

МИНОБРНАУКИ РОССИИ
САНКТ-ПЕТЕРБУРГСКИЙ ГОСУДАРСТВЕННЫЙ
ЭЛЕКТРОТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ
«ЛЭТИ» ИМ. В.И. УЛЬЯНОВА (ЛЕНИНА)
Кафедра информационных систем

ОТЧЕТ
по практической работе №1
по дисциплине «Программирование»
Тема: Типы данных, определяемые пользователем. Структуры

Студент(ка) гр. 0324

Сотина Е. А.

Преподаватель

Глущенко А. Г.

Санкт-Петербург

2021

Цель работы.

Изучение и организация структур; получение практических навыков работы со структурами; определение преимуществ и недостатков использования структур.

Основные теоретические положения.

Структуры представляют собой группы связанных между собой, как правило, разнотипных переменных, объединенных в единый объект, в отличие от массива, все элементы которого однотипны. В языке C++ структура является видом класса и обладает всеми его свойствами. Чаще всего ограничиваются тем, как структуры представлены в языке C:

```
struct [имя_типа] {  
    тип_1 элемент_1;  
    тип _2 элемент_2;  
    ...  
    тип_k элемент_k;  
} [ список_описателей ];
```

Описание структуры начинается ключевым словом struct. Каждая входящая в структуру переменная называется членом (полем, элементом) структуры и описывается типом данных и именем. Поля структуры могут быть любого типа данных. Их количество не лимитировано.

Вся эта конструкция является инструкцией языка программирования, поэтому после нее всегда должен ставиться символ ‘;’.

При описании структуры память для размещения данных не выделяется. Работать с описанной структурой можно только после того, как будет определена переменная (переменные) этого типа данных, только при этом компилятор выделит необходимую память.

Для инициализации структуры значения ее элементов перечисляют в фигурных скобках в порядке их описания:

```
struct complex{  
    float real, im;  
} data [2][2] = {  
    {{1,1}, {2,2}},  
    {{3,3}, {4,4}}  
};
```

Все поля структурных переменных располагаются в непрерывной области памяти одно за другим. Общий объем памяти, занимаемый структурой, равен

сумме размеров всех полей структуры. Для определения размера структуры следует использовать инструкцию `sizeof()`.

Для того чтобы записать данные в структурную переменную, необходимо каждому полю структуры присвоить определенное значение. Для этого необходимо использовать оператор `'.'` («точка»):

```
struct Stack { // Стек
    float arr[100];
    short topIndex;
};
...
Stack stack; // Объявляем переменную типа Stack
Stack.arr[0] = 1;
...
```

При доступе к определенному полю его следует рассматривать как обычную переменную, тип данных которой соответствует типу этого поля. Поля структур могут участвовать в качестве операндов любых выражений, допускающих использование операндов соответствующего типа данных.

Копирование данных из одной структурной переменной в другую осуществляется простой операцией присваивания, независимо от количества полей и размера структуры (это можно делать только в том случае, когда обе переменные одного и того же типа).

В программировании очень часто используются такие конструкции, как массивы структур. Например, сведения о студентах некоторой учебной группы можно хранить в массиве студентов:

```
t_Student Gruppa_N [30];
```

Был определен 30-элементный массив, каждый элемент которого предназначен для хранения данных одного студента. Получение доступа к данным некоторого студента из группы *N* осуществляется обычной индексацией переменной массива. Поскольку поля структуры могут быть любого типа данных, то они в свою очередь могут быть другой структурой или массивом других структур:

```
struct Stud
{
    char FN[100];
    short listNumber;
};

struct Group
{
```

```

    int groupNumber;
    short students;
    Stud stud[30];
};

```

Но в структуре поля нельзя использовать элемент, тип которого совпадает с типом самой структуры, так как рекурсивное использование структур запрещено.

Любая структурная переменная занимает в памяти определенное положение, характеризующееся конкретным адресом. Для работы с адресами структурных переменных (как и для простых переменных) можно использовать указатели. Указатели на структурные переменные определяются точно так же, как и для обычных переменных. Разыменование указателя (обращение к данным по адресу, хранящемуся в указателе) осуществляется также обычным образом.

Через указатели можно работать с отдельными полями структур. Для доступа к полю структуры через указатель используется оператор '→' («стрелка»), а не «точка».

Структуры можно использовать в качестве параметров функций, как и обычные переменные. Для структур поддерживаются все три механизма передачи данных: по значению, через указатели и по ссылке.

Передачу структур в функции по значению необходимо использовать аккуратно:

```

void WriteStudent ( t_Student S )
{
    cout << "Фамилия: " << S.Fam << endl;
    cout << "Имя: " << S.Name << endl;
    cout << "Год рождения: " << S.Year << endl;
    if ( S.Sex )
        cout << "Пол: " << "М\n";
    else
        cout << "Пол: " << "Ж\n";
    cout << "Средний балл: " << S.Grade << endl;
}

```

Вызов такой функции сопровождается дополнительным расходом памяти для создания локальной переменной *S* и дополнительными затратами времени на физическое копирование данных из аргумента в параметр *S*. Учитывая то, что объем структур может быть очень большим, эти дополнительные затраты вычислительных ресурсов могут быть чрезмерными.

Предпочтительно использование передачи структуры по указателю или ссылке:

```
void WriteStudent ( t_Student *S )
{
    cout << "Фамилия: " << S -> Fam << endl;
    cout << "Имя: " << S -> Name << endl;
    cout << "Год рождения: " << S -> Year << endl;
    if ( S -> Sex )
        cout << "Пол: " << "М\n";
    else
        cout << "Пол: " << "Ж\n";
    cout << "Средний балл: " << S -> Grade << endl;
}
```

Фактической передачи данных в функцию не осуществляется. Дополнительные затраты памяти для создания локальной переменной небольшие – это адрес памяти (4 байта, независимо от размера самой структуры). Вызов такой функции будет происходить быстрее, а расход памяти будет существенно меньше, чем при передаче данных по значению.

