Московский Авиационный Институт

(Национальный Исследовательский Университет)

Факультет информационных технологий и прикладной математики

Кафедра вычислительной математики и программирования

**Курсовая работа**

**по курсу «Языки и методы программирования»**

**II Семестр**

**Задание 6:**

**Обработка последовательной файловой структуры на языке Си**

|  |  |
| --- | --- |
| Студент: | Почечура А.А. |
| Группа: | М8О-106Б-20 |
| Преподаватель: | Дубинин А.В. |
|  |  |
| Оценка: |  |
| Дата: |  |
| Подпись: |  |

**Москва, 2021 г.**

**Содержание**

[1. Введение 3](#_Toc67586040)

[2. Работа с файлами 3](#_Toc67586041)

[3. Постановка задачи 8](#_Toc67586045)

[4. Общий метод решения 9](#_Toc67586046)

[5. Общие сведения о программе 9](#_Toc67586047)

[6. Описание логической структуры 10](#_Toc67586048)

[7. Таблица переменных и констант 10](#_Toc67586049)

[8. Входные данные 11](#_Toc67586050)

[9. Содержание файлов 11](#_Toc67586051)

[10. Демонстрация работы программы 19](#_Toc67586052)

[11. Заключение 21](#_Toc67586053)

[12. Список используемых источников 21](#_Toc67586054)

# 1. Введение

Долговременные файлы – неотъемлемая часть любой операционной системы, именно в них хранится большинство данных пользователей - фото, видео и другие документы, так как поместить всё в оперативную память невозможно из-за её низкого объёма и того, что она энергозависима, что ставит всю долговременность файлов под вопрос.

Так что все компьютеры должны уметь работать с этими файлами. Однако, компьютеры бывают разные – у них разные архитектуры, файловые системы и протоколы подключения к самим устройствам внешней памяти, на которых эти файлы и лежат.

# 2. Работа с файлами

Работа с файлами на самом деле представляет собой запрос ядру операционной системы на осуществление какого-либо действия путём отправления ему системного вызова.

Open() – открыть файл – системный вызов, при вызове которого ОС проверяет, существует ли файл, есть ли у пользователя необходимые права и можно ли его прочитать. После всех проверок, этому файлу присваивается файловый дескриптор - это неотрицательное целое число, через которое можно впоследствии получить доступ к этому файлу.

Когда создается новый поток ввода-вывода, ядро возвращает процессу, создавшему поток ввода-вывода, его файловый дескриптор. По умолчанию Unix-оболочки связывают файловый дескриптор 0 с потоком стандартного ввода процесса (терминал), файловый дескриптор 1 - с потоком стандартного вывода (терминал), и файловый дескриптор 2 - с потоком диагностики (куда обычно выводятся сообщения об ошибках). Это соглашение соблюдается многими Unix-оболочками и многими приложениями — и ни в коем случае не является составной частью ядра.

## Функция fopen()

Файл открывается при помощи функции fopen, которая возвращает информацию потока ввода-вывода, прикреплённого к указанному файлу или другому устройству, с которого идет чтение (или в который идет запись). В случае неудачи функция возвращает нулевой указатель.

Функция fopen по сути представляет собой «обертку» более высокого уровня системного вызова open операционной системы Unix. Аналогично, fclose является оберткой системного вызова Unix close, а сама структура FILE языка Си зачастую обращается к соответствующему файловому дескриптору Unix.

Таких “обёрток” в языке Си предостаточно – помимо обозначенных выше, есть ещё fscanf, fgetc, fwrite, fprintf, getchar. Например, getchar() представляет собой системный вызов read() на поток 0 и может быть заменён функцией fgetc на поток 0.

Параметр mode (режим) для fopen и freopen должен быть строковый и начинаться с одной из следующих последовательностей:

|  |  |
| --- | --- |
| **r** | Открыть текстовый файл для чтения.  Чтение начинается с начала файла. |
| **r+** | Открыть для чтения и записи. Чтение или запись начинаются с начала файла. |
| **w** | "Урезать" файл до нулевой длины или создать текстовый файл и открыть его для записи.  Запись начинается с начала файла. |
| **w+** | Открыть для чтения и записи. Файл создается, если до этого его не существовало, в противном случае он "урезается".  Чтение или запись начинаются с начала файла. |
| **a** | Открыть для дописывания (записи в конец файла). Файл создается, если до этого его не существовало. Запись осуществляется в конец файла. |
| **a+** | Открыть для чтения и дописывания (записи в конец файла). Файл создается, если до этого его не существовало.  Чтение или запись производятся с конца файла. |

Если функции fopen() удалось открыть указанный файл, возвращается указатель на FILE. Если же файл не может быть открыт, возвращается NULL.

