Московский Авиационный Институт
(Национальный Исследовательский Университет)
Факультет прикладной математики и физики
Кафедра вычислительной математики и программирования

КУРСОВАЯ РАБОТА

по курсу
"Информатика"
"Архитектура ЭВМ, системное программное обеспечение"
II семестр
Задание 6 "Обработка последовательной файловой структуры на языке СИ"

Студент: Пашкевич А. Р.
Группа: 08-107, № по списку 10

Руководитель: Ридли М. К.
Ридли А. Н.

Оценка: _____

Дата: _____

Подпись: _____

Москва, 2017

АННОТАЦИЯ

В данном документе описывается программа, написанная в соответствии с постановкой задачи на курсовое проектирование по теме "Обработка последовательной файловой структуры на языке СИ" по дисциплине "Архитектура ЭВМ, системное программное обеспечение". Данная программа предназначена для обработки информации, содержащей сведения о студентах (ФИО, код группы, оценки по экзаменам и зачетам) генерации и вывода на экран и в бинарный файл таблиц. Входными данными является текстовый файл с данными об успеваемости студентов. Для проверки работы программы разработан тестовый пример. Результаты тестирования доказывают, что программа правильно выполняет все операции по обработке входных данных и формирования выходных данных.

1. ПОСТАНОВКА ЗАДАЧИ

1) Разработать последовательную структуру данных для представления простейшей базы данных на файлах в СП Си в соответствии с заданным вариантом.

2) Составить программу генерации внешнего нетекстового файла заданной структуры, содержащего представительный набор записей (не более 20).

3) Распечатать содержимое сгенерированного файла в виде таблицы.

4) Выполнить над ним заданное действие и распечатать результат. Действие по выборке данных из файла оформить в виде отдельной программы с параметрами запроса, получаемыми из командной строки UNIX. Имя файла с бинарными данными является обязательным параметром второй программы.

База данных содержит информацию об успеваемости студентов данной группы по всем предметам: Фамилия, инициалы, пол, номер группы, оценки по экзаменам и зачетам. В соответствии с заданным вариантом требуется напечатать список потенциальных стипендиатов – студентов, у которых одна 3, а все остальные оценки 4 и 5 или все 5 и одна 4.

2. МЕТОД РЕАЛИЗАЦИИ

1) Создать набор тестовых файлов по 20 записей в каждом.

2) Использовать команды обработки текстовых файлов ОС UNIX и переадресацию ввода-вывода.

3) Составить программу генеррирующую внешний нетекстовый файл.

4) Перенаправить все содержимое нетекстового представления в файл out.db.

5) Распечатать содержимое файла в виде таблицы.

6) Сравнить при помощи условия if (){…} значение параметра p и данные об оценках каждого студента. Если оценки удовлетворяют условию p, то выводим всю информацию об этом студенте.

3. ФУНКЦИОНАЛЬНОЕ НАЗНАЧЕНИЕ

Данная программа, преобразует текстовой файл в бинарный, генерирует внешний бинарный файл, распечатывает таблицу. Работа с бинарными файлами очень удобна. Во-первых, хранить в памяти компьютера бинарные коды намного проще, занимает меньший объем памяти. Во-вторых, с файлом бинарном виде гораздо быстрее и удобнее производить всевозможные операции. К тому же такие файлы хранится в оперативной памяти в том виде, в каком был создан файл. Таким образом, нетекстовое представление обладает большими преимуществами.

4. ОБЩИЕ СВЕДЕНИЯ О ПРОГРАММЕ

Необходимое программное и аппаратное обеспечение: компилятор gcc

Операционная система: GNU/Linux, UNIX, MS Windows

Язык: Си

Система программирования: Си

Способ вызова и загрузки: bash

4. ОРГАНИЗАЦИЯ ВХОДНЫХ И ВЫХОДНЫХ ДАННЫХ

Входные данные передаются в программу следующим способом: Чтение входных данных из предварительно подготовленного текстового файла, данные в котором хранятся в виде таблицы. Файл, может включать в себя неизвестное заранее количество строк, но должен иметь определенную структуру.

