



# Работа с файлами

Лекция 13

Иллюстративный материал к лекциям по алгоритмизации и программированию

Автор Саблина Н.Г.

2016 г.









### Содержание

Файлы и работа с ними

Ввод-вывод файла

Функции для работы с файлами

Задания на лабораторную работу

Контрольные вопросы

Итоги

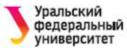
Определение некоторых понятий

Библиографический список

Автор





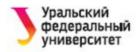




## Файлы и работа с ними

- **Файл** место на диске со своим именем, предназначенное для хранения информации.
- Для работы с файлами в языке Си создана специальная структура *FILE*.
- Структура *FILE* описана в *stdio.h*.







# Работа с файлами

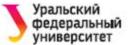
Для работы с файлами в программе нужно:

- описать указатель на объект типа *FILE*, например *FILE* \**in*;
- установить связи между указателем и конкретным файлом на диске (открыть файл) с помощью функции *fopen():*

in = fopen("имя файла","режим");

- Читать /записать в файл
- Закрыть файл при помощи функции fclose(), аргумент функции указатель на файл, а не имя файла









### **Функция** *fopen* (1 из 3)

- Возвращает указатель на структуру FILE
- Параметры
  - Имя файла
  - Режим описывает, как должен использоваться файл. Основные режимы работы с файлами:

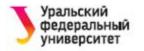
```
"r" – файл можно считать,
```

"w" - файл нужно записать,

"a" - файл можно дополнить.

- Коды являются строками, они заключаются в двойные кавычки.
- Если используется "w" для существующего файла, то старая версия файла стирается, а программа начинает заносить информацию во вновь созданный «чистый» файл с тем же именем.







## Режимы открытия файлов

- "r" Открыть для чтения
- "w" Создать для записи
- "а" Открыть для добавления в существующий файл
- "rb" Открыть двоичный файл для чтения
- "wb" Открыть двоичный файл для записи
- "ab" Открыть двоичный файл для добавления
- "r+" Открыть файл для чтения и записи
- "w+" Создать файл для чтения и записи
- "a+" Открыть для добавления или создать для чтения и записи
- "a+t" Открыть текстовый файл для добавления или создать для чтения и записи

- "r+b" Открыть двоичный файл для чтения и записи
- "w+b" Создать двоичный файл для чтения и записи
- "a+b" Открыть двоичный файл для добавления или создать для чтения и записи
- "rt" Открыть текстовый файл для чтения
- "wt" Создать текстовый файл для записи
- "at" Открыть текстовый файл для добавления
- "r+t" Открыть текстовый файл для чтения и записи
- "w+t" Создать текстовый файл для чтения и записи





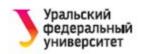


### Функция fopen (2 из 3)

- Если *fopen()* не может открыть требуемый файл, она возвращает значение *NULL*.
- Рекомендуется производить обработку возможных ошибок, например:





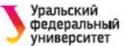




# Ввод-вывод данных из файла

- Для ввода информации в текстовой файл и чтения ее из файла можно использовать функции fprintf() и fscanf().
- Эти функции работают аналогично функциям *printf()* и *scanf()*, но имеют дополнительный аргумент для ссылки на сам файл.
- Указатель на файл первый в списке аргументов





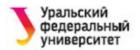




# Пример 1

```
#include <stdio.h>
main()
{ FILE fi;
int age;
fi = fopen("sam","r");
fscanf(fi,"%d",&age);
fclose(fi);
fi = fopen("data","a");
fprintf(fi,"Число %d из файла sam",age);
fclose(fi); }
```





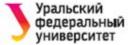


# Функции fgets() и fputs()

```
Для работы с текстовыми файлами удобно использовать
  функции fgets() и fputs(). Их описание имеет вид:
      fgets(char *s, int n, FILE *stream);
      fputs(char *s, FILE *stream);
Здесь:
s – символьный массив (строка),
n – максимальная длина считываемой строки,
stream – указатель на объект типа FILE.
```









### Функция fputs()

- Функция fputs() записывает, ограниченную символом '\0' строку (на которую указывает s) в файл, определённый указателем *stream*.
- Символ 10' в файл не переноситься, и символ 1n' не записывается в конце строки вместо '\0'.







