Лекция 2. Строковые файлы

Функция **feof()** (библиотека stdio.h)
Прототип **int feof (FILE * filestream);**

Параметры: filestream - указатель на объект типа FILE.
Возвращаемое значение: если достигнут конец файла, функция возвращает ненулевое значение, иначе возвращается нулевое значение.

Функция **feof** проверяет, достигнут ли конец файла, связанного с потоком, через параметр **filestream**. Возвращается значение, отличное от нуля, если конец файла был достигнут. Вызов данной функции выполняется после выполнения предыдущей операции с потоком, например, операции считывания, которая постепенно двигает внутренний указатель файла в конец. Дальнейшие операции с файлом, после достижения конца не будут выполняться до тех пор, пока внутренний указатель не будет сдвинут назад функциями **fseek** или **fsetpos**.

Прототип fopen:

FILE* fopen(const char * fname, const char * mode);

Если файл был успешно открыт, функция возвращает указатель на объект файла, иначе, возвращается нулевой указатель. Функция **fopen** открывает файл, имя которого указано в параметре **fname** и связывает его с потоком, который может быть идентифицирован для выполнения различных операций с файлом. Операции с потоком, выполнение которых разрешено определяются параметром **mode**.

```
1  FILE *fp;
2  char c;
3  fp = fopen("my_text.txt", "rt");
4  while( !feof(fp) )    // Пока не достигнут конец файла
5       {
6         c = getc(fp);
7         putchar(c);
8       }
   . . .
```

EOF

stdio.h C

End-of-File — конец файла

EOF определён как макро-константа. Макрос EOF заменяется целым отрицательным постоянным значением, которое зависит от системы, обычно значение равно -1.

Основное предназначение EOF — это указание конца файла. Например, при считывании данных из файла, программа может определить конец файла, прочитав это значение. В случае возникновения ошибки, работа с файлом прекращается и возвращается значение конца файла.

int putc(int character, FILE * filestream);

Функция **putc** помещает символ character в поток **filestream** и смещает позицию указателя текущего положения потока. Символ записывается в текущую позицию в потоке, на которую указывает внутренний указатель положения, после чего указатель положения смещается на один символ.

Параметры: character - символ, который должен быть помещён в поток, **filestream** - указатель на объект типа **FILE**, который идентифицирует поток, куда должен быть записан символ.

Возвращаемое значение: если ошибок не произошло при вызове функции, то возвращается целочисленное значение символа, который помещался в поток. При возникновении ошибки, возвращается EOF, а индикатор ошибки устанавливается.

int getc (FILE * filestream);

Функция возвращает символ из потока **filestream**, на который ссылается внутренний индикатор позиции файла. Внутренний индикатор после чтения сдвигается на один символ, т.о. теперь он будет указывать на следующий символ.

Параметры: filestream - указатель на объект типа **FILE**, который идентифицирует поток, из которого происходит чтение символа.

Возвращаемое значение: символ возвращается в виде целого значения (числовой код). Если конец файла достигнут, функция возвращает **EOF**. Если в процессе чтения происходит ошибка, то возвращается **EOF**, а переменной **errno** присваивается код ошибки.

int fseek(FILE * filestream, long int offset, int origin);

Функция **fseek** перемещает индикатор позиции в потоке. Устанавливает внутренний индикатор положения файла в новую позицию, которая определяются путем добавления смещения к исходному положению. Внутренний индикатор конца файла EOF очищается после вызова этой функции. Для потоков, открытых в режиме обновления (чтения + записи), вызов функции **fsetpos** позволяет переключаться между режимами чтения и записи.

Параметры: filestream - указатель на поток **FILE**, **offset** - количество байт для смещения относительно внутреннего индикатора файла, **origin** - точка отсчета смещения.