Имя входного файла по умолчанию test1.dat, а местонахождение — каталог data.

Содержимое и структура файла включает:
1) Фамилия студента
2) Имя
3) Отчество
4) Пол
5) Группа
6) Оценки по по 4 предметам 2 — 5.
7) Результаты по 3 зачетам (сдал не сдал)

Пример входного текстового файла:

```
Чистяков Андрей Сергеевич 0 08-103 4 4 4 4 PS PS PS
Казначеева Екатерина Сергеевна 1 08-103 5 5 5 5 PS PS PS
Еловский Даниил Михайлович 0 08-103 3 3 3 3 PS PS PS
Егоров Евгений Юрьевич 0 08-103 4 4 4 4 PS PS PS
Шерстобитова Софья Викторовна 1 08-103 4 4 4 4 PS PS PS
Нефедова Арина Борисовна 1 08-103 4 5 4 4 PS PS PS
Поморцев Данила Аркадьевич 0 08-103 5 3 3 4 PS PS PS
Перфильев Сергей Дмитриевич 0 08-103 3 2 4 2 PS PS PS
Сапин Сергей Сергеевич 0 08-103 3 4 4 5 PS PS PS
Юрманова Вероника Павловна 1 08-103 3 4 3 3 PS PS PS
Лебедева Анастасия Павловна 1 08-103 4 4 4 5 PS PS PS
Калинин Сергей Евгеньевич 0 08-103 4 4 4 3 PS PS PS
Круглова Светлана Андреевна 1 08-103 4 5 5 4 PS PS PS
Адильбеков Алтынбек Асылбекович 0 08-103 3 3 3 3 PS PS PS
Лымаренко Александра Руслановна 1 08-103 5 4 4 4 PS PS PS
Ефимов Александр Валерьевич 0 08-103 3 3 3 3 PS PS PS
Измайлов Сергей Александрович 0 08-103 2 4 3 2 PS PS PS
Захаревич Павел Андреевич 0 08-103 3 3 3 3 PS PS PS
Селезнев Владислав Сергеевич 0 08-103 4 5 5 5 PS PS PS
Рудик Сергей Олегович 0 08-103 3 5 4 4 PS PS PS
```

Разделителями являются символы пробела, табуляции или конца строки.

Выходные данные:
Чистяков Андрей Сергеевич 0 08-103 {4, 4, 4, 4} {1, 1, 1} { 0, 0, 4, 0, 3}

5. ВНУТРЕННЕЕ ПРЕДСТАВЛЕНИЕ ДАННЫХ

Таблица 1: Структура данных:

Тип данных	Имя поля	Размер	Описание
size_t	id		ключ целое число
char	surname	32	Фамилия студента, строка
char	name	32	Имя, строка

char	patronymic	32	Отчество, строка
enum	gender		MALE = 0, FEMALE = 1
char	group	8	Текстовое поле
int	grades	4	Одномерный массив, содержащий оценки по предметам
int	tests	3	Одномерный массив, содержащий результаты зачетов по предметам (0 — не сдал, 1 - сдал)
int	credit	5	Одномерный целочисленный массив, содержащий количество оценок 2, 3, 3, 5 и количество сданных тестов

```
enum gender_t{
    MALE,
    FEMALE
};

typedef struct __student__ {
    size_t id;
    char surname[32];
    char name[32];
    char patronymic[32];
    enum gender_t gender;
    char group[8];
    int grades[4];
    int tests[3];
    int credit[5];
} student_t;
```

3. ОРГАНИЗАЦИЯ ИСХОДНОГО КОДА

Код организован следующим образом:

Управляющая программа, обеспечивающая ввод данных, реализована в **generate.c**, а реализация структуры и функции с ней – в файлах **student.h** и **student.c**.