### Функция fgets() (1 из 2)

- Функция fgets() читает из определённого указателем stream файла не более (n-1) символов и записывает их в строку, на которую указывает s.
- Функция прекращает чтение, как только прочитает (*n*-1) символов или встретит символ новой строки '\*n*', который переноситься в строку *s*.







### Функция fgets() (2 из 2)

- Дополнительно в конец каждой строки записывается признак окончания строки '\0'. В случае успешного завершения функция возвращает указатель s.
- При ошибке или при достижении конца файла, при условии, что из файла не прочитан ни один символ, возвращается значение *NULL*. В этом случае содержимое массива, который адресуется указателем *s*, остаётся без изменений.







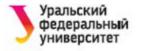
## Функции fread() и fwrite()

Если в файле отсутствует разбиение информации на строки, то есть файл носит ярко выраженный битовый (двоичный) характер, для операций ввода-вывода информации целесообразно пользоваться функциями *fread()* и *fwrite()*.

Описание этих функций имеет вид:

- fread(void \*buf, int size, int n, FILE \*stream);
- fwrite(void \*buf, int size, int n, FILE \*stream);









#### Здесь

- buf массив для чтения/записи информации,
- *size* размер считываемого блока в байтах,
- *n* количество блоков по *size* байт, считываемых (записываемых) за один раз,
- stream указатель на файл.

Таким образом, за один раз из файла считывается (или в файл записывается) *size\*n* байт информации.







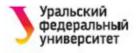


# Положение указателя в файле

Начальная позиция чтения/записи в файле устанавливается при открытии файла и может соответствовать начальному или конечному положению.

Существует возможность «быстрой» смены текущей позиции (без ввода-вывода информации). Для этого существует функция fseek(FILE \*stream, long n, int whence);







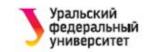
#### Здесь

- *stream* указатель на файл, в котором осуществляется перемещение;
- n целое число, которое указывает на сколько байт необходимо изменить текущую позицию в файле относительно точки, на которую указывает третий параметр функции;
- whence точка отсчета для изменения текущей позиции в файле.

Величина *whence* может принимать только одно из трех значений:

**0** (SEEK\_SET) – отчёт будет производиться от начала файла,







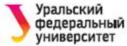
### Пример 2

В качестве примера приведём программу, которая считывает из файла f1 3 блока по 5 байт и записывает эти блоки в файл f2 в обратном порядке:

```
#include<stdio.h>
void main()
{ FILE *in, *out;
   char A[5];
   in=fopen("f1","r"); out=fopen("f2","w");
 fseek(in,10,SEEK SET);
      fread(A,5,1,in); fwrite(A,5,1,out);
 fseek(in,-10,SEEK_CUR);
     fread(A,5,1,in); fwrite(A,5,1,out);
fseek(in,-10,SEEK_CUR);
    fread(A,5,1,in); fwrite(A,5,1,out);
fclose(in); fclose(out); }
```









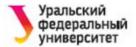
### **Функция** *feof()* (1 из 2)

Часто, при чтении информации из файла, необходимо знать, достигнут конец файла или нет.

Это можно сделать, используя функцию feof() обращение к которой выглядит следующим образом:

int feof (FILE \*stream);







### Функция feof() (2 из 2)

- Если при чтении из указанного файла достигнут его конец, то возвращается значение *NULL*, в противном случае возвращается ненулевое значение.
- Если не предпринималась попытка прочитать из файла
   отсутствующий символ, следующий за последним, то функция
   *feof*() не будет сигнализировать о том, что достигнут конец файла.

