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**

**«Работа с файлами в языке Си»**

|  |  |
| --- | --- |
| **Студент:** | Ватулин Валентин |
| **Группа:** | М8О-106Б-19 |
| **Преподаватель:** | Дубинин Алексей |
| **Оценка:** |  |
| **Дата:** |  |

Москва, 2020

**Содержание**

[Введение 3](#_Toc43596291)

[Структура типа FILE 4](#_Toc43596292)

[Функции для работы с файлами 5](#_Toc43596293)

[Файловый дескриптор 7](#_Toc43596294)

[Описание программы 8](#_Toc43596295)

[Заключение 9](#_Toc43596296)

# Введение

Почти всегда данные поступают не со стандартного потока ввода, а из файла, вывод также может идти в файл вместо стандартного потока вывода. В языке Си есть основными функциями для работы с файлами считаются fopen, fscanf, fprintf, fwrite. Но, несмотря на это, функциональность ввода-вывода языка Си по текущим стандартам реализуется на низком уровне.

# Структура типа FILE

Для каждого потока, который связывается с файлом, используется специальная управляющая структура, имеющая тип FILE.

* Эта структура определяется в файле <stdio.h>.
* Ее конкретная реализация системно зависима.
* Ее поля заполняются и используются операционной системой.

Библиотечные функции для работы с файлами используют указатель на эту структуру (указатель на файл).

* Файл предварительно должен быть открыт.
* По окончании всех операций с файлом его рекомендуется закрыть.
* Освободившийся указатель можно использовать для открытия другого файла.

Количество одновременно открытых файлов системно-зависимо.

Когда поток открывается для ввода-вывода, он связывается со стандартной структурой типа FILE, которая определена в stdio.h. Структура FILE содержит необходимую информацию о файле. Объект типа FILE в памяти размещается автоматически, эта задача лежит на функциях fopen или tmpfile. Задача высвобождения памяти выполняется функцией fclose. При подключении заголовочного файла “stdio.h” в проект, автоматически создаются три объекта типа FILE\*, потоки ввода/вывода и стандартный поток ошибок. К этим потокам можно обратиться через указатели stdin, stdout и stderr.

# Функции для работы с файлами

*1) FILE \*fopen(const char \*filename, const char \*mode);*

* Открывает файл с именем filename и возвращает указатель на связанный с этим файлом поток или NULL при ошибке;
* Строка mode задаёт способ доступа к файлу и тип потока (текстовый или бинарный);

-- r открыть только для чтения

-- w открыть только для записи

-- a открыть для записи в конец файла или создать новый, если не существует

-- r+ открыть существующий файл для чтения и записи

-- w+ создать новый файл для чтения и записи

-- a+ открыть (или создать) для чтения и записи в конце файла

* для указания типа потока к способу доступа добавляется сивмол t (текстовый) или b (бинарный), например, wt или a+b
* Если t или b не указаны, то тип потока по умолчанию задаётся константой \_fmode (O\_TEXT, O\_BINARY)

*2) int fclose(FILE \*fp);*

* Закрывает указанный поток, ранее открытый с помощью fopen
* Возвращает 0 при успешном завершении и EOF при ошибке

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

* 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.
* 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().
* Если поток открыт для текстовых операций, то возврат каретки и последовательности перевода строки автоматически транслируются в символы новых строк.

*4) size\_t fwrite(const void \* ptrvoid, size\_t size, size\_t count, FILE \* filestream);*

* Функция fwrite() записывает count объектов — каждый объект по size символов в длину — в поток, указанный stream, из символьного массива, указанного buf. Указатель позиции в файле продвигается вперед на количество записанных символов.
* Функция fwrite() возвращает количество действительно записанных объектов, которое в случае успеха равно затребованному количеству. Если количество записанных объектов меньше, чем это указано при вызове, то произошла ошибка.
* Если поток открыт для текстовых операций, то при записи в файл символ новой строки авто­матически транслируется в последовательность возврат каретки — перевод строки.