Файл может быть открыт либо в двоичном, либо в текстовом режиме. В текстовом режиме возврат каретки и последовательности перевода строки при вводе транслируется в символы «новая строка». При выводе — наоборот: символ новой строки транс­лируются в возврат каретки и перевод строки. Такого рода трансляция не производится при работе с двоичными файлами.

Если строка mode не задает ни b (для двоичных), ни t (для текстовых файлов), то тип открыва­емого файла определяется значением встроенной глобальной переменной \_fmode. По умолчанию \_fmode принимает значение O\_ТЕХТ, что означает текстовый режим. Она может быть установле­на и в O\_BINARY, что означает двоичный режим. Эти макросы определены в fcntl.h. Однако, в последнее время бинарный режим чтения не используется, так как между ним и текстовым чтением нет больше разницы.

Если при операции записи для открытия файла используется функция fopen(), то любой файл, уже имевший такое же имя, будет стерт и на его месте будет создан новый файл. Если файла с таким именем не существовало, он будет создан. Если необходимо дописать что-нибудь в конец файла, то необходимо использовать режим а. Если файл не существует, будет возвращена ошиб­ка. При открытии файла для операций чтения необходимо, чтобы он уже существовал. В против­ном случае будет возвращена ошибка. И наконец, если файл открывается для операций чтения/ записи, то он не стирается, если он уже существует, а если такого файла нет, он создается.

Чтение и запись могут накладываться друг на друга в потоке, открытом для чтения/записи, в любом порядке. Стоит заметить, что ANSI C требует, чтобы между чтением/записью использовались функции позиционирования в файле, пока операция ввода не встретит конец файла. Если это условие не выполняется, то при чтении разрешается возвращать результат, не совпадающий с самым последним изменением. То есть будет хорошим тоном (а иногда и действительно необходимым в Linux) использовать функции fseek() или fgetpos() между операциями чтения и записи в одном потоке. Эти операции могут фактически быть "пустыми" (как, например, *fseek(..., 0L,SEEK\_CUR)*, вызванная для того, чтобы возник ее побочный эффект синхронизации).

## Функция fseek()

Функция fseek() устанавливает позицию в потоке данных, заданным аргументом stream. Относительно установленной позиции будет осуществляться чтение и запись данных.

int fseek(FILE \*stream, long int offset, int whence);  
 Точка отсчета устанавливаемой позиции определяется аргументом whence, который может принимать значения:

* SEEK\_SET – смещение отсчитывается от начала файла
* SEEK\_CUR – смещение отсчитывается от текущей позиции
* SEEK\_END – смещение отсчитывается от конца файла

Смещение задается аргументом offset, причем положительное значение аргумента означает смещения вправо от указанной аргументом whence позиции, отрицательное – смещение влево.

Для двоичных потоков данных, смещение (offset) – это количество байт.

Для текстовых потоков данных смещение (offset) должно быть равно 0, либо получено с помощью функции ftell(), при этом точка отсчета (whence) должна иметь значение SEEK\_SET.

Открытие файла в режиме дописывания (**a** в качестве первого символа *mode*) приводит к тому, что все последующие операции записи в этот поток производятся в конец файла, как если бы перед ними была вызвана функция fseek(stream, 0, SEEK\_END);

## Функция fread()

Функция fread() считывает count объектов — каждый объект по size символов в длину — из потока, указанного stream, и помещает их в символьный массив, указанный в buf. Указатель позиции в файле продвигается вперед на количество считанных символов.

size\_t fread(void \*buf, size\_t size, size\_t count, FILE \*stream);

Функция fread() возвращает количество действительно считанных объектов. Если количество считанных объектов меньше, чем это указано при вызове, то либо произошла ошибка, либо был достигнут конец файла. Чтобы определить, что именно имело место, нужно использовать feof() или ferror().

# 3. Постановка задачи

Разработать последовательную структуру данных для представления простейшей базы данных на файлах в СП СИ в соответствии с заданным вариантом. Составить программу генерации внешнего нетекстового файла заданной структуры, содержащего представительный набор записей (15-20). Выполнить над файлом заданные действия: вывести название команды с наибольшим количеством побед её игроков.