Вывод содержимого выходного файла и обработка данных реализована в файле **execute.c**.

Таблица 2: Структура файлов проекта:

/include	
student.h	Заголовочный файл с описанием структуры записи о студенте
/src	
generate.c	Открытие входного файла. Инициация структуры **student** Генерация файла ***.db** из текстового файла. Вывод таблицы со считанными данными.
student.c	Файл с кодом для работы со структурой **student**. Считывает информацию о студенте из файла ***.dat**. Возвращает код ошибки или **1** (единицу) если все хорошо. Ситуации нулевых указателей и конца файла обрабатываются. Аналогично, обрабатываются проблемы при считывании какого-либо поля структуры. Вывод структуры в **stdout** в виде таблицы.
execute.c	Обработка в соответствии с заданными условиями данных из

	файла *.db и вывод результата. Вывод содержимого файла *.db в виде таблицы
makefile	Используется для управления сборкой проекта. Служит для сборки проекта и очистки каталогов проекта от образованных в процессе сборки файлов.

4. КАК ЭТО РАБОТАЕТ

Программа **generate** запускается с двумя дополнительными аргументами: имя входного текстового файла и имя выходного бинарного файла.

Если программа была запущена без этих аргументов, она работает с файлами, заданными по умолчанию (*data/test1.dat* и *data/student.db*).

Если оба файла открыты успешно, запускается функция student_fscanf, которая считывает информацию из входного текстового файла в соответствующие поля структуры student, и в случае успеха возвращает "1" (единицу).

Если функция ***student_fscanf()*** вернула 1, происходит запись структуры в бинарный файл. Запись осуществляется с помощью стандартной функции **fwrite**.

Программа **execute** запускается отдельно с двумя аргументами в виде имени бинарного файла *.db и параметром, задающим условия выборки студентов:

параметр **-p1** выводит список студентов, имеющих одну тройку, а остальные оценки 4 и 5;

параметр **-p2** — выводит таблицу с данными о студентах у которых одна 4, а остальные 5;

