

Основы алгоритмизации и программирования

Лекция 14

Файлы в языке Си

Введение

Файл – это набор данных, размещенный на внешнем носителе и рассматриваемый в процессе обработки как единое целое. В файлах размещаются данные, предназначенные для длительного хранения.

Виды файлов

```
graph TD; A[Виды файлов] --> B[Текстовые файлы]; A --> C[Бинарные (двоичные) файлы];
```

Текстовые файлы представляют собой последовательность ASCII символов и могут быть просмотрены и отредактированы с помощью любого текстового редактора. Эта последовательность символов разбивается на строки символов, при этом каждая строка заканчивается двумя кодами «перевод строки», «возврат каретки»: 13 и 10 (0xD и 0xA).

Бинарные (двоичные) файлы представляют собой последовательность данных, структура которых определяется программно. В языке Си не предусмотрены никакие заранее определенные структуры файлов. Все файлы рассматриваются компилятором как последовательность (поток байт) информации.

Введение

Для файлов определен **маркер** или **указатель** чтения-записи данных, который определяет текущую позицию доступа к файлу. С началом работы любой программы автоматически открываются стандартные потоки *stdin* и *stdout*.

В языке **Си** имеется большой набор функций для работы с файлами, большинство которых находятся в библиотеках *stdio.h* и *io.h*. При этом потоки данных, с которыми работают функции ввода-вывода данных по умолчанию, **буферизированы**. Это означает, что при открытии потока с ним автоматически связывается определенный участок ОП, который и называется **буфером**. Все операции чтения-записи ведутся через этот буфер. Его размер фиксирован специальной константой *BUFSIZ*, которая определена в файле *stdio.h* как **512** (хотя программно ее можно изменять).

Открытие файла

Каждому файлу в программе присваивается внутреннее логическое имя, используемое в дальнейшем при обращении к нему.

Логическое имя (идентификатор файла) – это указатель на файл, т.е. на область памяти, где содержится вся необходимая информация о файле.

Формат объявления указателя на файл:

```
FILE *ID_указателя_на_файл;
```

FILE – идентификатор структурного типа, описанный в стандартной библиотеке *stdio.h*

Данная структура
содержит
следующую информацию



Открытие файла



```
struct FILE {  
    short level;           – число оставшихся в буфере непрочитанных  
                           байт; обычный размер буфера – 512 байт; как  
                           только level = 0, в буфер из файла читается  
                           следующий блок данных;  
    unsigned flags;       – флаг статуса файла – чтение, запись,  
                           дополнение;  
    char fd;              – дескриптор файла, т.е. число, определяющее  
                           его номер;  
    unsigned char hold;   – переданный символ, т.е. ungetc-символ;  
    short bsize;          – размер внутреннего промежуточного буфера;  
    unsigned char         – значение указателя для доступа внутри буфера;  
    buffer;               задает начало буфера, начало строки или  
                           текущее значение указателя внутри буфера в  
                           зависимости от режима буферизации;  
    unsigned char         – текущее значение указателя для доступа внутри  
    *curp;                буфера; задает текущую позицию в буфере для  
                           обмена с программой;  
    unsigned istemp;      – флаг временного файла;  
    short token;          – флаг при работе с файлом;  
};
```

Открытие файла

Прежде чем начать работать с файлом, т.е. получить возможность чтения или записи информации в **файл**, его нужно **открыть для доступа**.

```
FILE* fopen(char * ID_файла, char  
*режим);
```

Данная функция берет внешнее представление – **физическое имя файла** на носителе (дискета, винчестер) и ставит ему в соответствие **логическое имя** (программное имя – указатель файла).

Физическое имя, т.е. **ID файла** и путь к нему задается первым параметром – строкой, например, **“g:Mas_dat.dat”** – файл с именем **Mas_dat** и расширением **dat**, находящийся на носителе, **“d:\\work\\Sved.txt”** – файл с именем **Sved** и расширением **txt**, находящийся на винчестере в каталоге **work**.

При успешном открытии функция **fopen** возвращает указатель на файл (в дальнейшем – указатель файла). При ошибке возвращается **NULL**. Данная ситуация обычно возникает, когда неверно указывается путь к открываемому файлу, например, если указать путь, запрещенный для записи.

Второй параметр – строка, в которой задается режим доступа к файлу.

Открытие файла

```
FILE* fopen(char * ID_файла, char  
*режим);
```

Второй параметр – строка, в которой задается режим доступа к файлу.