Действие по выборке данных из файла оформить в виде отдельной программы с параметрами запроса, вводимыми из стандартного входного тестового файла, или получаемыми из командной строки UNIX. Параметры задаются с помощью ключей -f (печать) и -p (параметры конкретного варианта задания).

**Содержимое и структура файла:**

Информация об игроках: “id игрока|никнейм игрока|команда|страна|призовые|

количество матчей|процент побед|”

**Действиe:**

Вывести название команды с наибольшим количеством побед и её игроков с количеством призовых, равным NUM или больше (NUM задаётся после ключа –p).

# 4. Общий метод решения

Работа идет с двумя программами СИ, одним заголовочным файлом (содержит описание структуры), текстовым файлом (хранятся данные) и бинарным файлом (туда записываются данные из текстового файла).

Первая программа читает данные из текстового файла и записывает их в бинарный.

Вторая программа производит печать данных из бинарного файла в виде таблицы (если указан ключ -f), или результат выполнения поставленной вариантом задачи (ключ -p).

# 5. Общие сведения о программе

Программное и аппаратное обеспечения для запуска данной программы на ПК не ограничено в выборе. Операционная система семейства Linux — Ubuntu. Язык программирования Си.

Имена файлов:

1) prog1.c;

2) gamers.txt;

3) bin.bin;

4) prog2.c;

5) vector.h;

6) statistics.h;

7) team\_victories.h;

8) vector\_functions.c.

# 6. Описание логической структуры

prog1.c:

Текстовой файл gamers.txt открывается на чтение с помощью функции fopen, текстовый файл bin.bin открывается на запись с помощью той же функции; затем в цикле с помощью функций fscanf, fwrite, пока не конец файла, происходит запись данных из текстового файла в бинарный файл bin.bin. В конце переписи данных файлы gamers.txt и bin.bin закрываются функцией fclose.

prog2.c:

Открывается bin.bin. При введённом ключе “-p” выполняется задание варианта, а именно: во время первого прохода в вектор записываются команды и их суммарные победы, которые подсчитываются во время записи. Запись команд без повторений происходит благодаря прохождению по вектору и поиску одинаковых команд. Далее находится команда с максимальным количеством побед и происходит второй проход по файлу bin.bin, во время которого находятся игроки соответствующей команды. Также игроки проходят соответствующую проверку на условие ключа –p: сравнивается количество их призовых с указанным после ключа. При введённом ключе “-f” выводится таблица игроков со всеми данными (текстовый аналог входного файла).

# 7. Таблица переменных и констант

Описание переменных структуры в statistics.h:

|  |  |
| --- | --- |
| int id; | Id игрока |
| char gamer[20]; | Ник игрока |
| char team[50]; | Команда игрока |
| char country[30]; | Страна |
| int prize\_money; | Призовые |
| int matches; | Количество сыгранных матчей |
| int proportion\_of\_vict; | Процент побед |

Описание переменных структуры в team\_victories.h:

|  |  |
| --- | --- |
| char team[30]; | Команда |
| int victories; | Количество побед |

В программе prog1.c происходит работа со структурой statistics.h, в программе prog2.c – со структурами team\_victories.h и statistics.h.

# 8. Входные данные

1. для prog1.c:

первый аргумент - текстовый файл для чтения с данными о игроках;

второй аргумент - бинарный файл для записи;

1. для prog2.c:

первый аргумент - бинарный файл с данными о студентах;

последующие аргументы – ключи (-p или -f) (для -p необходим параметр);

**Выходные данные**

1. Вывод таблицы (ключ -f);

2. Вывод таблицы полей, соответствующих заданию (ключ -p)