Точка отсчета может принимать значения:

SEEK_SET Начало файла
SEEK_CUR Текущее положение файла
SEEK_END Конец файла

Возвращаемое значение

В случае успеха функция возвращает нулевое значение, иначе возвращает ненулевое значение.

int fsetpos(FILE* filestream, const fpos_t * pos);
Функция **fsetpos** перемещает внутренний индикатор положения файла на новую позицию. Параметр **pos** является указателем на объект типа **fpos_t**, значение которого предварительно должно быть получено с помощью вызова функции **fgetpos**. Для потоков, открытых в режиме обновления (чтения + записи), вызов функции **fsetpos** позволяет переключаться между режимами чтения и записи.
Параметры: filestream - указатель на файл **FILE**, **position** - указатель на объект типа fpos_t, содержащий позицию указателя, полученную с помощью функции **fgetpos**.
Возвращаемое значение
В случае успеха функция возвращает нулевое значение, иначе возвращает ненулевое значение и инициализирует глобальную переменную errno положительным значение, которое может быть интерпретировано функцией **perror.**

int fgetpos(FILE * filestream, fpos_t * position);

Функция **fgetpos** возвращает текущую позицию в файле. Через параметры **filestream** и **position** функция получает информацию, необходимую для идентификации текущего положения внутреннего указателя потока. Параметр **position** должен указывать на уже существующий объект типа **fpos_t**, который предназначен, в будущем, только для использования, в качестве параметра функции **fsetpos**. Чтобы получить целочисленное значение типа **int** внутреннего индикатора позиции файла, используйте функцию **ftell**.

Параметры: filestream - указатель файл FILE, **position** - указатель на объект типа fpos_t.

Возвращаемое значение: функция возвращает нулевое значение в случае успешного выполнения, иначе - ненулевое значение.

```
fpos_t   pos; // переменная для хранения текущей позиции
...
fgetpos (ptrFile,&pos);  // получить текущую позицию в файле
....
fsetpos (ptrFile,&pos);     //установить новую позицию в файле
…
```

Построчный ввод/вывод

char* fgets(char * str, int num, FILE * filestream);

Функция **fgets** считывает символы из потока и сохраняет их в виде строки в параметр **str** до тех пор, пока не будет достигнут конец строки или конец файла. Символ новой строки **‘\n’** прекращает работу функции **fgets**, но он считается допустимым символом, и поэтому он копируется в строку **str**. Символ конца строки **‘\0’** добавляется в строку после прочитанных символов.

Параметры: **str** - указатель на массив типа **char**, в который сохраняются считанные символы, **num** - максимальное количество символов для чтения, включая нулевой символ, **filestream** - указатель на объект типа **FILE**, который идентифицирует поток, из которого считываются символы (**stdin** может быть использован для чтения из стандартного ввода.

Возвращаемое значение: в случае успеха функция возвращает указатель на **str**. Если конец файла был достигнут и ни один символ не был прочитан, содержимое **str** не меняется и возвращается нулевой указатель **NULL**. При ошибке возвращается нулевой **NULL**.

После операции чтения из файла файловый индикатор смещается на количество прочитанных символов.

int fputs(const char * string, FILE * filestream);

Функция fputs записывает строку, указанную в параметре string в поток filestream. Функция начинает копирование с адреса, указанного в string, пока не будет достигнут нулевой символ . Этот нулевой символ не копируется в поток.

Параметры: string - массив, содержащий нуль-терминированную последовательность символов для записи в поток, **filestream** - указатель на объект типа **FILE**, куда будет записана строка.

Возвращаемое значение: в случае успеха возвращается неотрицательное значение, в случае ошибки - EOF.

```
1       char buf[81];
2       FILE *fp;
3       fp = fopen("mytext.txt", “rt");
4       while(!feof(fp))
5        {
6          fgets(buf, 80, fp); // Чтение строки из файла
7          printf(“%s”, buf);
8        }
```

Создать текстовый файл, представленный в виде строк

Точка – окончание ввода

Мама

мыла

раму

.

Создать текстовый файл, представленный в виде строк

```
1char buf[101];
2FILE *fp;
3fp = fopen("mytext.txt", "wt");
4scanf(”%s”, buf);     // Ввели строку 1-ю строку
5while( buf[0] != ‘ . ’ )   // Точка – окончание ввода
6{
7  fputs(buf, fp);   // Запись строки в файл
8  scanf(“%s”, buf);
9}
```

Результат

Мамамыларамy

Создать текстовый файл, представленный в виде строк

```
1  char buf[101];
2  FILE *fp;
3  fp = fopen("mytext.txt", "wt");
4  scanf(”%s”, buf);      // Ввели строку 1-ю строку
5  while(buf[0] != ‘ . ’)  // Точка – окончание ввода
6  {
7    fputs(buf, fp); // Запись строки в файл
8    putc(‘\n’, fp); // Дописываем в файл символ \n
9    scanf(“%s”, buf);
10 }
```