Возможные значения данного параметра:

w – файл открывается для записи (*write*); если файла с заданным именем нет, то он будет создан; если же такой файл уже существует, то перед открытием прежняя информация уничтожается;

r – файл открывается для чтения (*read*); если такого файла нет, то возникает ошибка;

a – файл открывается для добавления (*append*) новой информации в конец;

r+ (*w+*) – файл открывается для редактирования данных, т.е. возможны и запись, и чтение информации;

a+ – то же, что и для *a*, только запись можно выполнять в любое место файла (доступно и чтение файла);

t – файл открывается в текстовом режиме;

b – файл открывается в двоичном режиме;

Последние два режима используются совместно с рассмотренными выше.

Возможны следующие комбинации режимов доступа: “*w+b*”, “*wb+*”, “*rw+*”,

“*w+t*”, “*rt+*”.

Открытие файла

По умолчанию файл открывается в текстовом режиме.

Текстовый режим отличается от двоичного тем, что при открытии файла как текстового пара символов «перевод строки» и «возврат каретки» заменяется на один символ «перевод строки» для всех функций записи данных в файл, а для всех функций вывода – наоборот – символ «перевод строки» заменяется на два символа – «перевод строки» и «возврат каретки».

Пример

```
FILE *f; //объявляется указатель на файл f;  
f = fopen (" d:\\work\\Dat_sp.dat ", "w"); //открывается для записи файл с  
логическим именем f, имеющий физическое имя Dat_sp.dat и находящийся  
на диске d в каталоге work, или более кратко:  
FILE *f = fopen ("d:\\work\\Dat_sp.dat", "w");
```


Заккрытие файла

После работы с файлом доступ к нему необходимо **заккрыть** с помощью функции



```
int fclose (указатель  
файла);
```

Для закрытия нескольких файлов введена функция



```
void fcloseall (void);
```

Если требуется изменить режим доступа к открытому в настоящий момент файлу, то его необходимо сначала закрыть, а затем вновь открыть с другими правами доступа. Для этого используется функция



```
FILE* freopen (char *ID_файла, char *режим, FILE  
*указатель_файла);
```

Данная функция сначала закрывает файл, заданный в третьем параметре (**указатель файла**), как это выполняет функция *fclose*, а затем выполняет действия, аналогичные функции *fopen*, используя указанные первый и второй параметры (открывает файл с *ID_файла* и правами доступа *режим*).

Временные файлы

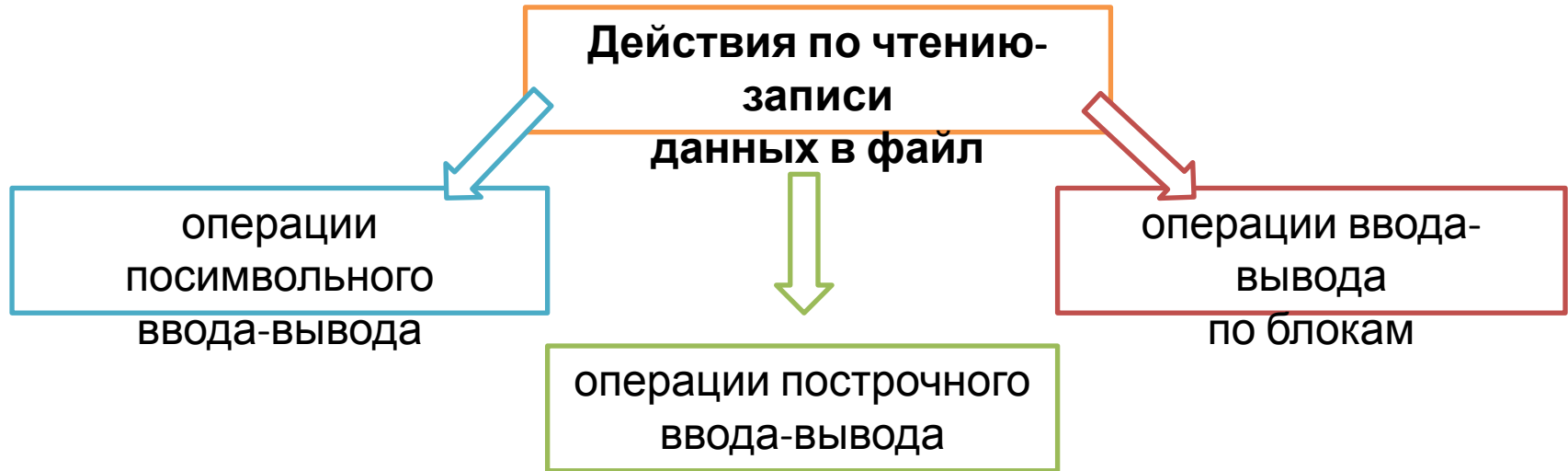
В языке **Си** имеется возможность работы с **временными файлами**, которые нужны только в процессе работы программы и должны быть удалены после выполнения некоторых вычислений. В этом случае используется функция



```
FILE* tmpfile (void);
```