Передача по ссылке по эффективности эквивалентна передаче данных через указатель. Однако, поскольку при передаче данных по ссылке все адресные преобразования берет на себя компилятор, существенно упрощается программирование действий со структурами. При использовании ссылочных параметров структурных типов доступ к членам структуры осуществляется обычным способом – с помощью оператора «точка».

Недостатком этих способов является то, что случайные изменения значений полей структуры внутри функции отразятся на значении аргумента после окончания работы функции. Если необходимо предотвратить изменения переданных по адресу аргументов, можно при определении соответствующего параметра объявить его константой (использовать спецификатор `const`).

Постановка задачи.

Необходимо создать массив структур, содержащий информацию о студентах: ФИО, пол, номер группы, номер в списке группы, оценки за прошедшую сессию (всего 3 экзамена и 5 дифференцированных зачетов), форма обучения, отметка времени о внесении или изменении данных. Ввод и изменение данных обо всех студентах должен осуществляться в файл `students`.

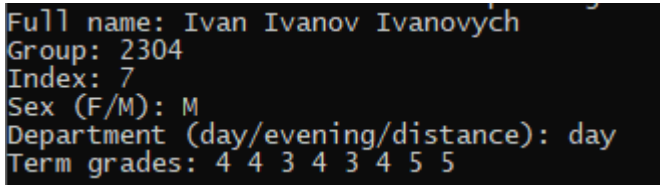
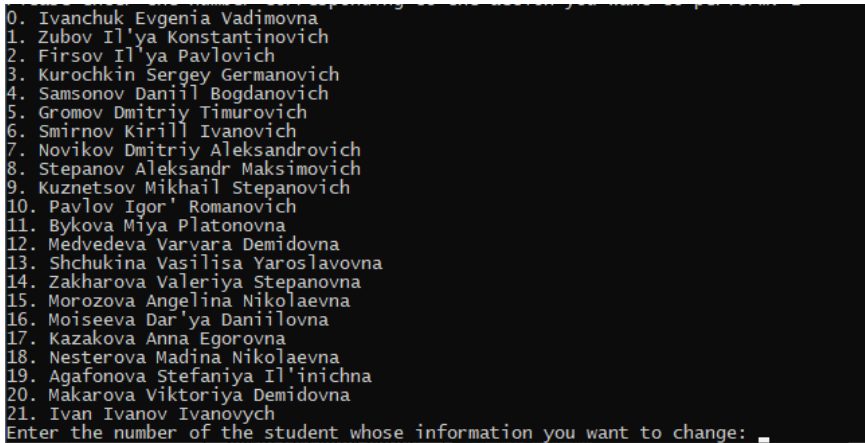
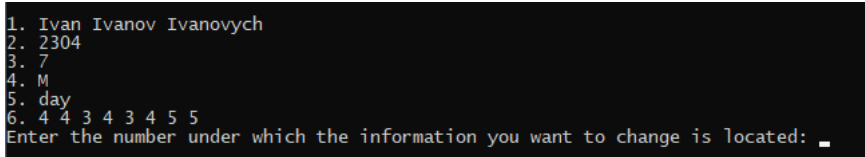
Написать функции, реализующие операции со структурами (ввод данных с клавиатуры):

1. Создание новой записи о студенте.
2. Внесение изменений в уже имеющуюся запись.
3. Вывод всех данных о студентах.
4. Вывод информации обо всех студентах группы N. N – инициализируется пользователем.
5. Вывод топа самых успешных студентов с наивысшим по рейтингу средним баллом за прошедшую сессию.
6. Вывод количества студентов мужского и женского пола.
7. Определение количества студентов, которые будут получать стипендию (стипендия начисляется, если у студента нет троек и очная форма обучения).
8. Вывод данных о студентах, которые не получают стипендию; учатся только на «хорошо» и «отлично»; учатся только на «отлично»;
9. Вывод данных о студентах, имеющих номер в списке – k.
10. Вывод всех записей, сделанных в день, который введет пользователь. Вывод всех записей, сделанных после полудня. Вывод всех записей, сделанных до полудня.

Выполнение работы.

Код программы представлен в приложении А.

Ввод пользователем и обработка данных	Работа алгоритма и вывод на экран
Меню	
При запуске программы перед пользователем появляется окно с меню, где он может выбрать тип будущей вводимой последовательност и	<p>Меню:</p> <pre>1. Create a new student record 2. Make changes to an existing record 3. Display all student data 4. Output all data about students who meet the condition 5. Display the top students by average grade per session 6. Print the number of students who meet the condition 7. Display information about students depending on the time when the entry was made 0. Exit Please enter the number corresponding to the action you want to perform:</pre> <p>Проверка на ввод символов, которые не входят в диапазон выбора:</p> <pre>1. Create a new student record 2. Make changes to an existing record 3. Display all student data 4. Output all data about students who meet the condition 5. Display the top students by average grade per session 6. Print the number of students who meet the condition 7. Display information about students depending on the ti me when the entry was made 9. Exit Please enter the number corresponding to the action you w ant to perform: ffff ff Incorrect input Enter a number: fff ff ffff Incorrect input Enter a number: fffffff Incorrect input Enter a number: 9 Для продолжения нажмите любую клавишу . . . _</pre>

