**Московский авиационный институт (национальный исследовательский университет)**

Институт №3. Факультет «Системы управления, информатика и электроэнергетика». Кафедра №304

**Пояснительная записка к курсовой работе**

по дисциплине «Программирование на языке высокого уровня»

«Разработка средствами С++ информационной системы поддержки работы деканата института»

Выполнил студент группы М30-207Б-18

Кривонос А. А. \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Принял к.т.н., доцент

Офицеров В.П.\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Москва 2019

**Задание на курсовую работу по дисциплине**

«Программирование на языке высокого уровня»

студенту группы М30-207Б-18

Кривоносу Александру Александровичу

Тема: «Разработка средствами С++ информационной системы поддержки работы деканата института»

Задание:

1. Сформулировать функциональные и нефункциональные требования к информационной файловой системе.
2. Спроектировать структуры данных для заданной предметной области.
3. Реализовать информационную систему средствами С++.

Решаемые задачи:

• Справочные данные - список студентов (ФИО, группа, адрес и т.п.)

• Расписание занятий (предмет, преподаватель, группа и т.п.)

• Контроль посещаемости занятий

Срок начала: 2 сентября 2019 г.

Срок окончания: 15 декабря 2019 г.

Руководитель:

доцент, к.т.н. Офицеров Владимир Петрович

(Ф.И.О) (подпись)

Исполнитель:

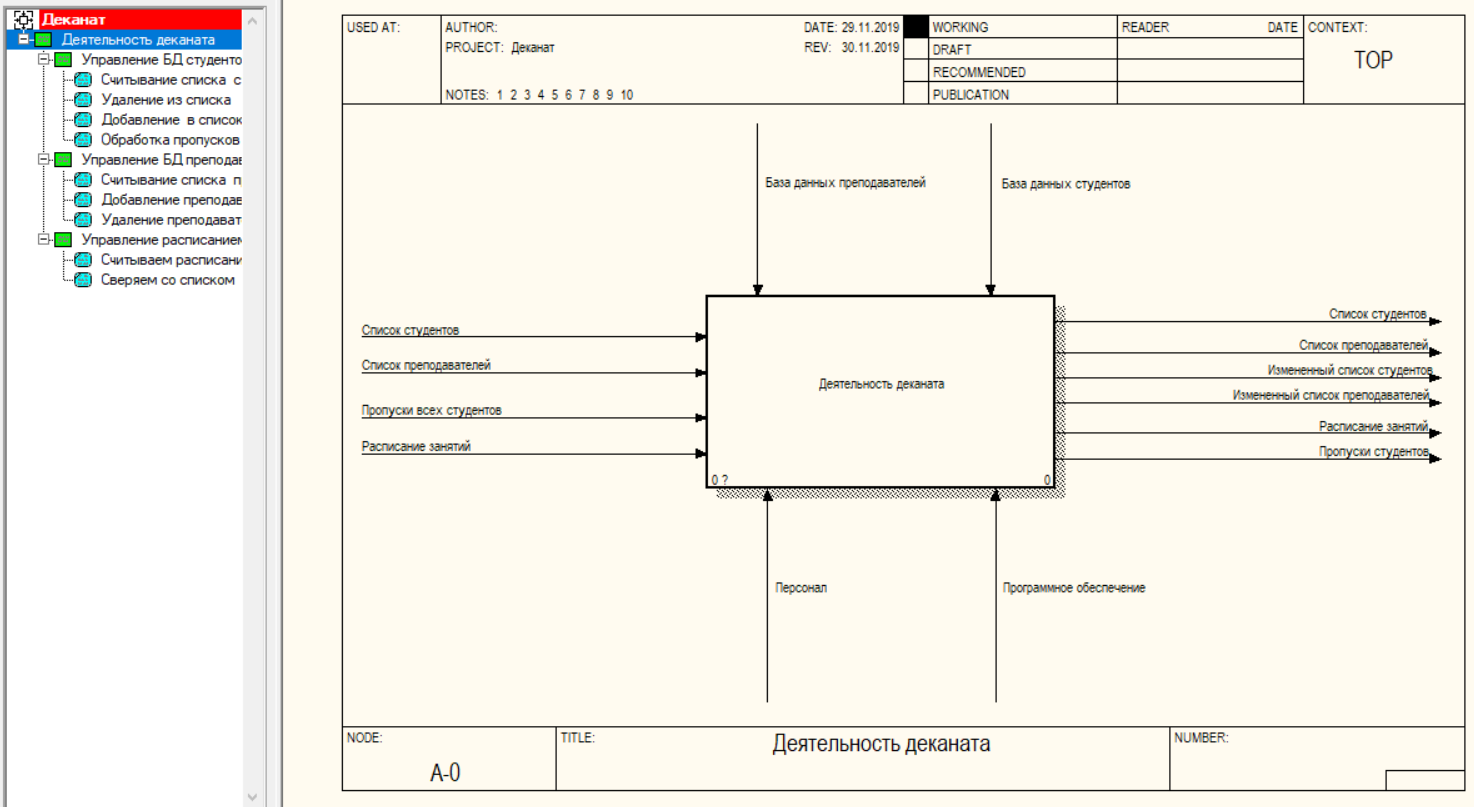
студент группы M3О-207Б Кривонос Александр Александрович

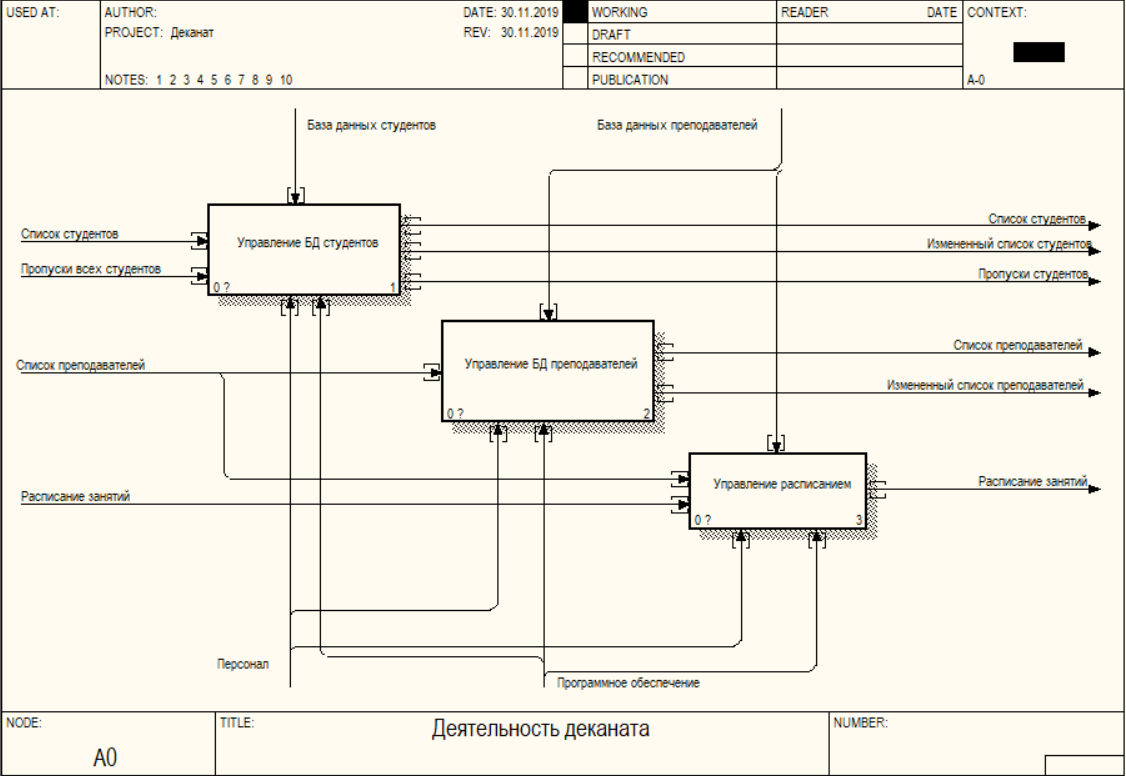
(Ф.И.О) (подпись)

**Содержание**

1. Определение функциональных требований к системе……………………...3
   1. Моделирование среды функционирования деканата…………………..3
   2. Моделирование функций деканата…………………………………….....4
   3. Формирование функциональных требований…………………………...5
2. Разработка технического задания…………………………………………….6
3. Проектирование информационной системы поддержки работы деканата..8
   1. Модель среды функционирования информационной системы………..8
   2. Моделирование работы информационной системы……………………9
   3. Реализация системы информационной поддержки работы деканата...10
      1. Фрагменты текста программы реализующие основные функциональные требования……………………………………….11
      2. Примеры работы системы…………………………………………15
4. Выводы………………………………………………………………………..17

Приложение А. Исходные коды информационной системы поддержки работы деканата………………………………………………………………18**1. Определение функциональных требований к системе**