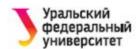




## Функции для работы с файлами

- fopen() Открыть файл
- fclose() Закрыть файл
- putc() Записать символ в поток
- getc() Прочитать символ из потока
- fseek() Переместить указатель позиции файла на указанное место
- fprintf() Форматная запись в файл
- fscanf() Форматное чтение из файла
- feof() Возвращает значение "истинно", если достигнут конец файла
- ferror() Возвращает значение "ложно", если обнаружена ошибка
- fread() Читает блок данных из потока

23.04.2016



### Задания для



#### самостоятельного решения

#### Вариант 1.

Напечатать текст из файла, подчёркивая (ставя минусы в соответствующих позициях следующей строки) все входящие в него заглавные буквы.

#### Вариант 2.

Скопировать содержимое одного текстового файла в другой, исключая пустые строки.

#### Вариант 3.

Напечатать последнюю из самых коротких строк текстового файла.

#### Вариант 4.

Разработать программу, которая построчно печатает содержимое текстового файла, вставляя в начало каждой печатаемой строки её порядковый номер (он должен занимать 4 позиции) и пробел.

#### Вариант 5.

Разработать программу, подсчитывающую количество символов, слов и строк в текстовом файле.







#### Вариант 6.

Скопировать строки из одного текстового файла в другой, расположив их в нем в порядке уменьшения длинны.

#### Вариант 7.

Разработать программу, копирующую из одного текстового файла в другой строки, оканчивающиеся восклицательным знаком.

#### Вариант 8.

Разработать программу, которая ставит в начало каждой стоки текстового файла столько пробелов, сколько в ней встречается их.

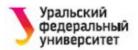
#### Вариант 9.

Написать программу, которая копирует текст из файла, в другой файл, заменяя все строчные буквы на прописные.





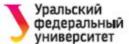
- 1. Что такое файл?
- 2. Как описать указатель на объект типа *FILE*?
- 3. Как установить связь между указателем и конкретным файлом на диске?
- 4. Какое значение может принимать второй параметр функции *fopen()*?
- 5. Как отследить ошибку в программе, которая может возникнуть при открытии несуществующего файла на чтение?
- 6. Каково предназначение параметров функций *fread()* и *fwrite()*?
- 7. Для чего используется функция *fseek()*?
- 8. Как отследить конец файла при чтении информации из него?
- 9. Для чего необходима функция *fclose()*?
- 10. В каких случаях при вводе строки следует отдавать предпочтение функции *gets()* перед функцией *scanf(*

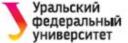


# Определение некоторых понятий



- **Включаемый файл** текстовый файл, являющийся частью транслируемого модуля, поименованный в директиве #include исходного файла C++ или в другом включаемом файле.
- **Входной поток** поток, из которого можно производить чтение.
- Выходной поток- поток, в который можно производить запись.
- Исполняемый файл- файл. Который операционная система может выполнять без дальнейшей трансляции или интерпретации.





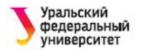


# Рассмотренные вопросы:

Итоги

- Файлы
- Функции для работы с файлами
- Ввод-вывод файла

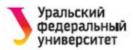




### Библиографический список

- Подбельский В.В. Язык СИ++. Учебное пособие. М.: Финансы и статистика, 2003. 560 с.
- Павловская Т.А. С/С++. Программирование на языке высокого уровня: учебник для студентов вузов, обучающихся по направлению "Информатика и вычисл. техника" СПб.: Питер, 2005.
   461 с.
- Березин Б.И. Начальный курс С и С++ / Б.И. Березин, С.Б. Березин. М.: ДИАЛОГ-МИФИ, 2001. 288 с
- Каширин И.Ю., Новичков В.С. От С к С++. Учебное пособие для вузов. М.: Горячая линия Телеком, 2005. 334 с.









Автор:

Саблина Наталья Григорьевна

Ст. преподаватель

каф. РТС УрФУ