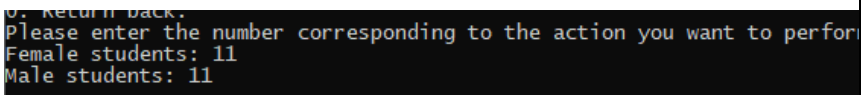
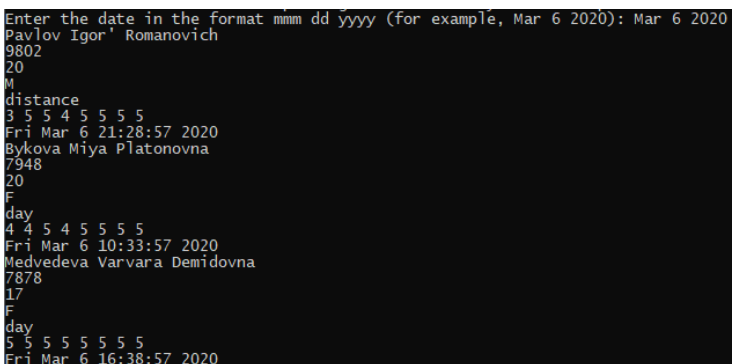
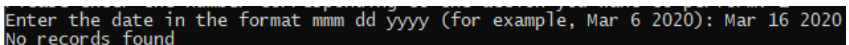
Создание новой записи	
При вводе пользователем корректного значения пункта меню и выбора создания новой записи, пользователь может добавить данные о студенте, но только на русском языке.	Ввод информации выглядит следующим образом: 
Внесение изменений в имеющуюся запись	
Если пользователь допустил ошибку в создании карточки студента или данные изменились, он может внести изменения по пунктам, которые были созданы ранее.	<p>При необходимости внесения изменений в запись о студенте, перед пользователем появляется меню:</p>  <p>Выбрав нужного ученика, на выбор пользователю даётся список информации, которую он может поменять:</p>  <p>Как только пользователь вводит корректное значение (в данном случае от 1 до 10), его ответ обрабатывается и предлагается изменить существующие данные на новые. После пользователь возвращается (при необходимости) в главное меню.</p>

Продолжение Таблицы

Вывод всей информации о студентах		
Данное действие позволяет вывести всех студентов, находящихся в списке.	Ivanchuk Evgenia Vadimovna 9894 6 F evening 3 4 5 5 5 5 5 Sat Feb 29 20:05:44 2020 Zubov Il'ya Konstantinovich 5801 2 M distance 4 4 3 3 4 4 5 4 Sun Feb 23 15:35:39 2020 Firsov Il'ya Pavlovich 7878 3 M evening 5 5 4 5 3 3 4 3 Sun Feb 23 23:35:39 2020 Kurochkin Sergey Germanovich 5866 3 M evening 3 3 5 3 4 5 4 3 Sun Feb 23 03:38:39 2020 Samsonov Daniil Bogdanovich 9894 15 M day 3 3 5 5 4 4 5 4 Tue Feb 25 00:14:46 2020	

Продолжение Таблицы

Вывод всей информации о студентах в зависимости от условий	
Данные о студентах можно сортировать по номеру группы, по оценкам (хорошисты и отличники), по номеру в списке своей группы.	<p>Если студент с нужным номером не найден:</p> <pre>Please enter the student's number in the list: 88 No such student found</pre> <p>Вывод информации:</p> <pre>Please enter the student's number in the list: 6 Ivanchuk Evgenia Vadimovna 9894 6 F evening 3 4 5 5 5 5 5 Sat Feb 29 20:05:44 2020</pre>
Рейтинг студентов	
Пользователь в главном меню может выбрать такую опцию как рейтинг студентов, чтобы наглядно увидеть список студентов с минимальными данными и рейтингу их оценок.	<p>Вывод топа самых успешных студентов с наивысшим по рейтингу средним баллом за прошедшую сессию</p> <pre>1. Medvedeva Varvara Demidovna = 5 2. Shchukina Vasilisa Yaroslavovna = 5 3. Nesterova Madina Nikolaevna = 5 4. Agafonova Stefaniya Il'ichna = 5 5. Zakharova Valeriya Stepanovna = 4.75 6. Ivanchuk Evgenia Vadimovna = 4.625 7. Pavlov Igor' Romanovich = 4.625 8. Bykova Miya Platonovna = 4.625 9. Smirnov Kirill Ivanovich = 4.5 10. Stepanov Aleksandr Maksimovich = 4.5 11. Moiseeva Dar'ya Daniilovna = 4.5 12. Kazakova Anna Egorovna = 4.375 13. Gromov Dmitriy Timurovich = 4.25 14. Kuznetsov Mikhail Stepanovich = 4.25 15. Samsonov Daniil Bogdanovich = 4.125 16. Morozova Angelina Nikolaevna = 4.125 17. Makarova Viktoriya Demidovna = 4.125 18. Firsov Il'ya Pavlovich = 4 19. Ivan Ivanov Ivanovich = 4</pre>

Информация о студентах в количестве	
Если пользователь хочет узнать количество студентов каждого пола или получивших стипендию.	<p>Подсчёт количества студентов разных полов:</p> 
Вывод информации о студентах в зависимости от времени сделанной записи	
Пользователь может отсортировать записи по времени их создания или последнего редактирования: он может ввести дату, чтобы получить записи, сделанные в этот день, получить записи, сделанные до полудня или после полудня	<p>Выбирая возможность получить записи в конкретный день, пользователь может ввести дату в определённом формате. Если записи были найдены, их список будет выведен:</p>  <p>В противном случае, пользователь получит сообщение о том, что таких записей не существует:</p> 

Выводы.

Были изучены и применены структуры на практике.

Разработана программа, способная выводить и записывать данные, введённые пользователем. Также программа способна осуществлять сортировку введённых данных по параметрам, определяемые пользователем.