**1.1. Моделирование среды функционирования деканата**

**1.2. Моделирование функций деканата**

**1.3. Формирование функциональных требований**

Данная система должна иметь следующие характеристики:

* Файл с данными для авторизации должен иметь формат “.bin” и быть организован следующим образом: “логин пароль”
* Файл с информацией о студентах должен иметь формат “.txt” и иметь следующие данные:

“Фамилия Имя Отчество Год поступления Группа Телефон Домашний адрес”

* Файл с информацией о преподавателях должен иметь формат “.txt” и иметь следующие данные:

“Фамилия Имя Отчество Телефон”

* Файл с информацией о расписании должен иметь формат “.txt” и иметь следующие данные:

“Номер учебной недели

Дата

Предмет Время проведения Номер аудитории ФИО преподавателя”

* Файл с информацией о пропусках студентов должен иметь формат “.txt” и быть организован следующим образом:

“ФИО студента

Дата(когда отсутствовал) Предмет(на котором отсутствовал)”

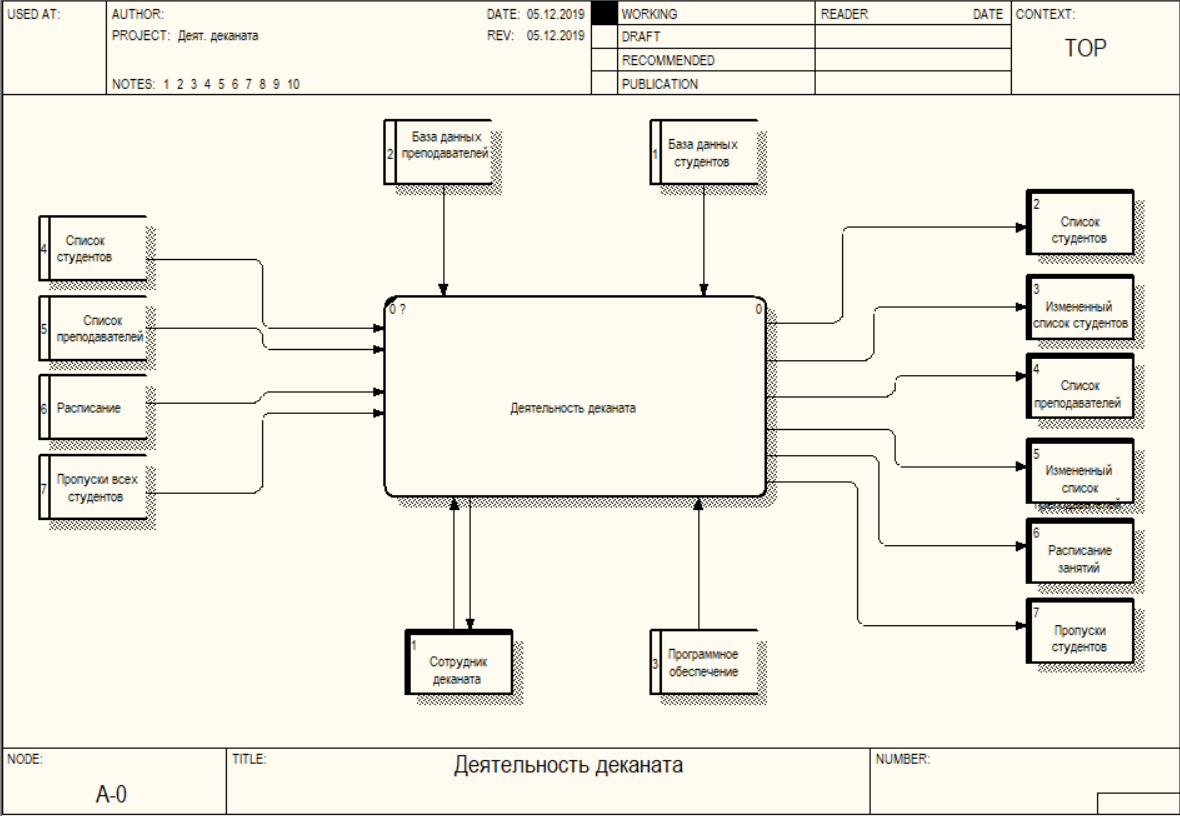
**2. Разработка технического задания**

Разработать информационную систему поддержки деятельности деканата с помощью средств языка “C” реализующий следующие функциональные и нефункциональные требования:

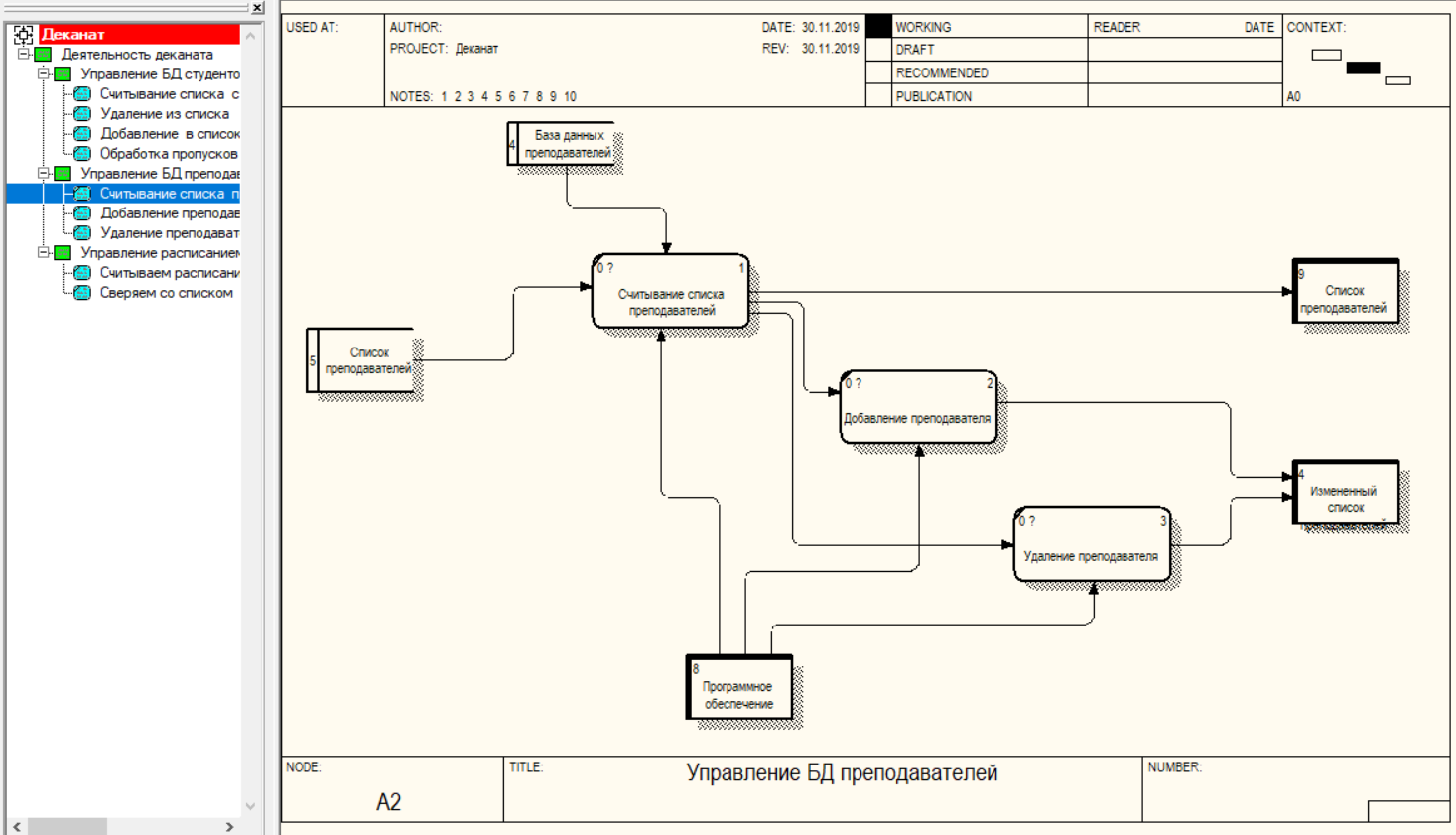
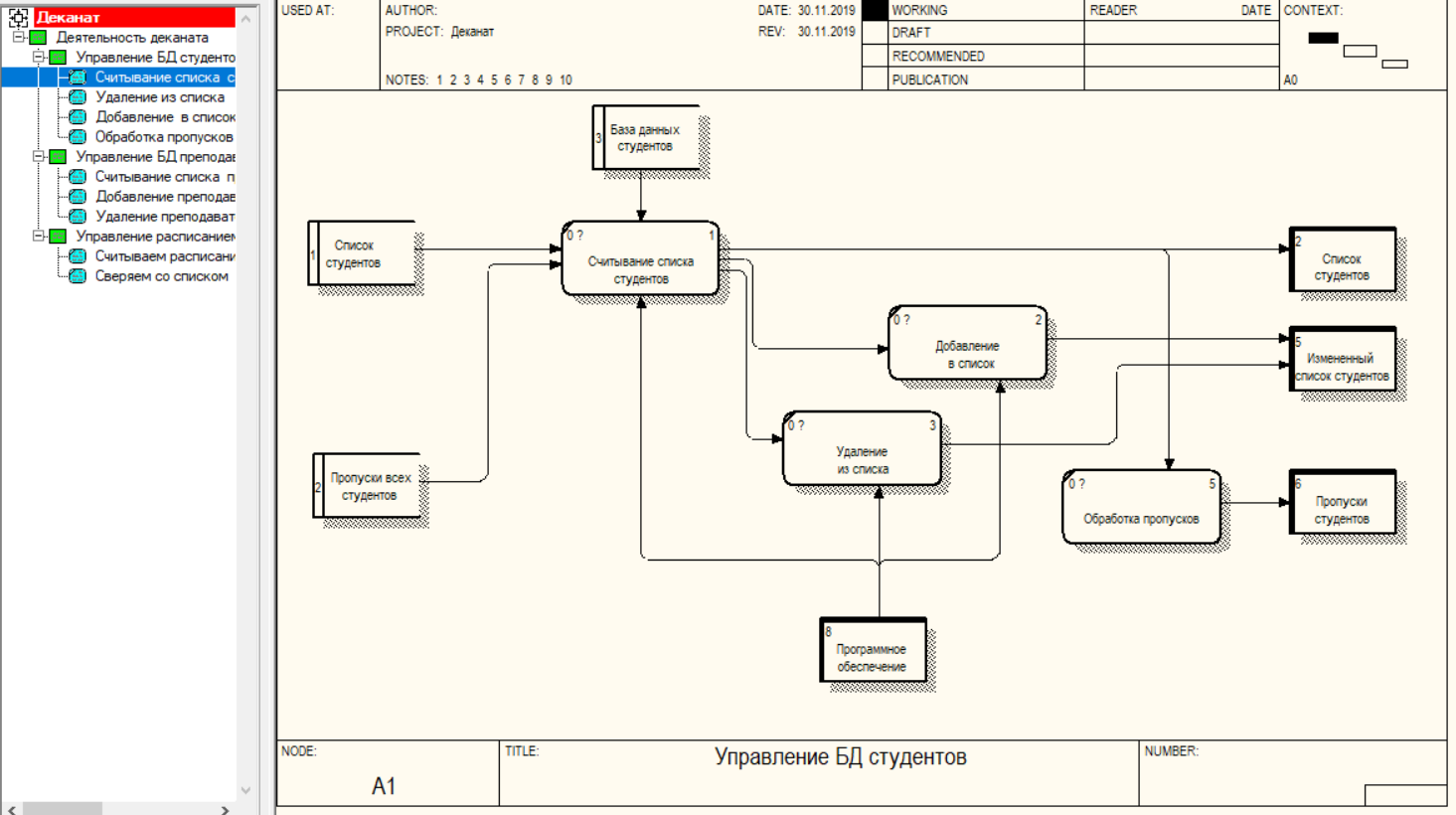
|  |  |
| --- | --- |
| Функциональные | Нефункциональные |
| Реализовать несколько профилей с различным функционалом (пользователь «представитель деканата», пользователь «преподаватель», пользователь «студент») | Сравнить введенные в консоль логин и пароль с данными для авторизации из файла и осуществить вход под нужным пользователем |
| Вывод общего списка студентов | Считывание из файла данных о студенте, инициализация массива структур студентов и вывод на экран консоли этой информации |
| Добавление студента в список | Увеличение размера массива структур, считывание введенных в консоль данных о студенте, добавление нового студента |
| Удаление из списка студента по ФИО | Поиск по массиву структур студента с введенными ФИО и удаление из массива структур |
| Вывод общего списка преподавателей | Считывание из файла данных о преподавателей, инициализация массива структур преподавателей и вывод на экран консоли этой информации |
| Добавление преподавателей в список | Увеличение размера массива структур, считывание введенных в консоль данных о преподавателе, добавление нового преподавателя |
| Удаление из списка преподавателя по ФИО | Поиск по массиву структур преподавателя с введенными ФИО и удаление из массива структур |
| Вывод расписания для студентов: всего доступного, на неделю, на определенную дату | Считывание информации о расписании из файла, инициализация массива структур расписания:  1) вывод всего массива структур на экран консоли  2) поиск требуемой недели в общем списке и вывод расписания этой недели на экран консоли  3) поиск введенной даты и вывод расписания на этот день на экран консоли |
| Вывод расписания для преподавателя | Поиск по массиву структур расписания требуемого преподавателя, вывод на кран консоли информации о проводимом предмете |
| Вывод пропусков студентов | Поиск по массиву структур студентов требуемого студента, поиск по массиву структур пропусков требуемого студента, вывод на экран консоли информации о прогулах данного студента |

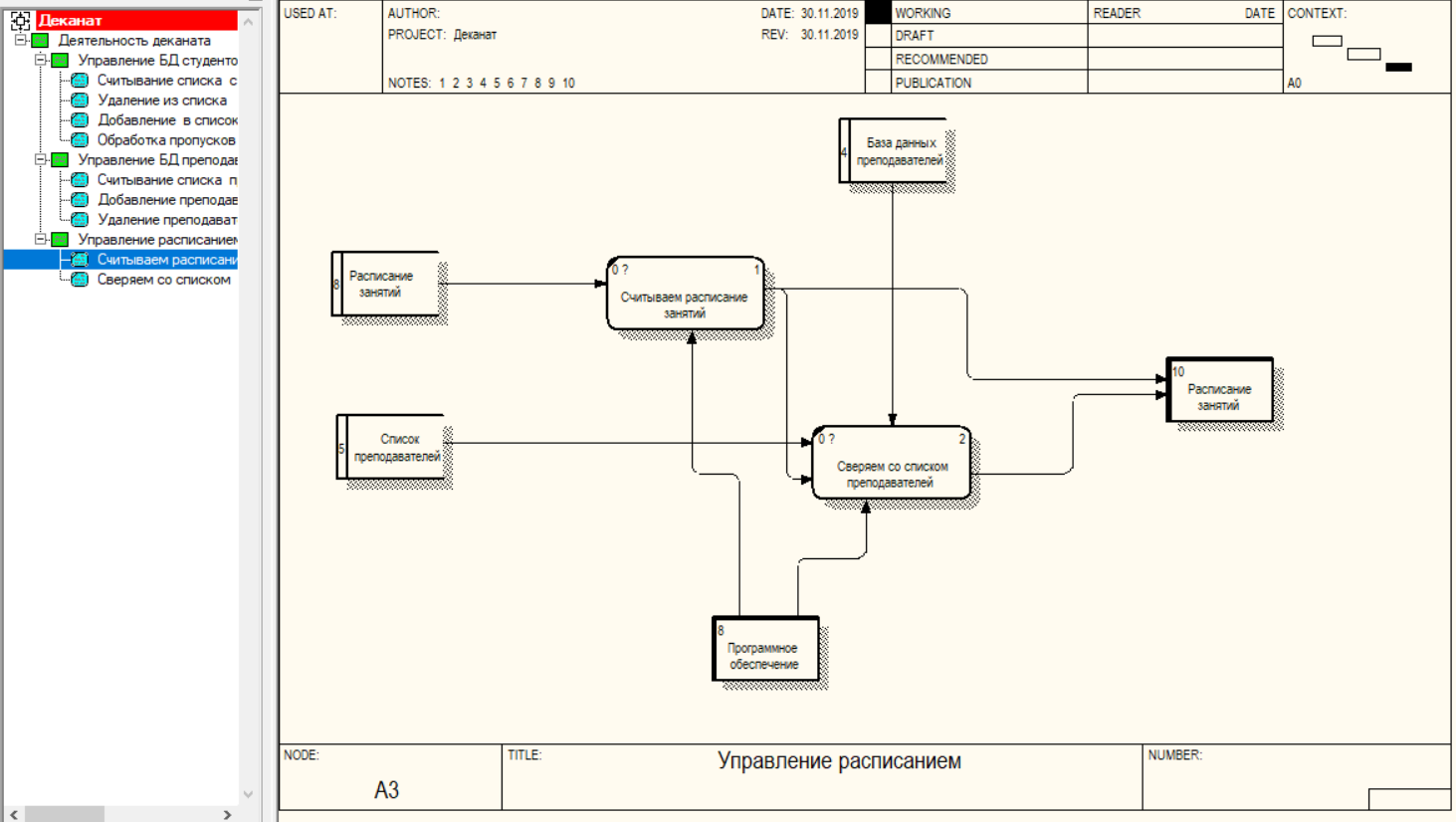
**3**. **Проектирование информационной системы поддержки работы деканата**