параметр **-f** — выводит таблицу с данными о всех имеющихся в файле *.db студентах.

```
$ ./generate data/test5.dat data/out.db
File data/test5.dat open for reading...
File data/out.db open for writing...

Table 1: Generated data.
+-----+------------------+------------------+------------------+------------+------------+------------------+
| ID  | SURNAME          | NAME             | PATRONYMIC       | GENDER     | GROUP      | GRADES           |
+-----+------------------+------------------+------------------+------------+------------+------------------+
|   1 | Шентяпин         | Владислав        | Валерьевич       | Male       | 08-105     | 5 5 5 5 PS PS PS |
|   2 | Прокофьева       | Анастасия        | Владимировна     | Female     | 08-105     | 4 4 4 5 PS PS PS |
|   3 | Севостьянова     | Анастасия        | Сергеевна        | Female     | 08-105     | 4 4 4 4 PS PS PS |
|   4 | Хаустов          | Алексей          | Владиславович    | Male       | 08-105     | 4 2 3 4 PS PS PS |
|   5 | Доможиров        | Владислав        | Александрович    | Male       | 08-105     | 2 5 3 3 PS PS PS |
|   6 | Мараховский      | Николай          | Александрович    | Male       | 08-105     | 2 3 4 4 PS PS PS |
|   7 | Куликова         | Светлана         | Евгеньевна       | Female     | 08-105     | 4 4 3 4 PS PS PS |
|   8 | Марулин          | Никита           | Юрьевич          | Male       | 08-105     | 3 3 4 3 PS PS PS |
|   9 | Ярославцев       | Илья             | Сергеевич        | Male       | 08-105     | 4 3 5 4 PS PS PS |
|  10 | Ким              | Данил            | -                | Male       | 08-105     | 5 4 4 4 PS PS PS |
|  11 | Чуев             | Денис            | Сергеевич        | Male       | 08-105     | 4 4 3 4 PS PS PS |
|  12 | Елисеев          | Сергей           | Николаевич       | Male       | 08-105     | 3 3 3 4 PS PS PS |
|  13 | Кочмарик         | Нина             | Николаевна       | Female     | 08-105     | 3 4 5 4 PS PS PS |
|  14 | Жук              | Дарья            | Васильевна       | Female     | 08-105     | 2 5 4 4 PS PS PS |
|  15 | Короткова        | Мария            | Игоревна         | Female     | 08-105     | 4 5 5 5 PS PS PS |
|  16 | Рябков           | Даниил           | Владимирович     | Male       | 08-105     | 4 4 4 4 PS PS PS |
|  17 | Свеженцев        | Владимир         | Иосифович        | Male       | 08-105     | 3 3 4 3 PS PS PS |
|  18 | Быков            | Федор            | Викторович       | Male       | 08-105     | 2 2 2 2 PS FL FL |
|  19 | Худайберенов     | Максим           | Батырович        | Male       | 08-105     | 4 3 3 2 PS PS PS |
|  20 | Кротов           | Родион           | Евгеньевич       | Male       | 08-105     | 4 4 3 4 PS PS PS |
+-----+------------------+------------------+------------------+------------+------------+------------------+
```

```
$ ./execute data/out.db -p1
File data/out.db open for reading...

Table 3: Scholarship recipients.
+-----+------------------+------------------+------------------+------------+------------+------------------+
| ID  | SURNAME          | NAME             | PATRONYMIC       | GENDER     | GROUP      | GRADES           |
+-----+------------------+------------------+------------------+------------+------------+------------------+
|   1 | Шентяпин         | Владислав        | Валерьевич       | Male       | 08-105     | 5 5 5 5 PS PS PS |
|   2 | Прокофьева       | Анастасия        | Владимировна     | Female     | 08-105     | 4 4 4 5 PS PS PS |
|   3 | Севостьянова     | Анастасия        | Сергеевна        | Female     | 08-105     | 4 4 4 4 PS PS PS |
|   7 | Куликова         | Светлана         | Евгеньевна       | Female     | 08-105     | 4 4 3 4 PS PS PS |
|   9 | Ярославцев       | Илья             | Сергеевич        | Male       | 08-105     | 4 3 5 4 PS PS PS |
|  10 | Ким              | Данил            | -                | Male       | 08-105     | 5 4 4 4 PS PS PS |
|  11 | Чуев             | Денис            | Сергеевич        | Male       | 08-105     | 4 4 3 4 PS PS PS |