ПРИЛОЖЕНИЕ А

РАБОЧИЙ КОД

Название файла: lab1.cpp

```
#include <iostream>
#include <fstream>
#include <string>
#include <iomanip>
#include <ctime>
#include <time.h>
#pragma warning(disable : 4996)
using namespace std;

struct Profile //анкета студента
{
    string fullName;
    char sex;
    unsigned short int group;
    unsigned short int numberList;
    int term[8];
    char depart[9];
    string date; // Дата внесения изменения в запись (post/update)
};

//удаление лишних пробелов (для дат)
string DelSpaces(string s)
{
    for (int j = 0; j < s.length(); j++)
    {
        if (s[j] == ' ')
        {
            while (s[j + 1] == ' ') s.erase(j + 1, 1);
        }
    }
    if (s[0] == ' ') s.erase(0, 1);
    if (s[s.length() - 1] == ' ') s.erase(s.length() - 1, 1);
    return s;
}

//считывание с экрана информации о студенте
void newStudent()
{
    Profile Student;
    cin.clear();
```

```

char trash;
cout << "Full name: ";
cin >> trash;
getline(cin, Student.fullName);
Student.fullName = trash + Student.fullName;
cout << "Group: ";
cin >> Student.group;
cout << "Index: ";
cin >> Student.numberList;
cout << "Sex (F/M): ";
cin >> Student.sex;
cout << "Department (day/evening/distance): ";
cin >> Student.depart;
cout << "Term grades: ";
bool temp = false; //имеются ли двойки?
for (int i = 0; i < 8; i++)
{
    cin >> Student.term[i];
    if (Student.term[i] == 2) { temp = true; } //да, двойка
имеется
}

//запоминание даты считывания
struct tm* loctime;
time_t curtime;
time(&curtime);
loctime = localtime(&curtime);
Student.date = asctime(loctime);
Student.date = DelSpaces(Student.date);

if (temp) { cout << '\n' << "This student will be expelled. The
profile will not be saved in the database."; }
else
{
    /*Попытка создать файл с введёнными данными*/
    ofstream fout("students.txt", ios_base::app);
    if (!fout.is_open()) { cout << '\n' << "Saving error!"; }
    else
    {
        /* == Вывод записи == */
        fout << Student.fullName << "\n";
        fout << Student.group << "\n";
        fout << Student.numberList << "\n";
        fout << Student.sex << "\n";
    }
}

```

```

        fout << Student.depart << "\n";
        for (int i = 0; i < 8; i++)
        {
            fout << Student.term[i] << " ";
        }
        fout << "\n";
        fout << Student.date;
        fout.close();
    }
}

// Функция посчёта количества студентов
int countStudents()
{
    ifstream fin("students.txt");
    if (fin.is_open())
    {
        int temp = 0; //количество строк
        string data;
        while (!fin.eof()) //пока указатель потока не достигнет конца
        файла
        {
            getline(fin, data); //считывается строка
            temp++;
        }
        fin.close();
        int n;
        n = temp / 7; //количество строк поделить на кол-во строк
        одной анкеты студента = кол-во анкет студента
        return n;
    }
    else return 0;
}

//вывод файла всей информации о студентах
void outputStudents()
{
    ifstream fin("students.txt");
    if (!fin.is_open()) // если файл не открыт
        cout << "File does not exist\n"; // сообщить об этом
    else
    {
        int temp;

```

```

        temp = countStudents();
        if (temp == 0)
            cout << "The File is empty";
        else
        {
            string data; // буфер промежуточного хранения
считываемого из файла текста
            while (!fin.eof())
            {
                getline(fin, data); // Считываем очередную строку
                cout << data << '\n'; // Выводим строку на экран
            }
            fin.close();
        }
    }
}

//подсчёт количества студентов разных полов
void F_and_M()
{
    ifstream fin("students.txt");
    if (!fin.is_open()) { cout << "Error!"; }
    else
    {
        int size;
        size = countStudents();
        if (size == 0) { cout << "The database is empty." << endl; }
        else
        {
            Profile* student = new Profile[size];
            char trash;
            for (int i = 0; i < size; i++) // Считываем данные всех
студентов в массив структур
            {
                getline(fin, student[i].fullName, '\n');
                fin >> student[i].group >> student[i].numberList >>
student[i].sex;
                fin >> student[i].depart;
                for (int j = 0; j < 8; j++) {
                    fin >> student[i].term[j];
                }
                fin >> trash;
                getline(fin, student[i].date, '\n');
            }
        }
    }
}

```

```

        fin.close();

        //подсчёт и вывод
        int f = 0,
            m = 0;
        for (int i = 0; i < size; i++)
        {
            if (student[i].sex == 'F') { f++; }
            if (student[i].sex == 'M') { m++; }
        }
        cout << "\nFemale students: " << f << "\n";
        cout << "Male students: " << m << "\n";
        //конец подсчёта и вывода

        delete[] student;
    }
}

//Количество студентов со стипендией
void stipend()
{
    ifstream fin("students.txt");
    if (!fin.is_open()) { cout << "Error!"; }
    else
    {
        int size;
        size = countStudents();
        if (size == 0) { cout << "The database is empty." << endl; }
        else
        {
            Profile* student = new Profile[size];
            char trash;
            bool three; //наличие троек
            int s = 0; //количество студентов со стипендией
            for (int i = 0; i < size; i++) // Считываем данные всех
студентов в массив структур
            {
                getline(fin, student[i].fullName);
                fin >> student[i].group >> student[i].numberList >>
student[i].sex;

                fin >> student[i].depart;
                three = false; //по умолчанию троек нет
                for (int j = 0; j < 8; j++) {

```



```

        fin >> student[i].term[j];
        if (student[i].term[j] == 3) { three = true; }
    /*если встречена тройка, запоминаем это
    */
    fin >> trash;
    getline(fin, student[i].date);
    student[i].date = trash + student[i].date;
    if (!(three) && (student[i].depart[1] == 'a')) {
s++; } /*подсчёт студентов со стипендией
    */
    fin.close();

    cout << "\nStudents with a stipend: " << s; /*Вывод

    delete[] student;

    }
}

//Вывод данных о студентах без стипендии
void notStipend()
{
    ifstream fin("students.txt");
    if (!fin.is_open()) { cout << "Error!"; }
    else
    {
        int size;
        size = countStudents();
        if (size == 0) { cout << "The database is empty." << endl; }
        else
        {
            Profile* student = new Profile[size];
            char trash;
            bool check; /*true - надо вывести информацию. Степендия
не может быть получена
            for (int i = 0; i < size; i++) // Считываем данные всех
студентов в массив структур
            {
                check = false; /*по умолчанию false
                getline(fin, student[i].fullName);
                fin >> student[i].group >> student[i].numberList >>
student[i].sex;

                fin >> student[i].depart;