* 1. **Модель среды функционирования информационной системы**

****

* 1. **Моделирование работы информационной системы**





* 1. **Реализация системы информационной поддержки работы деканата**
     1. **Фрагменты текста программы реализующие основные функциональные требования**

*Вывод списка студентов*

void getStudents(int &strCountS, studentInf \*student) {

/\*печать поэлементно структуры студент\*/

for (int f = 0; f < strCountS; f++) {

printf("%s\t%s\t%s\t%s\t%s\t%s\t%s", &student[f].surname, &student[f].name, &student[f].secondname, &student[f].enterYear, &student[f].group, &student[f].phone, &student[f].adress);

}

printf("\n");

}

*Вывод списка преподавателей*

void getTeachers(int &strCountT, teacherInf \*teacher) {

/\*печать поэлементно структуры преподаватель\*/

for (int f = 0; f < strCountT; f++) {

printf("%s\t%s\t%s\t%s\n", &teacher[f].tSurname, &teacher[f].tName, &teacher[f].tSecondname, &teacher[f].tPhone);

}

}

*Вывод расписания на определенную дату*

void getSchedule(int &strCountSc, scheduleInf \*schedule, int \*weekIndex, int \*dateIndex, int &sizeWeekIndex, int &sizeDateIndex) {

int f = -1; //переменная для выбора действия

int i = 0; //индексная переменная

int j = 0; //индексная переменная

int k = 0; //индексная переменная

int y = -1; //индексная переменная

int num = -1; //номер недели

bool weekExist = false; //флаг существования недели

char dateWritten[256]; //дата

bool dateExist = false; //флаг существования даты

weekIndex[sizeWeekIndex] = 2 \* weekIndex[sizeWeekIndex - 1] - weekIndex[sizeWeekIndex - 2]; //расчет следующего индекса недель

printf("Выберите дальнейшее действие:\n1. Вывести все доступное расписание\n2. Вывести расписание определенной недели\n3. Вывести расписание на определенную дату\n");

scanf("%d", &f);

if (f == 1) {

//печать всего доступного расписания

for (i = 0; i < strCountSc; i++) {

if (i == weekIndex[k]) {

printf("%s %d\n", schedule[i].week, schedule[i].numWeek);

k++;

}

else if (i == dateIndex[j]) {

printf("%s\n", schedule[i].date);

j++;

}

else if (i != dateIndex[j]) {

printf("%s %s %s %s %s %s\n", schedule[i].subject, schedule[i].classTime, schedule[i].numOfAudience, schedule[i].teacherN, schedule[i].teacherS, schedule[i].teacherSN);

}

}

}

if (f == 2) {

printf("Введите номер недели: ");

scanf("%d", &num);

/\*проверка на существование введенной недели\*/

for (y = 0; y < sizeWeekIndex; y++) {

if (num == schedule[weekIndex[y]].numWeek) { //если номер введенной недели совпадает с каким-то номером недели массива индексов недель

weekExist = true;

break;

}

else if ((y == sizeWeekIndex - 1) && (weekExist == false)) {

system("cls");

printf("Выбранная вами неделя недоступна.\n");

break;

}

}

if (weekExist == true) {

//печать расписания на выбранную неделю

for (i = weekIndex[y]; i < weekIndex[y + 1]; i++) {

if (i == weekIndex[y]) {

printf("%s %d\n", schedule[i].week, schedule[i].numWeek);

}

else if (i == dateIndex[(num - 1) \* 5 + j]) {

printf("%s\n", schedule[i].date);

j++;

}

else {

printf("%s %s %s %s %s %s\n", schedule[i].subject, schedule[i].classTime, schedule[i].numOfAudience, schedule[i].teacherN, schedule[i].teacherS, schedule[i].teacherSN);

}

}

}

}

if (f == 3) {

printf("Введите дату (в формате dd:mm:yy): ");

scanf("%s", &dateWritten);

/\*проверка на существование выбранной даты\*/

if (strcmp(dateWritten, schedule[dateIndex[sizeDateIndex - 1]].date) == 0) {

for (y = 0; y < sizeDateIndex; y++) {

if (strcmp(dateWritten, schedule[dateIndex[y]].date) == 0) {

dateExist = true;

break;

}

else if ((y == sizeDateIndex - 1) && (dateExist == false)) {

system("cls");

printf("Выбранная вами дата недоступна.\n");

}

}

if (dateExist == true) {

//печать расписания на выбранную дату

for (i = dateIndex[y]; i < strCountSc; i++) {

if (i == dateIndex[y]) {

printf("%s\n", schedule[i].date);

}

else {

printf("%s %s %s %s %s %s\n", schedule[i].subject, schedule[i].classTime, schedule[i].numOfAudience, schedule[i].teacherN, schedule[i].teacherS, schedule[i].teacherSN);

}

}

}

}

else {

//расчет последнего индекса массива дат

dateIndex[sizeDateIndex] = 2 \* dateIndex[sizeDateIndex - 1] - dateIndex[sizeDateIndex - 2];

/\*проверка на существование выбранной даты\*/

for (y = 0; y < sizeDateIndex; y++) {

if (strcmp(dateWritten, schedule[dateIndex[y]].date) == 0) {

dateExist = true;

break;

}

else if ((y == sizeDateIndex - 1) && (dateExist == false)) {

system("cls");

printf("Выбранная вами дата недоступна.\n");

}

}

if (dateExist == true) {

//печать расписания на выбранную дату

for (i = dateIndex[y]; i < dateIndex[y + 1]; i++) {

if (i == dateIndex[y]) {

printf("%s\n", schedule[i].date);

}

else {

printf("%s %s %s %s %s %s\n", schedule[i].subject, schedule[i].classTime, schedule[i].numOfAudience, schedule[i].teacherN, schedule[i].teacherS, schedule[i].teacherSN);

}

}

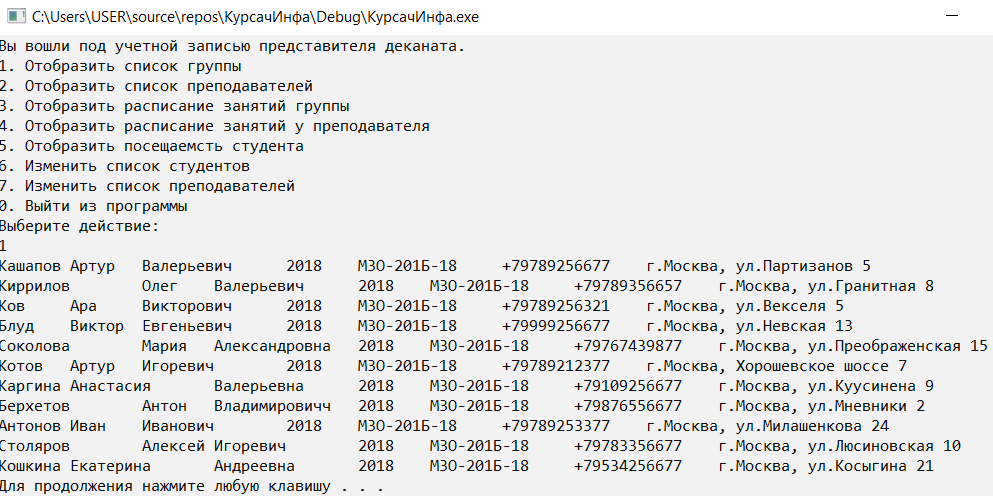
}

}

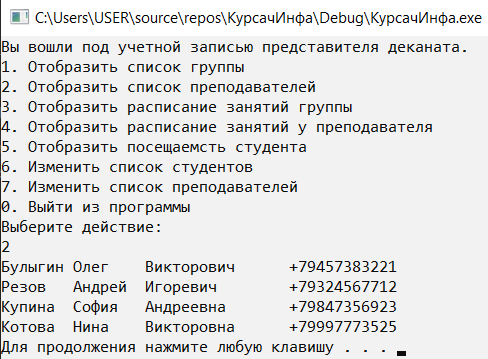
}

}

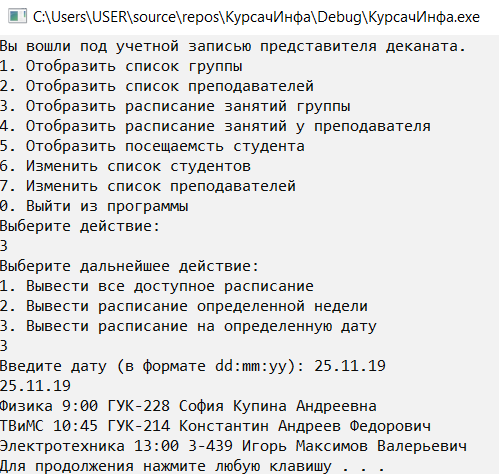
* + 1. **Примеры работы системы**

*Вывод списка студентов*

*Вывод списка преподавателей*



*Вывод расписания на определенную дату*



**Приложение А**

**Код программы**

#define \_CRT\_SECURE\_NO\_WARNINGS

#include <stdio.h>

#include <stdlib.h>

#include <string.h>

#include<locale.h>

#include<conio.h>

#include <Windows.h>

using namespace std;