|  13 | Кочмарик         | Нина             | Николаевна       | Female     | 08-105     | 3 4 5 4 PS PS PS |
|  15 | Короткова        | Мария            | Игоревна         | Female     | 08-105     | 4 5 5 5 PS PS PS |
|  16 | Рябков           | Даниил           | Владимирович     | Male       | 08-105     | 4 4 4 4 PS PS PS |
|  20 | Кротов           | Родион           | Евгеньевич       | Male       | 08-105     | 4 4 3 4 PS PS PS |
+-----+------------------+------------------+------------------+------------+------------+------------------+

$ ./execute data/out.db -p2
File data/out.db open for reading...

Table 3: Scholarship recipients.
+-----+------------------+------------------+------------------+------------+------------+------------------+
| ID  | SURNAME          | NAME             | PATRONYMIC       | GENDER     | GROUP      | GRADES           |
+-----+------------------+------------------+------------------+------------+------------+------------------+
|   1 | Шентяпин         | Владислав        | Валерьевич       | Male       | 08-105     | 5 5 5 5 PS PS PS |
|  15 | Короткова        | Мария            | Игоревна         | Female     | 08-105     | 4 5 5 5 PS PS PS |
+-----+------------------+------------------+------------------+------------+------------+------------------+

$ ./execute data/out.db -f
File data/out.db open for reading...

Table 2: Print data.
+-----+------------------+------------------+------------------+------------+------------+------------------+
| ID  | SURNAME          | NAME             | PATRONYMIC       | GENDER     | GROUP      | GRADES           |
+-----+------------------+------------------+------------------+------------+------------+------------------+
|   1 | Шентяпин         | Владислав        | Валерьевич       | Male       | 08-105     | 5 5 5 5 PS PS PS |
|   2 | Прокофьева       | Анастасия        | Владимировна     | Female     | 08-105     | 4 4 4 5 PS PS PS |
|   3 | Севостьянова     | Анастасия        | Сергеевна        | Female     | 08-105     | 4 4 4 4 PS PS PS |
|   4 | Хаустов          | Алексей          | Владиславович    | Male       | 08-105     | 4 2 3 4 PS PS PS |
|   5 | Доможиров        | Владислав        | Александрович    | Male       | 08-105     | 2 5 3 3 PS PS PS |
|   6 | Мараховский      | Николай          | Александрович    | Male       | 08-105     | 2 3 4 4 PS PS PS |
|   7 | Куликова         | Светлана         | Евгеньевна       | Female     | 08-105     | 4 4 3 4 PS PS PS |
|   8 | Марулин          | Никита           | Юрьевич          | Male       | 08-105     | 3 3 4 3 PS PS PS |
|   9 | Ярославцев       | Илья             | Сергеевич        | Male       | 08-105     | 4 3 5 4 PS PS PS |
|  10 | Ким              | Данил            | -                | Male       | 08-105     | 5 4 4 4 PS PS PS |
|  11 | Чуев             | Денис            | Сергеевич        | Male       | 08-105     | 4 4 3 4 PS PS PS |
|  12 | Елисеев          | Сергей           | Николаевич       | Male       | 08-105     | 3 3 3 4 PS PS PS |
|  13 | Кочмарик         | Нина             | Николаевна       | Female     | 08-105     | 3 4 5 4 PS PS PS |
|  14 | Жук              | Дарья            | Васильевна       | Female     | 08-105     | 2 5 4 4 PS PS PS |
|  15 | Короткова        | Мария            | Игоревна         | Female     | 08-105     | 4 5 5 5 PS PS PS |
|  16 | Рябков           | Даниил           | Владимирович     | Male       | 08-105     | 4 4 4 4 PS PS PS |
|  17 | Свеженцев        | Владимир         | Иосифович        | Male       | 08-105     | 3 3 4 3 PS PS PS |
|  18 | Быков            | Федор            | Викторович       | Male       | 08-105     | 2 2 2 2 PS FL FL |
|  19 | Худайберенов     | Максим           | Батырович        | Male       | 08-105     | 4 3 3 2 PS PS PS |
|  20 | Кротов           | Родион           | Евгеньевич       | Male       | 08-105     | 4 4 3 4 PS PS PS |
+-----+------------------+------------------+------------------+------------+------------+------------------+
```

