Московский авиационный институт

Факультет информационных технологий и прикладной математики

Кафедра вычислительной математики и программирования

КУРСОВОЙ ПРОЕКТ

по курсу “Практикум на ЭВМ”

II семестр

Тема “Обработка последовательной файловой структуры на языке Си”

Работу выполнила:

Студентка группы М8О-106Б-21

Деревянко Екатерина Андреевна

Работу проверил:

Старший преподаватель

Дубинин Алексей Владимирович

Москва 2022

**Оглавление**

[Введение 3](#_Toc15229)

[Теоретическая часть 4](#_Toc2653)

[Файловый ввод и вывод 4](#_Toc31816)

[Файловый дескриптор 7](#_Toc27093)

[Работа с аргументами программы 9](#_Toc31346)

[Практическая часть 10](#_Toc4928)

[Задание 10](#_Toc9616)

[Протокол 10](#_Toc25188)

[Результат программ 11](#_Toc29392)

[Заключение 13](#_Toc17990)

[Список источников 14](#_Toc29032)

# Введение

Часто может получится так, что входные или выходные данные для программы слишком велики, чтобы вводить их вручную и выводить на экран. Тогда на помощь приходят файлы, но их тоже надо уметь обрабатывать и понимать, как программа работает с ними. Поэтому мной были составлены две программы, первая переводит текстовый документ в бинарный, а вторая обрабатывает полученный бинарный файл.

# **Теоретическая часть**

## Файловый ввод и вывод

Для доступа к файлу в языке Си применяется тип данных FILE. Это структурный тип, имя которого задано с помощью оператора typedef в стандартном заголовочном файле "stdio.h". FILE содержит внутреннюю информацию о состоянии соединения с ассоциированным файлом, включая информацию о позиции курсора, информацию о буферизации, индикатор конца файла и ошибки. Программисту не нужно знать, как устроена структура типа файл: ее устройство может быть системно зависимым, поэтому в целях переносимости программ обращаться явно к полям структуры FILE запрещено. Особенностью языка Си, который был впервые применен при разработке ОС UNIX, является отсутствие заранее спланированных структур файлов. Все файлы рассматриваются как неструктурированные последовательности байтов.

Открытие файла выполняется с помощью функции fopen, которая создает новый поток ассоциированный с открываемым файлом. Если файл не существует, то он будет создан. Синтаксис функции:

*FILE \* fopen (const char \*filename, const char \*opentype)*

Функция fopen открывает поток для ввода/вывода в/из файла filename и возвращает указатель на поток. Аргумент opentype — это строка, управляющая режимом открытия файла. Режимы открытия: r (чтение), r+ (чтение и запись), w (запись), w+ (запись и чтение), a (добавление), a+ (добавление и чтение). В некоторых операционных системах имеются различия в работе с текстовыми и бинарными файлами (к таким системам относятся MS DOS и MS Windows; в системе Unix различий между текстовыми и бинарными файлами нет). В таких системах при открытии бинарного файла к режиму следует добавлять букву " b " (от слова binary), а при открытии текстового файла - букву " t " (от слова text). В случае неуспеха fopen возвращает NULL.

Поток характеризуется позицией чтения и(или) записи. В ОС GNU\Linux это просто счетчик байтов относительно начала файла. При дисковом вводе/выводе позиция может меняться произвольно. Некоторые другие типы файлов позволяют то же. Файлы, поддерживающие смену позиции – файлы с произвольным доступом. Функция fseek (*int fseek (FILE \*stream, long int offset, int whence)*) используется для смены позиции чтения/записи в потоке. Значение whence должно быть одной из констант SEEK\_SET, SEEK\_CUR или SEEK\_END, которые соответствуют первому, текущему или последнему байту файла. fseek устанавливает позицию чтения/записи смещаясь на offset байт от позиции, указанной параметром whence.

После того как файл открыт, можно читать информацию из файла или записывать информацию в файл. Для этого есть функции бинарного чтения и записи fread и fwrite. Они называются бинарными потому, что не выполняют никакого преобразования информации при вводе или выводе: информация хранится в файле как последовательность байтов ровно в том виде, в котором она хранится в памяти компьютера. Функции возвращают соответственно количество считанных или записанных элементов. Функция fread принимает область памяти, куда будет записана считанная информация, размер элементов, их количество и файл, с которого производится чтение. Функция fwrite принимает то же самое, но первый аргумент это область памяти, откуда считываются данные, а файл указывается, в который будет записана информация.