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

/\*\*\*\*\*\*\*\*\*\*\*СТРУКТУРЫ\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

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

struct studentInf { //Структура студентов

char name[25]; //Имя студента

char surname[25]; //Фамилия студента

char secondname[25]; //Отчество студента

char enterYear[5]; //Год поступления

char group[12]; //Группа

char adress[50]; //Адрес

char phone[13]; //Телефон

};

struct teacherInf { //сСруктура преподавателей

char tName[25]; //Имя преподавателя

char tSurname[25]; //Фамилия преподавателя

char tSecondname[25]; //Отчество преподавателя

char tPhone[13]; //телефон преподавателя

};

struct scheduleInf { //Структура расписания

int numWeek; //Номер недели

char week[10]; //"Неделя"

char date[9]; //Дата

char subject[15]; //Предмет

char classTime[6]; //Время предмета

char numOfAudience[10]; //Номер аудитории

char teacherN[25]; //Имя преподавателя

char teacherS[25]; //Фамилия преподавателя

char teacherSN[25]; //Отчество преподавателя

};

struct authorizationData { //Структура пользователя

char login[25]; //Логин

char pasword[25]; //Пароль

};

struct absentInf { //Структура отсутствующих

char sName[256]; //Имя отсутствующего

char sSurName[256]; //Фамилия отсутствующего

char sSecondName[256]; //Отчество отсутствующего

char absentDate[10]; //Дата, когда отсутствовал

char abSubject[15]; //Предмет, на котором отсутствовал

};

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

/\*\*\*\*\*ПРОТОТИПЫ ФУНКЦИЙ\*\*\*\*\*\*\*\*/

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

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*Функция инициализации структур\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

int initialization(FILE \*students, int &strCountS, studentInf \*student,

FILE \*teachers, int &strCountT, teacherInf \*teacher,

FILE \*Schedule, int &strCountSc, scheduleInf \*schedule,

int \*weekIndex, int \*dateIndex, int &sizeWeekIndex, int &sizeDateIndex,

int &strCountAb, absentInf \*absent, FILE \*Absent, int \*absentIndex, int & sizeAbsentIndex);

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*Функция вывода списка студентов\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

void getStudents(int &strCountS, studentInf \*student);

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*Функция добавления студента\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

int AddStudent(FILE \*students, int &strCountS, studentInf \*student);

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*Функция удаления студента\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

int DeleteStudent(FILE \*students, int &strCountS, studentInf \*student);

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*Функция вывода списка преподавателей\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

void getTeachers(int &strCountT, teacherInf \*teacher);

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*Функция добавления преподавателя\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

int AddTeacher(FILE \*teachers, int &strCountT, teacherInf \*teacher);

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*Функция удаления преподавателя\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

int DeleteTeacher(FILE \*teachers, int &strCountT, teacherInf \*teacher);

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*Функция вывода расписания группы\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

void getSchedule(int &strCountSc, scheduleInf \*schedule, int \*weekIndex, int \*dateIndex, int &sizeWeekIndex, int &sizeDateIndex);

/\*\*\*\*\*\*\*\*\*\*Функция вывода расписания для преподавателя\*\*\*\*\*\*\*\*\*\*\*\*\*/

void GetScheduleForT(int &strCountSc, scheduleInf \*schedule, int \*weekIndex, int \*dateIndex, int &sizeWeekIndex, int &sizeDateIndex);

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*Функция вывода отсутствующих\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

void getAbsent(int &strCountAb, absentInf \*absent, int &strCountS, studentInf \*student, int \*absentIndex, int &sizeAbsentIndex);

int main() {

setlocale(LC\_ALL, "Russian"); //подключение русского языка

SetConsoleCP(1251); //изменение кодировки

SetConsoleOutputCP(1251); //изменение кодировки консоли

FILE \*authorization = fopen("authorization.bin", "rb"); //файл с данными авторизации

int i = -1; //переменная для определения аккаунта

int l = -1; //переменная для завершения программы

int h = -1; //переменная для выбора действия при изменении списка студентов

int p = -1; //переменная для выбора действия при изменении списка преподавателей

bool signedIn = false; //переменная подтверждения входа в аккаунт

/\*Для студента\*/

FILE \*students = NULL; //файл с данными студентов

int strCountS = 0; //количество строк в файле студентов

studentInf student[256]; //массив структур студентов

/\*Для учителя\*/

FILE \*teachers = NULL; //файл с данными преподавателейint strCountT = 0;

int strCountT = 0; //количество строк в файле преподавателей

teacherInf teacher[256]; //массив структур преподавателей

/\*Для расписания\*/

FILE \*Schedule = NULL; //файл с информацией о преподавателях

int strCountSc = 0; //количество строк в файле расписания

scheduleInf schedule[256]; //массив структур расписания

int dateIndex[256]; //массив индексов дат в файле расписания

int weekIndex[256]; //массив индексов недель в файле расписания

int sizeWeekIndex = 0; //индекс для массив индексов недель

int sizeDateIndex = 0; //индекс для массив индексов дат

/\*Для отсутствующих\*/

FILE \*Absent = NULL; //файл с информацией о пропусках

absentInf absent[256]; //массив структур отсутствующих

int strCountAb = 0; //количество строк в файле структур

int absentIndex[256]; //массив индексов ФИО отсутствующих

int sizeAbsentIndex = 0; //индекс для массива индексов ФИО отсутствующих

/\*Для авторизации\*/

authorizationData signIn[3]; //массив структур авторизационных данных

char log[256]; //логин

char pas[256]; //пароль

int f = 0; //индекс для логина в массиве структур авторизации

char c; //переменная для считывания пробела

for (int i = 0; i < 3; i++) {

while (1) { //бесконечный цикл

fread(&signIn[i].login[f], sizeof(char), 1, authorization); //считываем побайтно логин

if (signIn[i].login[f] == ' ') { //если символ - пробел, то прервать считывание логина

signIn[i].login[f] = '\0';

f = 0;

break;

}

else

f++; //иначе увеличиваем индекс для логина

}

while (1) { //бесконечный цикл

fread(&signIn[i].pasword[f], sizeof(char), 1, authorization); //считываем побайтно пароль

if ((signIn[i].pasword[f] == '\r') || (signIn[i].pasword[f] == '\n')) { //если символ - "возврат каретки" или "новая строка", то прервать считывание логина

signIn[i].pasword[f] = '\0';

f = 0;

break;

}

else

f++; //иначе увеличиваем индекс для пароля

}

fread(&c, sizeof(char), 1, authorization); //считываем пробел между логином и паролем

}

printf("Войдите в учетную запись.\nВведите логин: ");

scanf("%s", &log); //считываем введенный логин

printf("Введите пароль: ");

scanf("%s", &pas); //считываем введенный пароль

for (i = 0; i < 3; i++) {

if ((strcmp(signIn[i].login, log) == 0) && ((strcmp(signIn[i].pasword, pas)) == 0)) { //находим совпадающий аккаунт

signedIn = true;

break;

}

}

system("cls"); //очищаем экран

if (signedIn == true) { //если пользователь авторизовался

while (l != 0) {

switch (i) {

case 0:

initialization(students, strCountS, student, //инициализация структур

teachers, strCountT, teacher, Schedule, strCountSc, schedule, weekIndex, dateIndex, sizeWeekIndex, sizeDateIndex,

strCountAb, absent, Absent, absentIndex, sizeAbsentIndex);

printf("Вы вошли под учетной записью представителя деканата.\n\

1. Отобразить список группы\n\

2. Отобразить список преподавателей\n\

3. Отобразить расписание занятий группы\n\

4. Отобразить расписание занятий у преподавателя\n\

5. Отобразить посещаемсть студента\n\

6. Изменить список студентов\n\

7. Изменить список преподавателей\n\

0. Выйти из программы\n");

printf("Выберите действие: \n");

scanf("%d", &l); //считываем введенное число

switch (l) {

case 1:

getStudents(strCountS, student); //вывести список студентов

break;

case 2:

getTeachers(strCountT, teacher); //вывести список преподавателей

break;

case 3:

getSchedule(strCountSc, schedule, weekIndex, dateIndex, sizeWeekIndex, sizeDateIndex); //вывести расписание для группы

break;

case 4:

GetScheduleForT(strCountSc, schedule, weekIndex, dateIndex, sizeWeekIndex, sizeDateIndex); //вывести расписание для преподавателя

break;

case 5:

getAbsent(strCountAb, absent, strCountS, student, absentIndex, sizeAbsentIndex); //вывести прогулы студента

break;

case 6:

system("cls");

printf("Выберите дальнейшее действие:\n1. Добавить студента\n2. Удалить студента\n");

scanf("%d", &h);

switch (h) {

case 1:

AddStudent(students, strCountS, student); //добавить студента

break;

case 2:

DeleteStudent(students, strCountS, student); //удалить студента

break;

default:

printf("\nВыбранной функции не существует!\n");

break;

}

break;

case 7:

system("cls");

printf("Выберите дальнейшее действие:\n1. Добавить преподавателя\n2. Удалить преподавателя\n");

scanf("%d", &p);

switch (p) {

case 1:

AddTeacher(teachers, strCountT, teacher); //добавить преподавателя

break;

case 2:

DeleteTeacher(teachers, strCountT, teacher); //удалить преподавателя

break;

default:

printf("\nВыбранной функции не существует!\n");

break;

}

break;

case 0:

break;

default:

printf("\nВыбранной функции не существует!\n");

break;

}

break;

case 1:

initialization(students, strCountS, student, //инициализация структур

teachers, strCountT, teacher, Schedule, strCountSc, schedule,

weekIndex, dateIndex, sizeWeekIndex, sizeDateIndex,

strCountAb, absent, Absent, absentIndex, sizeAbsentIndex);

printf("Вы вошли под учетной записью преподавателя.\n\

Выберите дальнейшее действие:\n1. Отобразить список группы\n\

2. Отобразить список преподавателей\n\

3. Отобразить расписание занятий у преподавателя\n\

0. Выйти из программы\n");

scanf("%d", &l);

switch (l) {

case 1:

getStudents(strCountS, student); //вывести список студентов

break;

case 2:

getTeachers(strCountT, teacher); //вывести список преподавателей

break;

case 3:

GetScheduleForT(strCountSc, schedule, weekIndex, dateIndex, sizeWeekIndex, sizeDateIndex); //вывести расписание для преподавателя

break;

case 0:

break;

default:

printf("\nВыбранной функции не существует!\n");

break;

}

break;

case 2:

initialization(students, strCountS, student, //инициализация структур

teachers, strCountT, teacher, Schedule, strCountSc, schedule,

weekIndex, dateIndex, sizeWeekIndex, sizeDateIndex,

strCountAb, absent, Absent, absentIndex, sizeAbsentIndex);

printf("Вы вошли под учетной записью студента.\n\

Выберите дальнейшее действие:\n1. Отобразить список группы\n\

2. Отобразить расписание занятий группы\n\

3. Отобразить посещаемсть студента\n\

0. Выйти из программы\n");

scanf("%d", &l);

switch (l) {

case 1:

getStudents(strCountS, student); //вывести список студентов

break;

case 2:

getSchedule(strCountSc, schedule, weekIndex, dateIndex, sizeWeekIndex, sizeDateIndex); //вывести расписание для группы

break;

case 3:

getAbsent(strCountAb, absent, strCountS, student, absentIndex, sizeAbsentIndex); //вывести прогулы студента

break;

case 0:

break;

default:

printf("\nВыбранной функции не существует!\n");

break;

}

break;

}

/\*Очищение памяти отведенной для переменных\*/

memset(student, 0, sizeof(student));

memset(teacher, 0, sizeof(teacher));

memset(schedule, 0, sizeof(schedule));

memset(absent, 0, sizeof(absent));

strCountS = 0;

strCountT = 0;

strCountSc = 0;

strCountAb = 0;

memset(absentIndex, 0, sizeof(absentIndex));

sizeAbsentIndex = 0;

memset(dateIndex, 0, sizeof(dateIndex));

memset(weekIndex, 0, sizeof(weekIndex));

sizeWeekIndex = 0;

sizeDateIndex = 0;

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

if (l != 0) { //условие завершения программы

system("pause");

system("cls");

}

}

}

else

printf("Ошибка авторизации.\n");

memset(authorization, 0, sizeof(authorization));

system("pause");

return 0;

}

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

/\*\*\*\*\*РЕАЛИЗАЦИЯ ФУНКЦИЙ\*\*\*\*\*\*\*\*/

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

int initialization(FILE \*students, int &strCountS, studentInf \*student,

FILE \*teachers, int &strCountT, teacherInf \*teacher,

FILE \*Schedule, int &strCountSc, scheduleInf \*schedule,

int \*weekIndex, int \*dateIndex, int &sizeWeekIndex, int &sizeDateIndex,

int &strCountAb, absentInf \*absent, FILE \*Absent, int \*absentIndex, int & sizeAbsentIndex) {

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*Инициализация структуры student\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

students = fopen("Students.txt", "rt"); //открываем файл

char \*s = (char\*)malloc(256); //строка для считывания всего файла

int i; //индексная переменная

/\*проверяем, существует ли файл\*/

if (students == NULL) {

printf("Ошибка открытия файла!");

system("pause");

return 1;

}

/\*проверяем, пуст ли файл\*/

fseek(students, 0, SEEK\_END); //переводим указатель в конец

long posS = ftell(students); //запоминаем эту позицию

if (posS <= 0) {

printf("Файл пуст!");

system("pause");

return 2;

}

rewind(students); //переводим указатель в начало

while (fgets(s, 256, students) != NULL) { //считываем файл до конца

if (s[0] != '\n') {

strCountS++; //увеличиваем счетчик строк

}

}

rewind(students);

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*считываем информацию из файла в массив структур\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

for (i = 0; i < strCountS; i++) {

fscanf(students, "%s%s%s%s%s%s", &student[i].surname, &student[i].name, &student[i].secondname, &student[i].enterYear, &student[i].group, &student[i].phone);

int a = fgetc(students);

fgets(student[i].adress, 50, students);

}

fclose(students); //закрываем файл

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*Инициализация структуры teacher\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

teachers = fopen("Teachers.txt", "rt"); //открываем файл

int j; //индексная переменная

char \*p = (char\*)malloc(256); //строка для считывания всего файла

/\*проверяем, существует ли файл\*/

if (teachers == NULL) {

printf("Ошибка открытия файла!");

system("pause");

return 3;

}

/\*проверяем, пуст ли файл\*/

fseek(teachers, 0, SEEK\_END); //переводим указатель в конец

long posT = ftell(teachers); //запоминаем эту позицию

if (posT <= 0) {

printf("Файл пуст!");

system("pause");

return 4;

}

rewind(teachers); //переводим указатель в начало

while (fgets(p, 256, students) != NULL) { //считываем файл до конца

strCountT++; //увеличиваем счетчик строк

}

rewind(teachers);

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*считываем информацию из файла в массив структур\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

for (j = 0; j < strCountT; j++) {

fscanf(teachers, "%s%s%s%s", &teacher[j].tSurname, &teacher[j].tName, &teacher[j].tSecondname, &teacher[j].tPhone);

}

fclose(teachers); //закрываем файл

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*Инициализация структуры schedule\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

Schedule = fopen("Schedule.txt", "rt"); //открываем файл

char \*sc = (char\*)malloc(256); //строка для считывания всего файла

int l = 0; //индексная переменная

int m = 0; //индекс для массива индексов дат

int k = 0; //индекс для массива индексов недель

/\*проверяем, существует ли файл\*/

if (Schedule == NULL) {

printf("Ошибка открытия файла!");

system("pause");

return 5;

}

/\*проверяем, пуст ли файл\*/

fseek(Schedule, 0, SEEK\_END); //переводим указатель в конец

long posSce = ftell(Schedule); //запоминаем эту позицию

if (posSce <= 0) {

printf("Файл пуст!");

system("pause");

return 6;

}

rewind(Schedule); //переводим указатель в начало

while (fgets(sc, 256, Schedule) != NULL) { //считываем файл до конца

if (strstr(sc, "Неделя") != NULL) {

weekIndex[k] = strCountSc; //заполняем массив индексами недель

k++;

}

if (isdigit((unsigned char)sc[0])) {

dateIndex[m] = strCountSc; //заполняем массив индексами дат

m++;

}

strCountSc++; //увеличиваем счетчик строк

}

rewind(Schedule);

sizeWeekIndex = k; //запоминаем размер массива индексов недель

sizeDateIndex = m; //запоминаем размер массива индексов дат

m = 0; //обнуляем индексы

k = 0;

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*считываем информацию из файла в массив структур\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

for (l = 0; l < strCountSc; l++) {

if (l == weekIndex[k]) {

fscanf(Schedule, "%s%d", schedule[l].week, &schedule[l].numWeek);

k++;

}

else if (l == dateIndex[m]) {

fscanf(Schedule, "%s", schedule[l].date);

m++;

}

else if (l != dateIndex[m]) {

fscanf(Schedule, "%s%s%s%s%s%s", schedule[l].subject, schedule[l].classTime, schedule[l].numOfAudience, schedule[l].teacherS, schedule[l].teacherN, schedule[l].teacherSN);

}

}

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*Инициализация структуры Absent\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