*5) int fseek (FILE \*stream, long offset, int origin);*

* Аргумент offset — это выраженный в байтах сдвиг от позиции, определяемой origin, до новой позиции. Аргумент origin может принимать значения 0, 1 или 2, причем 0 означает начало файла, 1 — текущую позицию, а 2 — конец файла. В stdio.h определе­ны следующие макросы для origin: SEEK\_SET, SEEK\_CUR, SEEK\_END
* В случае успеха fseek() возвращает 0. Ненулевое значение означает неудачу. С помощью fseek() можно переместить указатель положения в любую точку внутри файла и даже за его пределы после конца файла. Однако попытка установить указатель перед началом файла будет восприня­та как ошибка.
* Функция fseek() сбрасывает флаг конца файла, связанный с указанным потоком. Кроме того, она обнуляет любую предыдущую ungetc() в том же потоке.

*6) int feof( FILE \*fp )*

* возвращает ненулевое значение, если при операции с файлом установился индикатор конца файла.

Также операции ввода-вывода можно осуществлять с помощью функций fgetc, fputc, ungetc, getw, putw, fscanf, fprintf,

Прочие функции: ferror, clearerr, fflush, flushall, ftell, rewind

# Файловый дескриптор

Файловый дескриптор – это просто число, которое ядро возвращает процессу, создавшему поток ввода-вывода. По умолчанию стандартный поток ввода определяется числом 0, стандартный поток вывода числом 1, а стандартный поток ошибок числом 2. В пределах одного процесса все дескрипторы уникальны (даже если они соответствуют одному и тому же файлу или устройству). Вся информация об открытых файлах содержится в таблице дескрипторов. Таблица дескрипторов, помимо всего прочего, содержит информацию о текущей позиции чтения-записи для каждого дескриптора. При завершении процесса все потоки самостоятельно закрываются, но если файлы не закрывать самостоятельно, продолжая работу программы, то соответствующие дескрипторы не освобождаются, что может привести к превышению лимита открытых файлов и к повышению затрат по памяти

По умолчанию Unix-оболочки связывают файловый дескриптор 0 с потоком стандартного ввода, файловый дескриптор 1 — с потоком стандартного вывода, и файловый дескриптор 2 — с потоком ошибок. Это соглашение соблюдается многими Unix-оболочками и многими приложениями — и ни в коем случае не является составной частью ядра.

# Описание программы

1) Задание: в текстовом файле дана информация об успеваемости студентов данной группы по всем предметам (фамилия, инициалы, пол, номер группы, отметки по экзаменам и зачетам). Необходимо выяснить, сколько студенток группы *p* имеют ровно одну пятерку.

2) Сценарий работы:

* Написать первую программу, которая переписывает данные из входного текстового файла в выходной в двоичном представлении (dump.c)
* Написать вторую программу, которая считывает бинарный файл и считает количество подходящих студенток (parse.c)

3) Для хранения информации используется следующая структура:

typedef struct {

char lastname[21];

char initials[3];

char sex;

int group;

unsigned int exam1, exam2, exam3, exam4;

} student;

# Заключение

Во время работы с файлами данные могут передаваться в своём внутреннем двоичном представлении или в текстовом формате, то есть в более удобочитаемом виде. Нужно уметь правильно работать с файлами, так как большая часть данных хранится в виде файлов.

В ходе подготовки проекта были изучены функции чтения и записи текстовых и бинарных файлов (fread, fprintf, fwrite, fscanf), а также другие вспомогательные функции (например, fseek, feof)

# Список литературы

1. Тип FILE и функции для работы с файлами:

<http://cppstudio.com/post/1544/>

<http://cppstudio.com>

1. Работа с C файлами:

<https://prog-cpp.ru/c-files/>

1. Файловый дескриптор:

<https://www.opennet.ru/docs/RUS/zlp/005.html>

<https://ru.wikipedia.org/wiki/Файловый_дескриптор>