```

```

        if (!(student[i].depart[1] == 'a')) { check = true;
    } //если не дневное обучение - степендии нет
        for (int j = 0; j < 8; j++) {
            fin >> student[i].term[j];
            if (student[i].term[j] == 3) { check = true; }
    //если встречена тройка, запоминаем это
        }
        fin >> trash;
        getline(fin, student[i].date);
        student[i].date = trash + student[i].date;

        //Вывод
        if (check)
        {
            cout << student[i].fullName << "\n";
            cout << student[i].group << "\n";
            cout << student[i].numberList << "\n";
            cout << student[i].sex << "\n";
            cout << student[i].depart << "\n";
            for (int j = 0; j < 8; j++) { cout <<
student[i].term[j] << " "; }
            cout << "\n" << student[i].date << "\n";
        }
        //конец вывода
    }
    fin.close();

    delete[] student;
}

}

}

//Вывод данных о студентах-отличниках
void excellentTerm()
{
    ifstream fin("students.txt");
    if (!fin.is_open()) { cout << "Error!"; }
    else
    {
        int size;
        size = countStudents();
        if (size == 0) { cout << "The database is empty." << endl; }
        else
        {

```

```

        Profile* student = new Profile[size];
        char trash;
        bool check; /*true - надо вывести информацию. Является
ОТЛИЧНИКОМ
        for (int i = 0; i < size; i++) // Считываем данные всех
студентов в массив структур
        {
            check = true; /*по умолчанию true
            getline(fin, student[i].fullName);
            fin >> student[i].group >> student[i].numberList >>
student[i].sex;
            fin >> student[i].depart;
            for (int j = 0; j < 8; j++) {
                fin >> student[i].term[j];
                if ((student[i].term[j] == 3) ||
(student[i].term[j] == 4)) { check = false; } /*если встречена 3 или 4,
запоминаем это
            }
            fin >> trash;
            getline(fin, student[i].date);
            student[i].date = trash + student[i].date;

            //Вывод
            if (check)
            {
                cout << student[i].fullName << "\n";
                cout << student[i].group << "\n";
                cout << student[i].numberList << "\n";
                cout << student[i].sex << "\n";
                cout << student[i].depart << "\n";
                for (int j = 0; j < 8; j++) { cout <<
student[i].term[j] << " "; }
                cout << "\n" << student[i].date << "\n";
            }
            //конец вывода
        }
        fin.close();

        delete[] student;
    }
}

//Вывод данных о хорошистах

```

```

void B_GradeTerm()
{
    ifstream fin("students.txt");
    if (!fin.is_open()) { cout << "Error!"; }
    else
    {
        int size;
        size = countStudents();
        if (size == 0) { cout << "The database is empty." << endl; }
        else
        {
            Profile* student = new Profile[size];
            char trash;
            bool check; /*true - надо вывести информацию. Является
ОТЛИЧНИКОМ
            for (int i = 0; i < size; i++) // Считываем данные всех
студентов в массив структур
            {
                check = true; /*по умолчанию true
                getline(fin, student[i].fullName);
                fin >> student[i].group >> student[i].numberList >>
student[i].sex;
                fin >> student[i].depart;
                for (int j = 0; j < 8; j++) {
                    fin >> student[i].term[j];
                    if (student[i].term[j] == 3) { check = false;
} /*если встречена 3, запоминаем это
                }
                fin >> trash; /*без него почему-то ломается
                getline(fin, student[i].date);
                student[i].date = trash + student[i].date;

                //Вывод
                if (check)
                {
                    cout << student[i].fullName << "\n";
                    cout << student[i].group << "\n";
                    cout << student[i].numberList << "\n";
                    cout << student[i].sex << "\n";
                    cout << student[i].depart << "\n";
                    for (int j = 0; j < 8; j++) { cout <<
student[i].term[j] << " "; }
                    cout << "\n" << student[i].date << "\n";
                }
            }
        }
    }
}

```

```

        //конец вывода
    }
    fin.close();

    delete[] student;
}

}

//Вывод данных о студентах группы N
void groupN(unsigned short int n)
{
    ifstream fin("students.txt");
    if (!fin.is_open()) { cout << "Error!"; }
    else
    {
        int size;
        size = countStudents();
        if (size == 0) { cout << "The database is empty." << endl; }
        else
        {
            Profile* student = new Profile[size];
            char trash;
            bool check; /*true - надо вывести информацию. Является
студентом группы N
            int k = 0;
            for (int i = 0; i < size; i++) // Считываем данные всех
студентов в массив структур
            {
                check = false; /*по умолчанию false
                getline(fin, student[i].fullName);
                fin >> student[i].group >> student[i].numberList >>
student[i].sex;

                if (student[i].group == n) { check = true; }
                fin >> student[i].depart;
                for (int j = 0; j < 8; j++) { fin >>
student[i].term[j]; }
                fin >> trash;
                getline(fin, student[i].date);
                student[i].date = trash + student[i].date;

                //вывод
                if (check)
                {

```

```

        cout << student[i].fullName << "\n";
        cout << student[i].group << "\n";
        cout << student[i].numberList << "\n";
        cout << student[i].sex << "\n";
        cout << student[i].depart << "\n";
        for (int j = 0; j < 8; j++) { cout <<
student[i].term[j] << " "; }
        cout << "\n" << student[i].date << "\n";
        k++;
    }
    //конец вывода
}
if (k == 0)
{
    cout << "No such student found";
}
fin.close();

delete[] student;
}
}

//Вывод данных о студентах номера k
void numberListK(unsigned short int k)
{
    ifstream fin("students.txt");
    if (!fin.is_open()) { cout << "Error!"; }
    else
    {
        int size;
        size = countStudents();
        if (size == 0) { cout << "The database is empty." << endl; }
        else
        {
            Profile* student = new Profile[size];
            char trash;
            bool check; /*true - надо вывести информацию. Является
студентом номера k
            int n = 0;
            for (int i = 0; i < size; i++) // Считываем данные всех
студентов в массив структур
            {
                check = false; /*по умолчанию false