Absent = fopen("Absent.txt", "rt"); //открываем файл

int n = 0; //индексная переменная

int b = 0; //индекс для массива индексов отсутствующих

char \*ab = (char\*)malloc(256); //строка для считывания всего файла

/\*проверяем, существует ли файл\*/

if (Absent == NULL) {

printf("Ошибка открытия файла!");

system("pause");

return 7;

}

/\*проверяем, пуст ли файл\*/

fseek(Absent, 0, SEEK\_END); //переводим указатель в конец

long posA = ftell(Absent); //запоминаем эту позицию

if (posA <= 0) {

printf("Файл пуст!\n");

system("pause");

return 8;

}

rewind(Absent); //переводим указатель в начало

while (fgets(ab, 256, Absent) != NULL) { //считываем файл до конца

if (!isdigit((unsigned char)ab[0])) {

absentIndex[b] = strCountAb; //заполняем массив индексами отсутствующих

b++;

}

strCountAb++; //увеличиваем счетчик строк

}

rewind(Absent);

sizeAbsentIndex = b; //запоминаем размер массива индексов отсутствующих

b = 0; //обнуляем индекс

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*считываем информацию из файла в массив структур\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

for (n = 0; n < strCountAb; n++) {

if (n == absentIndex[b]) {

fscanf(Absent, "%s%s%s", &absent[n].sSurName, &absent[n].sName, &absent[n].sSecondName);

b++;

}

else if (n != absentIndex[b]) {

fscanf(Absent, "%s", &absent[n].absentDate);

fgets(absent[n].abSubject, 50, Absent);

}

}

fclose(Absent);

return 9;

}

void getStudents(int &strCountS, studentInf \*student) {

/\*печать поэлементно структуры студент\*/

for (int f = 0; f < strCountS; f++) {

printf("%s\t%s\t%s\t%s\t%s\t%s\t%s", &student[f].surname, &student[f].name, &student[f].secondname, &student[f].enterYear, &student[f].group, &student[f].phone, &student[f].adress);

}

printf("\n");

}

int AddStudent(FILE \*students, int &strCountS, studentInf \*student) {

/\*ввод и считывание данных для добавления\*/

printf("Введите данные студента через пробел(ФИО, год поступления, группа, номер телефона, адрес):\n");

scanf("%s%s%s%s%s%s%c", &student[strCountS].surname, &student[strCountS].name, &student[strCountS].secondname, &student[strCountS].enterYear, &student[strCountS].group, &student[strCountS].phone);

fgets(student[strCountS].adress, 50, stdin);

/\*убираем у последней строки символ "новая строка"\*/

for (int i = 0; i < strlen(student[strCountS].adress); i++) {

if (student[strCountS].adress[i] == '\n') {

student[strCountS].adress[i] = '\0';

break;

}

}

strCountS++; //увеличиваем счетчик строк

students = fopen("Students.txt", "wt"); //открываем файл

/\*записываем в файл\*/

for (int i = 0; i < strCountS; i++) {

if (i != strCountS - 1) {

fprintf(students, "%s %s %s %s %s %s %s", student[i].surname, student[i].name, student[i].secondname, student[i].enterYear, student[i].group, student[i].phone, student[i].adress);

}

if (i == strCountS - 1) {

fprintf(students, "\n%s %s %s %s %s %s %s", student[i].surname, student[i].name, student[i].secondname, student[i].enterYear, student[i].group, student[i].phone, student[i].adress);

}

}

fclose(students); //закрываем файл

system("cls");

return 10;

}

int DeleteStudent(FILE \*students, int &strCountS, studentInf \*student) {

char deleteName[256]; //строка для введенного имени

char deleteSurname[256]; //строка ля введенной фамилии

char deleteSecondName[256]; //строка для введенного отчества

int j = -1; //флаг для найденного студента

bool copy = false; //флаг для копирования студентов в списке

/\*вводим и считываем данные о студенте\*/

printf("Введите ФИО студента, которого хотите удалить из списка(через пробелы):\n");

scanf("%s%s%s", &deleteSurname, &deleteName, &deleteSecondName);

for (int i = 0; i < strCountS; i++) {

if (((strcmp(student[i].surname, deleteSurname)) == 0) && ((strcmp(student[i].name, deleteName)) == 0) && ((strcmp(student[i].secondname, deleteSecondName)) == 0)) {

j = i;

copy = true;

break;

}

else if ((j == -1) && (i == strCountS - 1)) {

printf("Студент не найден.\n");

break;

}

}

if (copy == true) {

/\*копируем структуру студента\*/

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

strcpy(student[i].name, student[i + 1].name);

strcpy(student[i].surname, student[i + 1].surname);

strcpy(student[i].secondname, student[i + 1].secondname);

strcpy(student[i].enterYear, student[i + 1].enterYear);

strcpy(student[i].group, student[i + 1].group);

strcpy(student[i].phone, student[i + 1].phone);

strcpy(student[i].adress, student[i + 1].adress);

}

strCountS--;

/\*убираем у последней строки символ "новая строка"\*/

for (int i = 0; i < strlen(student[strCountS - 1].adress); i++) {

if (student[strCountS - 1].adress[i] == '\n') {

student[strCountS - 1].adress[i] = '\0';

break;

}

}

/\*перезаписываем структуру в файл\*/

students = fopen("Students.txt", "wt");

for (int i = 0; i < strCountS; i++) {

fprintf(students, "%s %s %s %s %s %s %s", student[i].surname, student[i].name, student[i].secondname, &student[i].enterYear, &student[i].group, &student[i].phone, &student[i].adress);

}

fclose(students); //закрываем файл

}

return 11;

}

void getTeachers(int &strCountT, teacherInf \*teacher) {

/\*печать поэлементно структуры преподаватель\*/

for (int f = 0; f < strCountT; f++) {

printf("%s\t%s\t%s\t%s\n", &teacher[f].tSurname, &teacher[f].tName, &teacher[f].tSecondname, &teacher[f].tPhone);

}

}

int AddTeacher(FILE \*teachers, int &strCountT, teacherInf \*teacher) {

/\*ввод и считывание данных о преподавателе\*/

printf("Введите данные преподавателя через пробел(ФИО, номер телефона):\n");

scanf("%s%s%s%s", &teacher[strCountT].tSurname, &teacher[strCountT].tName, &teacher[strCountT].tSecondname, &teacher[strCountT].tPhone);

strCountT++; //увеличиваем счетчик строк

/\*открываем файл и записываем в него информацию\*/

teachers = fopen("Teachers.txt", "wt");

for (int i = 0; i < strCountT; i++) {

fprintf(teachers, "%s %s %s %s\n", &teacher[i].tSurname, &teacher[i].tName, &teacher[i].tSecondname, &teacher[i].tPhone);

}

fclose(teachers); //закрываем файл

system("cls");

return 12;

}

int DeleteTeacher(FILE \*teachers, int &strCountT, teacherInf \*teacher) {

char deleteName[256]; //строка для введенного имени

char deleteSurname[256]; //строка ля введенной фамилии

char deleteSecondName[256]; //строка для введенного отчества

int j = -1; //флаг для найденного студента

bool copy = false; //флаг для копирования студентов в списке

/\*вводим и считываем данные о преподавателе\*/

printf("Введите ФИО преподавателя, которого хотите удалить из списка(через пробелы):\n");

scanf("%s%s%s", &deleteSurname, &deleteName, &deleteSecondName);

for (int i = 0; i < strCountT; i++) {

if (((strcmp(teacher[i].tSurname, deleteSurname)) == 0) && ((strcmp(teacher[i].tName, deleteName)) == 0) && ((strcmp(teacher[i].tSecondname, deleteSecondName)) == 0)) {

j = i;

copy = true;

break;

}

else if ((j == -1) && (i == strCountT - 1)) {

printf("Преподаватель не найден.\n");

system("pause");

break;

}

}

if (copy == true) {

/\*копируем структуру преподавателя\*/

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

strcpy(teacher[i].tSurname, teacher[i + 1].tSurname);

strcpy(teacher[i].tName, teacher[i + 1].tName);

strcpy(teacher[i].tSecondname, teacher[i + 1].tSecondname);

strcpy(teacher[i].tPhone, teacher[i + 1].tPhone);

}

strCountT--;

/\*перезаписываем структуру в файл\*/

teachers = fopen("Teachers.txt", "wt");

for (int i = 0; i < strCountT; i++) {

fprintf(teachers, "%s %s %s %s\n", &teacher[i].tSurname, &teacher[i].tName, &teacher[i].tSecondname, &teacher[i].tPhone);

}

fclose(teachers);

}

system("cls");

return 13;