# 9. Содержание файлов

**gamers.txt**

1|RAMZES666|Свободный агент|Россия|1585956|785|73|

2|iLTW|Свободный агент|Россия|76836|1851|11|

3|Nightfall|Virtus.pro|Россия|122097|939|27|

4|Dendi|B8|Украина|381872|885|38|

5|Topson|OG|Финляндия|5513607|478|93|

6|gpk|Virtus.pro|Россия|106002|1709|17|

7|Miracle|Team Nigma|Иордания|4788991|482|82|

8|SumaiL|Свободный агент|Пакистан|3685086|1139|64|

9|Iceberg|Natus Vincere|Украина|205277|1014|23|

10|Solo|NoTechies|Россия|1629302|509|77|

11|Cooman|HellRaisers|Украина|75882|1431|12|

12|V-Tune|Natus Vincere|Украина|75171|968|17|

13|Zayac|NoTechies|Кыргыстан|189840|1112|23|

14|Kingslayer|Virtus.pro|Россия|149632|985|24|

15|lilmealone|HellRaisers|Украина|875337|1044|52|

16|No[o]ne|AS Monaco Gambit|Украина|1608497|723|74|

17|fng|Alliance|Беларусь|521078|969|42|

18|youngG|HellRaisers|Беларусь|25322|879|13|

19|9pasha|Prosti Esli|Россия|1639844|759|70|

20|Pikachu|Team Unique|Украина|28529|849|13|

21|Cristallize|NoTechies|Украина|321250|413|33|

22|fn|Свободный агент|Украина|461270|1230|50|

23|Abed|Evil Geniuses|Филиппины|453936|1279|48|

24|Arteezy|Evil Geniuses|Канада|2200706|442|78|

25|Saksa|OG|Македония|983572|970|53|

26|Nikobaby|Alliance|Болгария|280759|1101|26|

27|s4|Alliance|Швеция|10873629|1971|83|

28|Nix|Свободный агент|Россия|29150|522|24|

29|SoNNeikO|AS Monaco Gambit|Россия|339173|1137|23|

30|MagicaL|Свободный агент|Украина|232790|564|34|

31|MATUMBAMAN|Team Secret|Финляндия|3899751|803|84|

32|DyrachYO|Winstrike Team|Россия|10400|803|10|

33|w33|Team Nigma|Румыния|2186705|873|76|

34|MidOne|OG|Малайзия|1817425|823|69|

35|Fly|Evil Geniuses|Израиль|2338725|634|78|

36|sobad|Team Spirit|Беларусь|16700|1177|8|

37|Daxak|Prosti Esli|Россия|102550|743|33|

38|Immersion|AS Monaco Gambit|Россия|153605|1521|29|

39|dream'|AS Monaco Gambit|Кыргыстан|116010|1078|30|

40|Save-|Virtus.pro|Молдавия|122097|956|32|

**statistics.h**

#ifndef \_statistics\_h\_

#define \_statistics\_h\_

typedef struct{

int id;

char gamer[20];

char team[50];

char country[30];

int prize\_money;

int matches;

int proportion\_of\_vict;

} statistics;

#endif

**prog1.c**

#include <stdio.h>

typedef struct{

int id;

char gamer[20];

char team[50];

char country[30];

int prize\_money;

int matches;

int proportion\_of\_vict;

} statistics;

int main(int argc, char \* argv[]){

if(argc!=3){

fprintf(stderr, "Неверное количество аргументов\n");

return 1;

}

FILE \*in = fopen(argv[1], "r");

if(in==NULL){

fprintf(stderr, "Что-то не так с файлом '%s'\n", argv[1]);

return 2;

}

FILE \*out = fopen(argv[2],"w");

if(out==NULL){

fprintf(stderr, "Что-то не так с файлом '%s'\n", argv[2]);

fclose(in);

return 3;

}

statistics a;

int k=0;

while((k=fscanf(in, "%d|%[^|]|%[^|]|%[^|]|%d|%d|%d|", &a.id, a.gamer, a.team, a.country, &a.prize\_money, &a.matches, &a.proportion\_of\_vict)) && k==7){

fwrite(&a,sizeof(statistics),1,out);

if(ferror(out)!=0){

fprintf(stderr, "Файл для записи переполнен\n");

fclose(in);

fclose(out);

return 4;

}

}

if(k!=-1){

fprintf(stderr, "Не хватает аргументов в строчке '%d'\n", a.id);

return 5;

}

if(ferror(in)!=0){

fprintf(stderr, "Ошибка чтения\n");

fclose(in);

fclose(out);

return 5;

}

fclose(in);

fclose(out);

return 0;

}

**vector.h**

#ifndef \_vector\_tv\_h\_

#define \_vector\_tv\_h\_

#include "team\_victories.h"

typedef struct {

int size;

team\_victories \*buffer;

} vector\_tv;

void vector\_tv\_create(vector\_tv \*v);

void vector\_tv\_destroy(vector\_tv \*v);

bool vector\_tv\_resize(vector\_tv \*v, int new\_size);

int vector\_tv\_size(vector\_tv \*v);

bool vector\_tv\_push(vector\_tv \*v, team\_victories f);