Результат

Мама
мыла
раму

Строки, строковые функции и указатели

Имеется строковый файл, в котором встречаются точки. Выполнить разрезание строки по позиции точки. Точка остается в первой строке. Результат записать в другой файл.

const char * strchr(const char * string, int symbol);
char * strchr(char * string, int symbol);

Функция **strchr** выполняет поиск первого вхождения символа **symbol** в строку **string**. Возвращает указатель на первое вхождение символа в строке. Завершающий нулевой символ ‘\0’ считается частью строки, он также может быть найден для получения указателя на конец строки.

char * strchr(const char * string, int symbol);

Параметры: string - указатель на строку, **symbol** - искомый символ, передается в функцию как целое число. После того как соответствующее значение будет найдено, функция преобразует его в символ.

Возвращаемое значение
Указатель на первое вхождение символа в строку. Если значение не найдено, функция возвращает нулевой указатель **NULL**.

char * strcpy(char * buf, const char * str);

Функция копирует строку **str**, включая завершающий нулевой символ в строку, на которую ссылается указатель **buf**. Чтобы избежать переполнения, строка для копирования **buf** должна быть достаточно длинной, чтобы в неё поместилась копируемая строка (включая завершающий нулевой символ). Копируемая строка и строка назначения не должны перекрываться в памяти.

Параметры: buf - указатель на строку назначения, куда будет скопирована строка-источник, **str** - указатель на копируемую строку.

Возвращаемое значение: указатель на строку назначения **buf**.

```
1    char *a, buf1[51] ,  buf2[51];
2    FILE *f, *g;
3    f = fopen("mystr.txt", “rt");
4    fgets(buf1, 50, f);
5    a = strchr(buf1,  ‘ . ’);
6    a++;
7    strcpy(buf2, a);
8    *a=‘\0’;
9    g  = fopen("mystr2.txt", “wt");
10   fputs(buf1, g);   putc(‘\n’, g);
11   fputs(buf2, g);
```

xxxxxxxxxx.zzzzzzzzzzz.

Другие возможности при открытии файла

- rt - открыть текстовой файл для чтения;
- wt - создать текстовый файл для записи;
- at - дополнить текстовый файл;

- rt+ - открыть текстовой файл для чтения и записи;
- wt+ - создать текстовый файл для чтения и записи;
- at+ - дополнить текстовый файл с предоставлением возможности записи и чтения.

Пример 1

```
1 FILE *f1, *f2;
2 f1 = fopen("text1.txt", "rt+");
3 f2 = fopen("text2.txt", "rt");
4 while (fgets(buf, 80, f1) != NULL ); // указатель
                                       // установлен в конце файла
5 while(fgets(buf, 80, f2) != NULL)
6 {
7    fputs(buf, f1);   // Запись строки в файл
8 }
```

Пример 2

```
1 FILE *f1, *f2;
2 f1 = fopen("text1.txt", "at");
3 f2 = fopen("text2.txt", "rt");
4 while(fgets(buf, 80, f2) != NULL)
5 {
6    fputs(buf, f1);   // Запись строки в файл
7 }
```

Создать файл из 10 целых чисел

```
1 FILE *f;
2 f = fopen("numbers.num", "wb");
3 int i, n;
4 for(i=0; i<10; i++)
5 {
6    scanf("%d", &n);
7    fpintf(f, "%6d", n);
8 }
```

Какова максимальная разрядность вводимых чисел?

Выполнить чтение целых чисел из файла и вывести на экран

```
1. FILE *f;
2. f = fopen("numbers.num", "rb");
3. int n;
4. while(!feof(fp))
5. {
6.     fscanf(f, "%d", &n);
7.     printf("%d", n);
8. }
```

Выполнить чтение целых чисел из файла и вывести на экран

```
1. FILE *f;
2. f = fopen("numbers.num", "rb");
3. int n;
4. while(!feof(fp))
5. {
6.     fscanf(f, "%d", &n);
7.     printf("%d", n);
8. }
```

Предотвращение зацикливания

```
1    int k=0;
2   . . .
3    while( условие )
4      {   // тело цикла
5          k++;
6          if( k>1000)   {
7              write(“k==%d\n”, k);
8              break;    }
9    }
```