```

```

        getline(fin, student[i].fullName);
        fin >> student[i].group >> student[i].numberList >>
student[i].sex;

        if (student[i].numberList == k) { check = true; }
        fin >> student[i].depart;
        for (int j = 0; j < 8; j++) { fin >>
student[i].term[j]; }
        fin >> trash;
        getline(fin, student[i].date);
        student[i].date = trash + student[i].date;

        //Вывод
        if (check)
        {
            cout << student[i].fullName << "\n";
            cout << student[i].group << "\n";
            cout << student[i].numberList << "\n";
            cout << student[i].sex << "\n";
            cout << student[i].depart << "\n";
            for (int j = 0; j < 8; j++) { cout <<
student[i].term[j] << " "; }
            cout << "\n" << student[i].date << "\n";
            n++;
        }
        //конец вывода
    }
    if (n == 0)
    {
        cout << "No such student found";
    }
    fin.close();

    delete[] student;
}

}

//Вывод данных до полудня
void tillNoon()
{
    ifstream fin("students.txt");
    if (!fin.is_open()) { cout << "Error!"; }
    else
    {

```

```

int size;
size = countStudents();
if (size == 0) { cout << "The database is empty." << endl; }
else
{
    Profile* student = new Profile[size];
    char trash;
    bool check; /*true - надо вывести информацию. Запись
сделана до 12:00
    int k = 0;
    for (int i = 0; i < size; i++) // Считываем данные всех
студентов в массив структур
    {
        check = false; /*по умолчанию false
        getline(fin, student[i].fullName);
        fin >> student[i].group >> student[i].numberList >>
student[i].sex;
        fin >> student[i].depart;
        for (int j = 0; j < 8; j++) { fin >>
student[i].term[j]; }
        fin >> trash;
        getline(fin, student[i].date);
        student[i].date = trash + student[i].date;

        //получаем время, когда была сделана запись
        string date1 = student[i].date;
        char date[9];
        for (int i = 0; i < 8; i++)
        {
            date[i] = date1[i + date1.size() - 13];
        }
        date[8] = '\0'; //получается время в формате
hh:mm:ss

        int hh = (int)date[0] - (int)'0'; //преобразуем из
string в int, вырезая ненужное и готовя для сравнения
        hh *= 10 + ((int)date[1] - (int)'0');
        if (hh < 12) { check = true; } //если запись
сделана раньше 12 часов, выводим

        //вывод
        if (check)
        {
            cout << student[i].fullName << "\n";

```



```

        cout << student[i].group << "\n";
        cout << student[i].numberList << "\n";
        cout << student[i].sex << "\n";
        cout << student[i].depart << "\n";
        for (int j = 0; j < 8; j++) { cout <<
student[i].term[j] << " "; }
        cout << "\n" << student[i].date << "\n";
        k++;
    }
    //конец вывода
}
if (k == 0)
{
    cout << "No records found";
}
fin.close();

delete[] student;
}
}

//Вывод данных после полудня
void afterNoon()
{
    ifstream fin("students.txt");
    if (!fin.is_open()) { cout << "Error!"; }
    else
    {
        int size;
        size = countStudents();
        if (size == 0) { cout << "The database is empty." << endl; }
        else
        {
            Profile* student = new Profile[size];
            char trash;
            bool check; /*true - надо вывести информацию. Запись
сделана после 12:00
            int k = 0;
            for (int i = 0; i < size; i++) // Считываем данные всех
студентов в массив структур
            {
                check = false; /*по умолчанию false
                getline(fin, student[i].fullName);

```

```

        fin >> student[i].group >> student[i].numberList >>
student[i].sex;
        fin >> student[i].depart;
        for (int j = 0; j < 8; j++) { fin >>
student[i].term[j]; }
        fin >> trash;
        getline(fin, student[i].date);
        student[i].date = trash + student[i].date;

        //получаем время, когда была сделана запись
        string date1 = student[i].date;
        char date[9];
        for (int i = 0; i < 8; i++)
        {
            date[i] = date1[i + date1.size() - 13];
        }
        date[8] = '\0'; //получается время в формате
hh:mm:ss

        //часы
        int hh = (int)date[0] - (int)'0'; //преобразуем из
string в int, вырезая ненужное и готовя для сравнения
        hh *= 10 + ((int)date[1] - (int)'0');
        //минуты
        int mm = (int)date[3] - (int)'0';
        mm *= 10 + ((int)date[4] - (int)'0');
        //секунды
        int ss = (int)date[6] - (int)'0';
        ss *= 10 + ((int)date[7] - (int)'0');

        if (hh >= 12)
        {
            if (hh == 12) { if ((mm != 0) || (ss != 0)) {
check = true; } }
            else { check = true; }
        } //если запись сделана после 12 часов, выводим

        //вывод
        if (check)
        {
            cout << student[i].fullName << "\n";
            cout << student[i].group << "\n";
            cout << student[i].numberList << "\n";
            cout << student[i].sex << "\n";

```

```

        cout << student[i].depart << "\n";
        for (int j = 0; j < 8; j++) { cout <<
student[i].term[j] << " "; }
        cout << "\n" << student[i].date << "\n";
        k++;
    }
    //конец вывода
}
if (k == 0)
{
    cout << "No records found";
}
fin.close();

delete[] student;
}
}

void thatDate(string Day)
{
    ifstream fin("students.txt");
    if (!fin.is_open()) { cout << "Error!"; }
    else
    {
        int size;
        size = countStudents();
        if (size == 0) { cout << "The database is empty." << endl; }
        else
        {
            Profile* student = new Profile[size];
            char trash;
            bool check; /*true - надо вывести информацию. Запись
сделана в данный день
            int k = 0;
            for (int i = 0; i < size; i++) // Считываем данные всех
студентов в массив структур
            {
                check = false; /*по умолчанию false
                getline(fin, student[i].fullName);
                fin >> student[i].group >> student[i].numberList >>
student[i].sex;
                fin >> student[i].depart;