ЗАКЛЮЧЕНИЕ

Данная программа, составлена в соответствии с постановкой задачи на курсовое проектирование по теме "Обработка последовательной файловой структуры на языке СИ" по дисциплине "Архитектура ЭВМ, системное программное обеспечение". При написании программы использованы методические указания по курсовому проектированию по дисциплине. Для проверки работоспособности программы и правильности обработки входных данных разработан тестовый пример. Тестирование программы подтвердило, что программа правильно выполняет обработку данных и выдаёт верные результаты. Всё это свидетельствует о работоспособности программы и позволяет сделать вывод о пригодности программы к решению практических задач по обработке информации о студентах.

student.h

```
#ifndef __STUDENT_H__
#define __STUDENT_H__

#include <stdarg.h>
#include <stdio.h>
#include <stdlib.h>
#include <string.h>

enum gender_t{
    MALE,
    FEMALE
};

typedef struct __student__ {
    size_t id;
    char surname[32];
    char name[32];
    char patronymic[32];
    enum gender_t gender;
    char group[8];
    int grades[4];
    int tests[3];
    int credit[5];
} student_t;

student_t student_init(void);

int student_fscanf(FILE *, student_t *);
int student_print(student_t *);

#endif
```

student.c

```
#include "student.h"

//  считываем информацию о студенте из файла f
int student_fscanf(FILE* f, student_t* member){

    if(!f){
        printf("f is null pointer\n");
        return 0;
    }

    if(feof(f)){
        return EOF;
    }

    if(!member) {
        printf("member is null\n");
        return -2;
    }
    if (fscanf(f, "%32s", (member->surname))!=1) {
        return -3;
    }

    if (fscanf(f, "%32s", (member->name))!=1) {
        return -4;
    }

    if (fscanf(f, "%32s", (member->patronymic))!=1){
        return -5;
    }

    int temp;
    if (fscanf(f, "%d", &temp)!=1){
        return -6;
    } else member->gender = temp;

    if (fscanf(f, "%8s", (member->group))!=1){
        return -7;
    }

    if(fscanf(f,"%1d%1d%1d%1d",(int *)&(member->grades[0]),(int *)&(member->grades[1]),(int *)&(member-
>grades[2]),(int *)&(member->grades[3]))!=4){
```

```
        return -8;
    }
    int test_cost = 0;
    for (int i=0; i<3; ++i) {
        char test[2];
        if (fscanf(f, "%2s", test)!=1) {
            return -9;
        } else {
            if (strcmp(test,"FL") == 0){
                member->tests[i] = 0;
            } else {
                member->tests[i] = 1;
                test_cost++;
            }
        }
    }
    int unsatisfactory=0, passing_grade=0, satisfactory=0, excellent=0;
    for(int i = 0 ; i < 4 ; ++i) {
        if (member->grades[i] == 2) unsatisfactory++;
        if (member->grades[i] == 3) passing_grade++;
        if (member->grades[i] == 4) satisfactory++;
        if (member->grades[i] == 5) excellent++;
    }
    member->credit[0]=unsatisfactory;
    member->credit[1]=passing_grade;
    member->credit[2]=satisfactory;
    member->credit[3]=excellent;
    member->credit[4]=test_cost;

    return 1;
}
// выводит строку таблицы в stdout
int student_print(student_t* member){
    fprintf(stdout, "| %3lu | ",member->id);
    fprintf(stdout, "%-*s | ",(int)((32 - strlen(member->surname)) / 2 + strlen(member->surname)),member-
>surname);
    fprintf(stdout, "%-*s | ",(int)((32 - strlen(member->name)) / 2 + strlen(member->name)),member->name);
    fprintf(stdout, "%-*s | ",(int)((32 - strlen(member->patronymic)) / 2 + strlen(member-
>patronymic)),member->patronymic);
    char *temp = (member->gender ? "Female" : "Male  ");
    fprintf(stdout, "%-*s | ",(int)((14 - strlen(temp)) / 2 + strlen(temp)),temp);
    fprintf(stdout, "%-*s | ",(int)((14 - strlen(member->group)) / 2 + strlen(member->group)),member-
>group);
    for(int i = 0 ; i < 4 ; i++) {
        fprintf(stdout, "%-1d ", member->grades[i]);
    }
    for(int i = 0 ; i < 3 ; i++) {
        printf(member->tests[i] ? "PS " : "FL ");
    }
    /*for(int i = 0 ; i < 4 ; i++) {
        fprintf(stdout, "%-1d ", member->credit[i]);
    }*/
    fprintf(stdout, "|\n");
    return 1;
}
```