#endif

**vector\_functions.c**

#include <stdio.h>

#include <string.h>

#include <stdlib.h>

#include <stdbool.h>

#include "vector.h"

void vector\_tv\_create(vector\_tv \*v){

v->size=0;

v->buffer=NULL;

}

void vector\_tv\_destroy(vector\_tv \*v){

v->size=0;

free(v->buffer);

v->buffer=NULL;

}

bool vector\_tv\_resize(vector\_tv \*v, int new\_size){

team\_victories \*n=realloc(v->buffer,sizeof(team\_victories)\*new\_size);

if(n!=NULL){

v->size=new\_size;

v->buffer=n;

return true;

} else {

return false;

}

}

int vector\_tv\_size(vector\_tv \*v){

return v->size;

}

bool vector\_tv\_push(vector\_tv \*v, team\_victories f){

if(!vector\_tv\_resize(v, (v->size)+1)){

return false;

}

strcpy(v->buffer[(v->size)-1].team,f.team);

v->buffer[(v->size)-1].victories=f.victories;

return true;

}

**team\_victories.h**

#ifndef \_team\_victories\_h\_

#define \_team\_victories\_h\_

typedef struct {

char team[30];

int victories;

} team\_victories;

#endif

**prog2.c**

#include <stdio.h>

#include <stdbool.h>

#include <stdlib.h>

#include <string.h>

#include "vector.h"

#include "statistics.h"

#include "team\_victories.h"

int main(int argc, char \*argv[]){

if(argc<3 || argc>5){

fprintf(stderr, "Неверное количество аргументов\n");

return 1;

}

int f\_read=0;

int f\_exicute=0;

int number\_of\_p=-1;

for(int i=2;i<argc;i++){

if(!strcmp(argv[i], "-f")){

f\_read=1;

} else if(!strcmp(argv[i], "-p")){

i++;

if(i<argc){

if((!atoi(argv[i]) && strcmp(argv[i],"0"))){

fprintf(stderr, "\"%s\" не является числом\n", argv[i]);

return 10;

}

} else{

fprintf(stderr, "После \"-p\" должно идти число\n");

return 30;

}

f\_exicute=1;

number\_of\_p=atoi(argv[i]);

} else{

fprintf(stderr, "\"%s\" не подходит в качестве аргумента.\n USAGE: ./<PROGRAMM NAME> <INPUT FILENAME> <KEYS{-p NUM, -f}>\n", argv[i]);

return 40;

}

}

FILE \*in=fopen(argv[1],"r");

if(in==NULL){

fprintf(stderr,"Ошибка чтения файла '%s'",argv[1]);

return 2;

}

statistics a;

if(f\_read){

while(fread(&a,sizeof(a),1,in)){

printf("%d|%s|%s|%s|%d|%d|%d|\n", a.id, a.gamer, a.team, a.country, a.prize\_money, a.matches, a.proportion\_of\_vict);

}

fseek(in,0,SEEK\_SET);

}

if(f\_exicute){

vector\_tv tmp={0, NULL};

vector\_tv \*v=&tmp;

vector\_tv\_create(v);

team\_victories f;

int status=0;

while(fread(&a,sizeof(a),1,in)){

strcpy(f.team,a.team);

f.victories=(a.matches)\*(a.proportion\_of\_vict)/100;

for(int i=0;i<vector\_tv\_size(v);i++){

if(!strcmp(a.team,v->buffer[i].team)){

status=1;

v->buffer[i].victories=v->buffer[i].victories+f.victories;

break;

}

}

if(status!=1){

if(!(vector\_tv\_push(v, f))){

fprintf(stderr, "Недостаточно места");

return 3;

}

}

status=0;

}

fseek(in,0,SEEK\_SET);

int max=0;

char most\_available\_team[50];

for(int i=0;i<vector\_tv\_size(v);i++){

if(v->buffer[i].victories>max){

max=v->buffer[i].victories;

strcpy(most\_available\_team,v->buffer[i].team);

}

}

printf("Команда с наибольшим количеством побед(одна из): %s\n", most\_available\_team);

printf("Игроки команды с количеством призовых %d или больше:\n", number\_of\_p);

while(fread(&a,sizeof(a),1,in)){

if(!strcmp(a.team,most\_available\_team) && a.prize\_money>=number\_of\_p){

printf("%s\n", a.gamer);

