause");

}

editAddedStudent(new\_student);

v.push\_back(new\_student);

}

bool otlichnik(Student a) {

if (((a.marks.os == 9) || (a.marks.os == 10)) &&

((a.marks.math == 9) || (a.marks.math == 10)) &&

((a.marks.oaip == 9) || (a.marks.oaip == 10)) &&

((a.marks.fil == 9) || (a.marks.fil == 10))) {

return true;

}

else return false;

}

void defaultBenefitChange(vector <Student>& v) {

cout << "Введите размер базовой стипендии: ";

int benefit = inputNumber(1, 500);

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

v[i].benefits = benefit;

if (v[i].marks.average <= 5) v[i].benefits = 0;

if (otlichnik(v[i]) && v[i].active == 1) v[i].benefits = 1.5 \* benefit;

if (otlichnik(v[i]) && v[i].active == 0) v[i].benefits = 1.25 \* benefit;

}

}

void deleteStudent(vector <Student>& v) {

cout << "Введите номер удаляемое записи: ";

int index = inputNumber(1, v.size()) - 1;

system("cls");

cout << "Вы действительно хотите удалить студента " << v[index].surname << " из системы?" << endl << "1) Да" << endl << "2) Отменить" << endl;

int choice = inputNumber(1, 2);

if (choice == 1) v.erase(v.begin() + index);

system("cls");

}

void readFileStudents(vector <Student>& v) {

ifstream fin(FILE\_OF\_DATA, ios::in); // Открыли файл для чтения

if (!fin.is\_open()) {

cout << "На данный момент не существует ни одного студента в системе." << endl << "Пожалуйста, добавьте нового студента!" << endl;

addStudent(v);

}

else

{

int i = 0;

while (!fin.eof())

{

if (i < v.size())

{

fin >> v[i].surname

>> v[i].group\_number

>> v[i].active

>> v[i].benefits

>> v[i].marks.average

>> v[i].marks.os

>> v[i].marks.math

>> v[i].marks.oaip

>> v[i].marks.fil

>> v[i].marks.hist

>> v[i].marks.pe

>> v[i].marks.ikg

>> v[i].marks.okp

>> v[i].marks.english;

i++;

}

else

{

break;

}

}

}

fin.close(); //Закрыли файл

}

void writeFileStudents(vector <Student>& v) {

ofstream fout(FILE\_OF\_DATA, ios::out); // Открыли файл для записи

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

{

fout << v[i].surname << " "

<< v[i].group\_number << " "

<< v[i].active << " "

<< v[i].benefits << " "

<< v[i].marks.average << " "

<< v[i].marks.os << " "

<< v[i].marks.math << " "

<< v[i].marks.oaip << " "

<< v[i].marks.fil << " "

<< v[i].marks.hist << " "

<< v[i].marks.pe << " "

<< v[i].marks.ikg << " "

<< v[i].marks.okp << " "

<< v[i].marks.english;

if (i < v.size() - 1)

{

fout << endl;

}

}

fout.close();

}

bool mySortByBenefit(Student a, Student b) {

return a.benefits > b.benefits;

}

void sortByBenefit(vector <Student>& v) {

system("cls");

sort(v.begin(), v.end(), mySortByBenefit);

}

bool mySortByActive(Student a, Student b) {

return a.active > b.active;

}

void sortByActive(vector <Student>& v) {

system("cls");

sort(v.begin(), v.end(), mySortByActive);

}

bool mySortByAverageMark(Student a, Student b) {

return a.marks.average > b.marks.average;

}

void sortByAverageMark(vector <Student>& v) {

system("cls");

sort(v.begin(), v.end(), mySortByAverageMark);

}

bool mySortBySurname(Student a, Student b) {

return a.surname < b.surname;

}

void sortBySurname(vector <Student>& v) {

system("cls");

sort(v.begin(), v.end(), mySortBySurname);

}