generate.c

```
#include "student.h"

#define DEFAULT_DATABASE_FILENAME "data/student.db";
#define DEFAULT_TEST_FILENAME "data/test1.dat";

int main(int argc, char *argv[]) {
    char *db_fname;
    FILE *out = NULL;
    FILE *pfile = NULL;
    char *in_fname;
    size_t row = 0;
    student_t student;

    if (argc == 3) {
        in_fname = argv[1];
        db_fname = argv[2];
    } else {
        db_fname = DEFAULT_DATABASE_FILENAME;
        in_fname = DEFAULT_TEST_FILENAME;
    }

    /* Open the file for reading */
    if(!(pfile = fopen(in_fname, "r"))) {
```

```
        printf("Error opening %s for reading. Program terminated.\n", in_fname);
        exit(1);
    } else printf("File %s open for reading...\n", in_fname);

    if(!(out = fopen(db_fname, "wb"))) {  // Create a new file we can write
        printf("Error opening %s for writing. Program terminated.\n", db_fname);
        exit (1);
    } else printf("File %s open for writing...\n", db_fname);

    printf("\nTable 1: Generated data.\n");
    printf("+-----+------------------+------------------+------------------+------------+------------
+------------------+\n");
    printf("| ID  | SURNAME          | NAME             | PATRONYMIC       | GENDER     | GROUP      |
GRADES           |\n");
    printf("+-----+------------------+------------------+------------------+------------+------------
+------------------+\n");
    while (student_fscanf(pfile, &student) == 1) {
        student.id = ++row;
        fwrite(&student, sizeof(student), 1, out);
        fflush(out);
        student_print(&student);
    }
    printf("+-----+------------------+------------------+------------------+------------+------------
+------------------+\n\n");
    fclose(out); // file close
    fclose(pfile); // Close the file

    return 0;
}
```

execute.c

```
#include "student.h"
//Программа загружает данные из бинарного файла

int main(int argc, char* argv[])
{
    char *fname;
    FILE *in = NULL;
    student_t m;
    int condition = 0;

    if (argc == 3) {
        fname = argv[1];
        if(!(in = fopen(fname, "r"))) {
            printf("Error opening %s for reading. Program terminated.\n", fname);
            exit(1);
        } else {
            printf("File %s open for reading...\n", fname);
            fseek(in, 0, SEEK_SET);
        }
    } else {
        printf("Invalid number of parameters\n");
        exit(1);
    }
    if (strcmp(argv[2],"-f") == 0) {
        printf("\nTable 2: Print data.\n");
        printf("+-----+------------------+------------------+------------------+------------+------------
+------------------+\n");
        printf("| ID  | SURNAME          | NAME             | PATRONYMIC       | GENDER     | GROUP      |
GRADES           |\n");
        printf("+-----+------------------+------------------+------------------+------------+------------
+------------------+\n");
        while (fread(&m, sizeof(m), 1, in) == 1) {
            student_print(&m);
        }
        printf("+-----+------------------+------------------+------------------+------------+------------
+------------------+\n\n");
    } else if (strcmp(argv[2],"-p1") == 0) {
        condition = 1;
    } else if (strcmp(argv[2],"-p2") == 0) {
        condition = 2;
    } else {
        printf("The second parameter may be only -f, -p1 or -p2 \n");
        exit(1);
    }
    if (condition) {
        printf("\nTable 3: Scholarship recipients.\n");
        printf("+-----+------------------+------------------+------------------+------------+------------
+------------------+\n");
        printf("| ID  | SURNAME          | NAME             | PATRONYMIC       | GENDER     | GROUP      |
```

```
GRADES           |\n");
        printf("+-----+------------------+------------------+------------------+------------+------------
+------------------+\n");
        while (fread(&m, sizeof(m), 1, in) == 1) {
            if (condition == 1) {
                if (m.credit[0]==0 && m.credit[1]<2 && m.credit[4]==3) {
                    student_print(&m);
                }
            } else {
                if (m.credit[0]==0 && m.credit[1]==0 && m.credit[2]<2 && m.credit[4]==3) {
                    student_print(&m);
                }
            }
        }
        printf("+-----+------------------+------------------+------------------+------------+------------
+------------------+\n\n");
    }
    fclose(in); // Close the file
    return 0;
}
```