}

}

vector\_tv\_destroy(v);

return 0;

}

}

# 10. Демонстрация работы программы

root@DESKTOP-5HM2HTK:~# gcc -std=c99 -pedantic -Wall prog1.c

root@DESKTOP-5HM2HTK:~# ./a.out gamers.txt bin.bin

root@DESKTOP-5HM2HTK:~# gcc -std=c99 -pedantic -Wall prog2.c vector\_functions.c

root@DESKTOP-5HM2HTK:~# ./a.out bin.bin -f -p 345678

1|RAMZES666|Свободный агент|Россия|1585956|785|73|

2|iLTW|Свободный агент|Россия|76836|1851|11|

3|Nightfall|Virtus.pro|Россия|122097|939|27|

4|Dendi|B8|Украина|381872|885|38|

5|Topson|OG|Финляндия|5513607|478|93|

6|gpk|Virtus.pro|Россия|106002|1709|17|

7|Miracle|Team Nigma|Иордания|4788991|482|82|

8|SumaiL|Свободный агент|Пакистан|3685086|1139|64|

9|Iceberg|Natus Vincere|Украина|205277|1014|23|

10|Solo|NoTechies|Россия|1629302|509|77|

11|Cooman|HellRaisers|Украина|75882|1431|12|

12|V-Tune|Natus Vincere|Украина|75171|968|17|

13|Zayac|NoTechies|Кыргыстан|189840|1112|23|

14|Kingslayer|Virtus.pro|Россия|149632|985|24|

15|lilmealone|HellRaisers|Украина|875337|1044|52|

16|No[o]ne|AS Monaco Gambit|Украина|1608497|723|74|

17|fng|Alliance|Беларусь|521078|969|42|

18|youngG|HellRaisers|Беларусь|25322|879|13|

19|9pasha|Prosti Esli|Россия|1639844|759|70|

20|Pikachu|Team Unique|Украина|28529|849|13|

21|Cristallize|NoTechies|Украина|321250|413|33|

22|fn|Свободный агент|Украина|461270|1230|50|

23|Abed|Evil Geniuses|Филиппины|453936|1279|48|

24|Arteezy|Evil Geniuses|Канада|2200706|442|78|

25|Saksa|OG|Македония|983572|970|53|

26|Nikobaby|Alliance|Болгария|280759|1101|26|

27|s4|Alliance|Швеция|10873629|1971|83|

28|Nix|Свободный агент|Россия|29150|522|24|

29|SoNNeikO|AS Monaco Gambit|Россия|339173|1137|23|

30|MagicaL|Свободный агент|Украина|232790|564|34|

31|MATUMBAMAN|Team Secret|Финляндия|3899751|803|84|

32|DyrachYO|Winstrike Team|Россия|10400|803|10|

33|w33|Team Nigma|Румыния|2186705|873|76|

34|MidOne|OG|Малайзия|1817425|823|69|

35|Fly|Evil Geniuses|Израиль|2338725|634|78|

36|sobad|Team Spirit|Беларусь|16700|1177|8|

37|Daxak|Prosti Esli|Россия|102550|743|33|

38|Immersion|AS Monaco Gambit|Россия|153605|1521|29|

39|dream'|AS Monaco Gambit|Кыргыстан|116010|1078|30|

40|Save-|Virtus.pro|Молдавия|122097|956|32|

Команда с наибольшим количеством побед(одна из): Свободный агент

Игроки команды с количеством призовых 345678 или больше:

RAMZES666

SumaiL

fn

root@DESKTOP-5HM2HTK:~#

# 11. Заключение

Хранение данных на диске в виде файлов позволяет обрабатывать гораздо большие объёмы информации, так как даже большой массив не вместит маленькой простой базы данных, что уж говорить о затратах оперативной памяти в таком процессе. В этой работе была продемонстрирована работа с примитивной базой данных в виде таблицы из семи столбцов. Использование файлов оказалось схожим с использованием массивов совместно с консолью, что позволило быстро освоиться с базовыми возможностями.

# 12. Список используемых источников

1. Методические указания к выполнению курсовых работ. Зайцев В. Е.
2. Описание функции fseek: <http://all-ht.ru/inf/prog/c/func/fseek.html>
3. Описание функции fread: <http://www.c-cpp.ru/content/fread>