Данная функция создает на диске временный файл с правами доступа **w+b**. После завершения работы программы или закрытия этого (временного) файла он автоматически удаляется

Запись-чтение информации



Для работы с текстовыми файлами в библиотеке языка **Си** содержится достаточно много функций, самыми распространенными из которых являются функции

fprintf

fscanf

fputs

fgets

Формат параметров этих функций практически такой же, как и формат рассмотренных ранее функций *printf*, *scanf*, *gets* и *puts*

Отличие состоит в том, что *printf* и другие функции работают по умолчанию с **экраном монитора** и **клавиатурой**, а функции *fprintf* и другие – с **файлом**, указатель которого является одним из параметров этих функций.

Запись-чтение информации

Пример

```
#include<stdio.h>
void main(void)
{
FILE *f1;
int a=2, b=3;
if( ! (f1 = fopen("d:\\work\\f_rez.txt","w+t") ) ) {           // f1 = NULL
puts("Open File Error!");
return;                // exit(1);
}
fprintf(f1,"\\t Файл результатов \\n");
fprintf(f1," %d плюс %d = %d\\n", a, b, a+b);
fclose(f1);
}
```

Просмотрев содержимое файла любым текстовым редактором, можно убедиться, что данные в нем располагаются точно так, как на экране, если воспользоваться функцией **printf** с такими же списками параметров.

Запись-чтение информации

Создание текстовых результирующих файлов обычно необходимо для оформления отчетов, различных документов, а также других текстовых материалов.

Бинарные (двоичные) файлы обычно используются для организации баз данных, состоящих, как правило, из объектов структурного типа. При чтении-записи бинарных файлов удобнее всего пользоваться функциями *fread* и *fwrite*, которые выполняют ввод-вывод данных блоками. Такой способ обмена данными требует меньше времени.

unsigned fread (void **p*, unsigned *size*, unsigned *n*, FILE **f*);

Данная функция выполняет считывание из файла *f* *n* блоков размером *size* байт каждый в область памяти, адрес которой *p*. В случае успеха функция возвращает количество считанных блоков. При возникновении ошибки или по достижении признака окончания файла – значение *EOF* (*End Of File* – признак окончания файла).

Обратное действие выполняет функция, при вызове которой в файл *f* будет записано *n* блоков размером *size* байт каждый из области памяти, начиная с адреса *p*

unsigned fwrite (void **p*, unsigned *size*, unsigned *n*, FILE **f*);

Позиционирование в файле

Каждый открытый файл имеет **скрытый указатель** на текущую позицию в нем. При открытии файла этот указатель устанавливается на начало данных, и все операции в файле будут производиться с данными, начинающимися в этой позиции

При каждом выполнении функции чтения или записи указатель смещается на количество прочитанных или записанных байт, т.е. устанавливается после прочитанного или записанного блока данных в файле – это **последовательный доступ к данным**

В языке **Си/C++** можно установить указатель на некоторую заданную позицию в файле. Для этого используют стандартную функцию ***fseek***, которая позволяет выполнить чтение или запись данных в произвольном порядке

```
int fseek(FILE *f, long size, int code);
```

Значение параметра ***size*** задает количество байт, на которое необходимо сместить указатель в файле ***f***, в направлении параметра ***code***, который может принимать следующие значения:

– смещение от начала файла –	0	(SEEK_SET);
– смещение от текущей позиции	1	(SEEK_CUR);
– смещение от конца файла –	2	(SEEK_END).

Позиционирование в файле

Таким образом, смещение может быть как **положительным**, так и **отрицательным**, но нельзя выходить за пределы файла.
В случае успеха функция возвращает **нулевое значение**, а в случае ошибки (например, попытка выхода за пределы файла) – **единицу**.
Доступ к файлу с использованием функции позиционирования (*fseek*) называют **произвольным доступом**.

Иногда нужно определить текущее положение в файле. Для этого используют функцию со следующей

декларацией

```
long ftell(FILE *f);
```

Функция возвращает значение указателя на текущую позицию в файле или **-1** в случае ошибки

Дополнительные файловые функции

`int fileno (FILE *f)`

определяет и возвращает значение дескриптора (*fd*) файла *f*, т.е. число, определяющее номер файла

`long filelength (int fd)`

возвращает длину файла, имеющего дескриптор *fd*, в байтах

`int chsize (int fd, long pos)`

выполняет изменение размера файла, имеющего номер *fd*, признак конца файла устанавливается после байта с номером *pos*

`int feof (FILE *f)`

возвращает ненулевое значение при правильной записи признака конца файла

`int fgetpos (FILE *f, long *pos)`

определяет значение текущей позиции *pos* файла *f*