```

```

        for (int j = 0; j < 8; j++) { fin >>
student[i].term[j]; }
        fin >> trash;
        getline(fin, student[i].date);
        student[i].date = trash + student[i].date;

        //сравниваем
        //24 символа, если дата с %dd, 23 если %d в полной
date
        //(Day).size(); 11 символов с %dd, 10 символов с %d
        //при совпадении разница размеров должна давать 13.
Отсеиваем при несовпадении
        int daySize = (Day).size();
        if (((student[i].date).size() - daySize) == 13)
        {
            string date1 = student[i].date;
            bool same = true; //проверяем на совпадение
месяцев и дней. По умолчанию true
            for (int i = 0; i < 6; i++)
            {
                if (Day[i] != date1[i + 4]) //y date1
надо пропустить первые 4 символа
                {
                    same = false;
                    break;
                }
            }
            for (int i = daySize - 1; i > 8; i--)
            {
                if (Day[i] != date1[i + 13])
                {
                    same = false;
                    break;
                }
            }
            if (same) //если месяца совпали
            {
                check = true;
            }
        }

        //вывод
        if (check)
        {

```

```

        cout << student[i].fullName << "\n";
        cout << student[i].group << "\n";
        cout << student[i].numberList << "\n";
        cout << student[i].sex << "\n";
        cout << student[i].depart << "\n";
        for (int j = 0; j < 8; j++) { cout <<
student[i].term[j] << " "; }
        cout << "\n" << student[i].date << "\n";
        k++;
    }
    //конец вывода
}
if (k == 0)
{
    cout << "No records found";
}
fin.close();

delete[] student;
}
}

void topTerm()
{
    ifstream fin("students.txt");
    if (!fin.is_open()) { cout << "Error!"; }
    else
    {
        int size;
        size = countStudents();
        if (size == 0) { cout << "The database is empty." << endl; }
        else
        {
            Profile* student = new Profile[size];
            float* term = new float[size]; /*массив со средними
оценками
            char trash;
            for (int i = 0; i < size; i++) // Считываем данные всех
студентов в массив структур
            {
                getline(fin, student[i].fullName, '\n');
                fin >> student[i].group >> student[i].numberList >>
student[i].sex;

```

```

        fin >> student[i].depart;
        term[i] = 0;
        for (int j = 0; j < 8; j++) {
            fin >> student[i].term[j];
            term[i] += (float)(student[i].term[j]);
        }
        term[i] /= 8;
        fin >> trash;
        getline(fin, student[i].date, '\n');
    }
    fin.close();

    //сортировка и вывод
    for (int i = 0; i < size; i++)
    {
        for (int j = 0; j < size - 1; j++)
        {
            if (term[j] < term[j + 1])
            {
                float m = term[j];
                term[j] = term[j + 1];
                term[j + 1] = m;

                string name = student[j].fullName;
                student[j].fullName = student[j +
1].fullName;
                student[j + 1].fullName = name;
            }
        }
    }

    //Вывод
    for (int i = 0; i < size; i++)
    {
        cout << i + 1 << ". " << student[i].fullName << " =
" << term[i] << "\n";
    }

    //конец подсчёта и вывода
    delete[] term;
    delete[] student;
}
}
}

```

```

void changeFile()
{
    ifstream fin("students.txt");
    if (!fin.is_open()) { cout << "Error!"; }
    else
    {
        int size;
        size = countStudents();
        if (size == 0) { cout << "The database is empty." << endl; }
        else
        {
            Profile* student = new Profile[size];
            char trash;
            for (int i = 0; i < size; i++) // Считываем данные всех
студентов в массив структур
            {
                getline(fin, student[i].fullName, '\n');
                fin >> student[i].group >> student[i].numberList >>
student[i].sex;
                fin >> student[i].depart;
                for (int j = 0; j < 8; j++) {
                    fin >> student[i].term[j];
                }
                fin >> trash;
                getline(fin, student[i].date, '\n');
                student[i].date = trash + student[i].date;
            }
            fin.close();

            //изменение информации
            for (int i = 0; i < size; i++)
            {
                cout << i << ". " << student[i].fullName << "\n";
            }
            cout << "Enter the number of the student whose
information you want to change: ";
            int numbStud;
            cin >> numbStud;
            if (numbStud >= size) { cout << "There is no such
student"; return; }
            cout << "\n1. " << student[numbStud].fullName << "\n";
            cout << "2. " << student[numbStud].group << "\n";
            cout << "3. " << student[numbStud].numberList << "\n";

```

```

        cout << "4. " << student[numbStud].sex << "\n";
        cout << "5. " << student[numbStud].depart << "\n6. ";
        for (int j = 0; j < 8; j++) { cout <<
student[numbStud].term[j] << " "; }
        cout << "\nEnter the number under which the information
you want to change is located: ";
        int sw;
        cin >> sw;

        bool check = false; //имеются ли изменения?
        bool temp = false; //имеются ли двойки?
        switch (sw)
        {
        case 1:
            cout << "\nEnter the student's new full name: ";
            cin >> trash;
            getline(cin, student[numbStud].fullName);
            student[numbStud].fullName = trash +
student[numbStud].fullName;
            check = true;
            break;
        case 2:
            cout << "\nEnter a new student group: ";
            cin >> student[numbStud].group;
            check = true;
            break;
        case 3:
            cout << "\nEnter a new number in the student list:
";

            cin >> student[numbStud].numberList;
            check = true;
            break;
        case 4:
            cout << "\nEnter the student's gender (F/M): ";
            cin >> student[numbStud].sex;
            check = true;
            break;
        case 5:
            cout << "\nEnter a new student learning format
(day/evening/distance): ";
            cin >> student[numbStud].depart;
            check = true;
            break;
        case 6:

```



```

        cout << "\nEnter new grades for the session: ";
        for (int i = 0; i < 8; i++)
        {
            cin >> student[numbStud].term[i];
            if (student[numbStud].term[i] == 2) { temp =
true; } //да, двойка имеется
        }
        if (temp) { cout << "The student has deuces, so he
will be removed.\n"; }
        check = true;
        break;
    default:
        cout << "\nYou didn't choose anything. Exit the
editor. ";
    }
    //изменения внесены

    //вводим в файл
    if (check)
    {
        /*Попытка создать файл с введёнными данными*/
        ofstream fout("students.txt");
        if (!fout.is_open()) { cout << '\n' << "Saving
error!"; }

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

                if (i == numbStud)
                {
                    if (!temp) {
                        /* == Вывод записи == */
                        fout << student[i].fullName <<
"\n";

                        fout << student[i].group <<
"\n";

                        fout << student[i].numberList
<< "\n";

                        fout << student[i].sex <<
"\n";

                        fout << student[i].depart <<
"\n";

                        for (int j = 0; j < 8; j++)

```

```

        {
            fout <<

student[i].term[j] << " ";

        }
        fout << "\n";

        //запоминание даты считывания
        struct tm* loctime;
        time_t curtime;
        string dateChange;
        time(&curtime);
        loctime = localtime(&curtime);
        dateChange = asctime(loctime);
        dateChange =

DelSpaces(dateChange);

        fout << dateChange;
    }
    cout << "Information changed
successfully\n";
}
else
{
    /* == Вывод записи == */
    fout << student[i].fullName <<

"\n";

    fout << student[i].group << "\n";
    fout << student[i].numberList <<

"\n";

    fout << student[i].sex << "\n";
    fout << student[i].depart << "\n";
    for (int j = 0; j < 8; j++)
    {
        fout << student[i].term[j] <<

" ";

    }

}

}
fout.close();

```

```

        }
        //конец вывода в файл
        delete[] student;
    }
}

int main()
{
    int sw = -1;
    while (sw != 9)
    {
        cout << "\n\nChoose what you want to do: \n";
        cout << "1. Create a new student record\n";
        cout << "2. Make changes to an existing record\n";
        cout << "3. Display all student data\n";
        cout << "4. Output all data about students who meet the
condition\n";
        cout << "5. Display the top students by average grade per
session\n";
        cout << "6. Print the number of students who meet the
condition\n";
        cout << "7. Display information about students depending on
the time when the entry was made\n";
        cout << "9. Exit\n";
        cout << "Please enter the number corresponding to the action
you want to perform: ";
        while (!(cin >> sw))
        {
            cin.clear();
            cout << "\nIncorrect input" << endl;
            cout << "Enter a number: ";
            string c;
            getline (cin, c);
        }
        switch (sw)
        {
        case 1:
            newStudent(); //новая запись о студенте
            break;
        case 2:
            changeFile(); //изменение записей
            break;
        case 3:

```

```

        outputStudents(); //вывод всех данных
        break;
    case 4:
        cout << "\n1. Output of information about all students
of group N\n";
        cout << "2. Output of data on students who do not
receive a scholarship\n";
        cout << "3. Output of data on students who study only on
\"good\" and \"excellent\"\n";
        cout << "4. Output of data on students who study only on
\"excellent\"\n";
        cout << "5. Output data about students who have a number
in the list k.\n";
        cout << "0. Return back.\n";
        cout << "Please enter the number corresponding to the
action you want to perform: ";
        int sw4;
        cin >> sw4;
        switch (sw4)
        {
            case 1:
                unsigned short int n;
                cout << "\nPlease enter the group number: ";
                cin >> n;
                groupN(n); //вывод данных студентов группы N
                break;
            case 2:
                notStipend(); //вывод данных студентов без
СТИПЕНДИИ
                break;
            case 3:
                B_GradeTerm(); //вывод данных студентов-хорошистов
                break;
            case 4:
                excellentTerm(); //вывод данных студентов-
ОТЛИЧНИКОВ
                break;
            case 5:
                unsigned short int k;
                cout << "\nPlease enter the student's number in the
list: ";
                cin >> k;
                numberListK(k); //вывод данных студентов номера k
                break;

```

```

        default:
            break;
    }
    break;
case 5:
    topTerm(); //вывод топа студентов по средней оценке
    break;
case 6:
    cout << "\n1. Output of the number of male and female
students\n";
    cout << "2. Output of the number of students who will
receive the scholarship\n";
    cout << "0. Return back.\n";
    cout << "Please enter the number corresponding to the
action you want to perform: ";
    int sw6;
    cin >> sw6;
    switch (sw6)
    {
    case 1:
        F_and_M(); // количество F и M
        break;
    case 2:
        stipend(); // количество студентов со стипендией
        break;
    default:
        break;
    }
    break;
case 7:
    cout << "\n1. Output of all entries made on the day you
enter\n";
    cout << "2. Output of all entries made after noon\n";
    cout << "3. Output of all entries made before noon\n";
    cout << "0. Return back.\n";
    cout << "Please enter the number corresponding to the
action you want to perform: ";
    int sw7;
    cin >> sw7;
    switch (sw7)
    {
    case 1:
    {

```

```

        cout << "\nEnter the date in the format mmm dd yyyy
(for example, Mar 6 2020): ";
        string userDate;
        char trash;
        cin >> trash;
        getline(cin, userDate);
        userDate = trash + userDate;
        thatDate(userDate); // вывод всех записей,
сделанных в этот день
        break;
    }
    case 2:
        afterNoon(); // записи после полудня
        break;
    case 3:
        tillNoon(); // записи до полудня
        break;
    default:
        break;
    }
    break;
default:
    break;
}

system("Pause");
return 0;
}

```