}

void getSchedule(int &strCountSc, scheduleInf \*schedule, int \*weekIndex, int \*dateIndex, int &sizeWeekIndex, int &sizeDateIndex) {

int f = -1; //переменная для выбора действия

int i = 0; //индексная переменная

int j = 0; //индексная переменная

int k = 0; //индексная переменная

int y = -1; //индексная переменная

int num = -1; //номер недели

bool weekExist = false; //флаг существования недели

char dateWritten[256]; //дата

bool dateExist = false; //флаг существования даты

weekIndex[sizeWeekIndex] = 2 \* weekIndex[sizeWeekIndex - 1] - weekIndex[sizeWeekIndex - 2]; //расчет следующего индекса недель

printf("Выберите дальнейшее действие:\n1. Вывести все доступное расписание\n2. Вывести расписание определенной недели\n3. Вывести расписание на определенную дату\n");

scanf("%d", &f);

if (f == 1) {

//печать всего доступного расписания

for (i = 0; i < strCountSc; i++) {

if (i == weekIndex[k]) {

printf("%s %d\n", schedule[i].week, schedule[i].numWeek);

k++;

}

else if (i == dateIndex[j]) {

printf("%s\n", schedule[i].date);

j++;

}

else if (i != dateIndex[j]) {

printf("%s %s %s %s %s %s\n", schedule[i].subject, schedule[i].classTime, schedule[i].numOfAudience, schedule[i].teacherN, schedule[i].teacherS, schedule[i].teacherSN);

}

}

}

if (f == 2) {

printf("Введите номер недели: ");

scanf("%d", &num);

/\*проверка на существование введенной недели\*/

for (y = 0; y < sizeWeekIndex; y++) {

if (num == schedule[weekIndex[y]].numWeek) { //если номер введенной недели совпадает с каким-то номером недели массива индексов недель

weekExist = true;

break;

}

else if ((y == sizeWeekIndex - 1) && (weekExist == false)) {

system("cls");

printf("Выбранная вами неделя недоступна.\n");

break;

}

}

if (weekExist == true) {

//печать расписания на выбранную неделю

for (i = weekIndex[y]; i < weekIndex[y + 1]; i++) {

if (i == weekIndex[y]) {

printf("%s %d\n", schedule[i].week, schedule[i].numWeek);

}

else if (i == dateIndex[(num - 1) \* 5 + j]) {

printf("%s\n", schedule[i].date);

j++;

}

else {

printf("%s %s %s %s %s %s\n", schedule[i].subject, schedule[i].classTime, schedule[i].numOfAudience, schedule[i].teacherN, schedule[i].teacherS, schedule[i].teacherSN);

}

}

}

}

if (f == 3) {

printf("Введите дату (в формате dd:mm:yy): ");

scanf("%s", &dateWritten);

/\*проверка на существование выбранной даты\*/

if (strcmp(dateWritten, schedule[dateIndex[sizeDateIndex - 1]].date) == 0) {

for (y = 0; y < sizeDateIndex; y++) {

if (strcmp(dateWritten, schedule[dateIndex[y]].date) == 0) {

dateExist = true;

break;

}

else if ((y == sizeDateIndex - 1) && (dateExist == false)) {

system("cls");

printf("Выбранная вами дата недоступна.\n");

}

}

if (dateExist == true) {

//печать расписания на выбранную дату

for (i = dateIndex[y]; i < strCountSc; i++) {

if (i == dateIndex[y]) {

printf("%s\n", schedule[i].date);

}

else {

printf("%s %s %s %s %s %s\n", schedule[i].subject, schedule[i].classTime, schedule[i].numOfAudience, schedule[i].teacherN, schedule[i].teacherS, schedule[i].teacherSN);

}

}

}

}

else {

//расчет последнего индекса массива дат

dateIndex[sizeDateIndex] = 2 \* dateIndex[sizeDateIndex - 1] - dateIndex[sizeDateIndex - 2];

/\*проверка на существование выбранной даты\*/

for (y = 0; y < sizeDateIndex; y++) {

if (strcmp(dateWritten, schedule[dateIndex[y]].date) == 0) {

dateExist = true;

break;

}

else if ((y == sizeDateIndex - 1) && (dateExist == false)) {

system("cls");

printf("Выбранная вами дата недоступна.\n");

}

}

if (dateExist == true) {

//печать расписания на выбранную дату

for (i = dateIndex[y]; i < dateIndex[y + 1]; i++) {

if (i == dateIndex[y]) {

printf("%s\n", schedule[i].date);

}

else {

printf("%s %s %s %s %s %s\n", schedule[i].subject, schedule[i].classTime, schedule[i].numOfAudience, schedule[i].teacherN, schedule[i].teacherS, schedule[i].teacherSN);

}

}

}

}

}

}

void GetScheduleForT(int &strCountSc, scheduleInf \*schedule, int \*weekIndex, int \*dateIndex, int &sizeWeekIndex, int &sizeDateIndex) {

int i; //индексная переменная

int j; //переменная для запоминания индекса преподавателя

int y = 0; //индексная переменная

int k = 0; //переменная для правильной печати даты

char tName[256]; //строка для имени

char tSurName[256]; //строка для фамилии

char tSecondName[256]; //строка для отчества

bool teacherFound = false; //флаг "найден ли преподаватель"

dateIndex[sizeDateIndex] = 2 \* dateIndex[sizeDateIndex - 1] - dateIndex[sizeDateIndex - 2]; //расчет последнего инекса массива индексов дат

printf("Введите ФИО преподавателя, чтобы увидеть его расписание: ");

scanf("%s%s%s", &tSurName, &tName, &tSecondName);

for (i = dateIndex[y]; i < dateIndex[y + 1]; i++) { //проверяем имена преподавателей между соседними датами

if ((strcmp(tName, schedule[i].teacherN) == 0) && (strcmp(tSurName, schedule[i].teacherS) == 0) && (strcmp(tSecondName, schedule[i].teacherSN) == 0)) {

teacherFound = true;

j = i;

k++;

}

if (teacherFound == true) {

//печать расписания для преподавателя

if (k == 1) {

printf("%s\n", schedule[dateIndex[y]].date);

}

printf("%s %s %s %s %s %s\n", schedule[j].subject, schedule[j].classTime, schedule[j].numOfAudience, schedule[j].teacherS, schedule[j].teacherN, schedule[j].teacherSN);

}

if (i == dateIndex[y + 1] - 1) {

y++;

k = 0;

}

teacherFound = false; //опускаем флаг:"преподаватель не найден"

}

}

void getAbsent(int &strCountAb, absentInf \*absent, int &strCountS, studentInf \*student, int \*absentIndex, int &sizeAbsentIndex) {

int i = 0; //индексная переменная

int j; //индексная переменная

int y = 0; //индексная переменная

int k = 0; //индексная переменная

int t = 0; //индексная переменная

int t1 = 0; //переменная для запоминания следующего индекса массива индексов отсутствующих

char nameS[256]; //строка для имени

char surnameS[256]; //строка для фамилии

char secondnameS[256]; //строка для отчества

bool studentFoundGeneral = false; //флаг "найден ли студент в общем списке"

bool studentFoundAbsent = false; //флаг "найден ли студент в списке отсутствующих"

absentIndex[sizeAbsentIndex] = 2 \* absentIndex[sizeAbsentIndex - 1] - absentIndex[sizeAbsentIndex - 2]; //расчет последнего инекса массива индексов отсутствующих

printf("Введите ФИО студента, пропуски которого вы хотите увидеть: ");

scanf("%s%s%s", &surnameS, &nameS, &secondnameS);

for (i = 0; i < strCountS; i++) {

//поиск студента в общем списке

if ((strcmp(nameS, student[i].name) == 0) && (strcmp(surnameS, student[i].surname) == 0) && (strcmp(secondnameS, student[i].secondname) == 0)) {

studentFoundGeneral = true;

printf("Студент найден в общем списке.\n");

break;

}

else if ((studentFoundGeneral == false) && (i == strCountS - 1)) {

printf("Студент не найден в общем списке.\n");

}

}

if (studentFoundGeneral == true) {

//поиск студента в списке отсутствующих

for (j = 0; j < strCountAb; j++) {

if ((strcmp(nameS, absent[j].sName) == 0) && (strcmp(surnameS, absent[j].sSurName) == 0) && (strcmp(secondnameS, absent[j].sSecondName) == 0)) {

studentFoundAbsent = true;

break;

}

}

if (studentFoundAbsent == true) {

for (t = 0; t < sizeAbsentIndex; t++) {

if (j == absentIndex[t]) {

t1 = absentIndex[t + 1]; //запоминаем последний индекс массива индексов отсутствующих

break;

}

}

//печать прогулов отсутствующего

for (t = j; t < t1; t++) {

if (t == j)

printf("%s %s %s\n", absent[t].sSurName, absent[t].sName, absent[t].sSecondName);

else if (t != j)

printf("%s %s", absent[t].absentDate, absent[t].abSubject);

}

printf("\n");

}

else if ((studentFoundAbsent == false) && (j = strCountAb - 1)) {

printf("Студент посещал все занятия.\n");

}

}

}