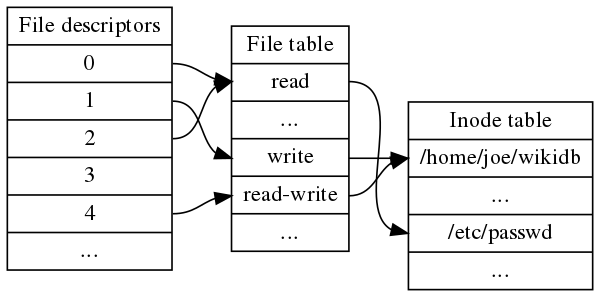
В отличие от функции бинарного ввода fread, которая вводит байты из файла без всякого преобразования непосредственно в память компьютера, функция форматного ввода fscanf предназначена для ввода информации с преобразованием ее из текстового представления в бинарное. Информация о количестве читаемых элементов, их типах и особенностях представления задается с помощью формата. В случае функции ввода формат - это строка, содержащая описания одного или нескольких вводимых элементов. Форматы, используемые функцией fscanf, аналогичны применяемым функцией scanf. Функция fscanf возвращает число успешно введенных элементов.

Функция fprintf используется для форматного вывода в файл. Данные при выводе преобразуются в их текстовое представление в соответствии с форматной строкой. В отличие от printf данная функция позволяет указывать поток, в который производится вывод. Например, можно вывести что-нибудь на стандартный поток ошибок.

## Файловый дескриптор

Файловый дескриптор — это неотрицательное число, которое является идентификатором потока ввода-вывода. Дескриптор может быть связан с файлом, каталогом, сокетом.

Обычно файловые дескрипторы выделяются последовательно. Есть пул свободных номеров. Когда вы создаете новый файл или открываете существующий, ему присваивается номер. Дескриптор для каждого процесса является уникальным. Но есть три жестко закрепленных индекса: 0 - стандартный ввод, 1 - стандартный вывод, 2 - стандартный поток ошибок. Когда завершается работа с файлом, присвоенный ему дескриптор освобождается и возвращается в пул свободных номеров. Он снова доступен для выделения под новый файл.



В традиционной реализации Unix дескрипторы индексируются в таблицу дескрипторов для каждого процесса, поддерживаемого ядром. Таблица файловых дескрипторов индексирует общесистемную таблицу файлов, открытых всеми процессами. В таблице файлов записывается режим, в котором открыт файл или другой ресурс — например, для чтения, записи, чтения и записи. Режим индексируется в таблицу индексных дескрипторов, описывающих фактические базовые файлы. В каждом индексном дескрипторе хранятся атрибуты и расположение дисковых блоков переданного объекта. Когда нужно выполнить ввод или вывод, процесс через системный вызов передает ядру дескриптор нужного файла. Ядро обращается к файлу от имени процесса. При этом у самого процесса нет доступа к файлу или таблице индексных дескрипторов.

Благодаря файловым дескрипторам не важно,  что и кто делает файлом после открытия, его могут удалить, его могут переименовать, ему могут поменять владельца или забрать права на чтение и запись, к нему все равно будет доступ.

## Работа с аргументами программы

 Аргумент командной строки — это информация, которая вводится в командной строке операционной системы вслед за именем программы, которая во время запуска может принимать параметры. Они являются аргументами функции main. Общий вид функции main следующий:

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

*...*

*}*

Первым аргументом argc является число переданных функции параметров. Второй аргумент – массив строк – собственно сами параметры. Так как параметры у функции могут быть любыми, то они передаются как строки, и уже сама программа должна их разбирать и приводить к нужному типу. Первым аргументом (argv[0]) всегда является имя программы.

Для доступа к параметру под номером i следует писать argv[i]. Выражение argv[i] — это строка, передаваемая в соответствии с соглашениями языка Си о строках. С ней можно проделывать все операции, как с обычной строкой.

Во многих средах все аргументы командной строки необходимо отделять друг от друга пробелом или табуляцией. Запятые, точки с запятой и тому подобные символы разделителями не считаются. В программах с аргументами командной строки часто делается следующее: в случае, когда пользователь запускает эти программы без ввода нужной информации, выводятся инструкции о том, как правильно указывать аргументы.

# Практическая часть

## Задание

Мной должна была быть выбрана предметная область, информацию по которой можно было бы представить в виде структуры и было бы около 50 записей. Мой выбор пал на страны Европы, потому что их можно перечислить все. Так же в структуре будут поля: столица, население страны и население столицы. Задание к бинарному файлу не должно было решаться одним проходом файла. В качестве задания я решила узнать, в каких странах Европы население страны выше среднего, но население столицы ниже. Также можно использовать необязательный аргумент -d, с которым программа выведет страны с населением выше среднего.

## Протокол

*Программа 1*

#include <stdio.h>

#include "countries.h"

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

if (argc!=3){

printf("Please choose 2 files\n");

return 1;

}

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

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

if (in==NULL){

printf("Can't open first file\n");

return 2;

}

if (out==NULL){

printf("Can't open second file\n");

return 3;

}

europe c;

while (fscanf(in, "%[^:]:%[^:]:%d:%d\n", c.country, c.capital, &c.countpop, &c.cappop)==4){

if (fwrite(&c, sizeof(c), 1, out)!=1){

printf("Error\n");

return 4;

}

}

}

*Программа 2.*

[#include](https://vk.com/im?sel=578361701&st=%23include) <stdio.h>

[#include](https://vk.com/im?sel=578361701&st=%23include) <string.h>

[#include](https://vk.com/im?sel=578361701&st=%23include) "countries.h"

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

int country\_pop=0;

int capital\_pop=0;

int n=0;

int k=0;

europe p;

float avg\_country=0;

float avg\_capital=0;

if (argc==2 && (strcmp(argv[1],"-d")!=0)){

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

if (in==NULL){

printf("Can't open file\n");

return 2;

}

while (fread(&p, sizeof(p), 1, in)==1){

country\_pop=country\_pop+p.countpop;

n=n+1;

capital\_pop=capital\_pop+p.cappop;

k=k+1;

}

fseek(in, 0, SEEK\_SET);

if (n==0){

printf("There is no countries in file\n");

return 4;

}

avg\_country=country\_pop/n;

avg\_capital=capital\_pop/k;

while (fread(&p, sizeof(p), 1, in)==1){

if ((p.countpop>avg\_country) && (p.cappop<avg\_capital)){

printf("%s\n", [p.country](https://vk.com/away.php?to=http://p.country&cc_key=));

}

}

}

else if (argc==3 && (strcmp(argv[2],"-d")==0)){

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

if (in==NULL){

printf("Can't open file\n");

return 5;

}

while (fread(&p, sizeof(p), 1, in)==1){

country\_pop=country\_pop+p.countpop;

n=n+1;

}

fseek(in, 0, SEEK\_SET);

if (n==0){

printf("There is no countries in file\n");

return 6;

}

avg\_country=country\_pop/n;

while (fread(&p, sizeof(p), 1, in)==1){

if (p.countpop>avg\_country){

printf("%s\n", [p.country](https://vk.com/away.php?to=http://p.country&cc_key=));

}

}

}

else {

printf("Please choose file\n");

}

}

## Результат программ

В результате первой программы создается бинарный файл, в который был переведен данный текстовый файл. В результате же второй программы выводится только одна страна - Нидерланды. Если использовать аргумент -d, то результатом будут:

United Kingdom

Germany

Spain

Italy

Netherlands

Poland

Russia

Romania

Ukraine

France

# Заключение

При работе с файлами в языке Си для каждого открывается отдельный поток, а работа происходит с указателями на эти потоки. Каждому файлу присваивается файловый дескриптор, благодаря которому сохраняется доступ к файлу после его открытия. В процессе написания программ я научилась работать с файлами в Си и узнала, как передавать аргументы командной строки программе.

# Список источников

1. Работа с файлами

<https://intuit.ru/studies/curriculums/4509/courses/67/lecture/1978?page=2>

1. Потоковый ввод и вывод

<https://ita.sibsutis.ru/sites/default/files/courses/pavu/FILE_IO.pdf>

1. Что такое файловый дескриптор

<https://timeweb.com/ru/community/articles/chto-takoe-faylovyy-deskriptor-prostymi-slovami>

1. Аргументы функции main

<https://cpp.com.ru/shildt_spr_po_c/06/0604.html>

1. Параметры командной строки

<https://learnc.info/c/cmd.html>