Функции ввода-вывода в C

А. Г. Фенстер, http://info.fenster.name

27 февраля 2010 г.

Функции вывода

`int putchar(int c);`
Действие: выводит один символ `c` в поток стандартного вывода `stdout` (т.е. «на экран», если вывод не перенаправлен).
Возвращает код напечатанного символа.
В случае ошибки возвращает константу `EOF`.

`int fputc(int c, FILE *f);`
Действие: выводит один символ `c` в файл `f`.
Возвращает код напечатанного символа. При `f == stdout` эквивалентна функции `putchar`.
В случае ошибки возвращает константу `EOF`.

`int puts(const char *s);`
Действие: выводит строку `s` до завершающего символа с кодом 0 в поток стандартного вывода. В конце выводится перевод строки.
Возвращает неотрицательное число.
В случае ошибки возвращает константу `EOF`.

`int fputs(const char *s, FILE *f);`
Действие: выводит строку `s` до завершающего символа с кодом 0 в файл `f`. При `f == stdout` эквивалентна функции `puts`, но перевод строки в конце не выводит.
Возвращает неотрицательное число.
В случае ошибки возвращает константу `EOF`.

`int printf(const char *fmt, ...);`
Действие: выводит текст, формат которого описан в строке `fmt`, в поток стандартного вывода. `fmt` — это строка, содержащая любые символы, а также 0 или более мест для подстановки параметра, описание которых начинается с символа `%`. Вместо каждого из них будет подставлен очередной параметр функции `printf`. Наиболее часто используемыми являются:
`%d` — целое число со знаком (decimal)
`%i` — то же самое (integer)
`%u` — целое число без знака (unsigned)
`%f` — вещественное число (floating point)
`%c` — символ (char)
`%s` — строка (string), т. е. значение типа `char *`
Подробное описание формата: `man 3 printf` (выход из справки — `q`).
Возвращает количество напечатанных символов.
В случае ошибки возвращает отрицательное число.

`int fprintf(FILE *f, const char *fmt, ...);`
Действие: аналогично функции `printf`, выводит текст, формат которого описан в строке `fmt`, но вывод производится в файл `f`.
Возвращает количество выведенных символов.
В случае ошибки возвращает отрицательное число.

`int sprintf(char *str, const char *fmt, ...);`
Действие: аналогично функции `printf`, но вывод производится в строку `str`. Эта строка завершается символом с кодом 0. Необходимо контролировать, что длины строки хватает для вывода всей информации.
Возвращает количество выведенных символов, не считая завершающего символа с кодом 0.
В случае ошибки возвращает отрицательное число.

`int snprintf(char *str, size_t size, const char *fmt, ...);`
Действие: аналогично функции `sprintf`, но длина результирующей строки, включая завершающий 0, ограничена `size` символами.
Возвращает количество символов, которые должны быть выведены (независимо от значения `size`).
В случае ошибки возвращает отрицательное число.

Функции ввода

`int getchar(void);`
Действие: читает один символ из потока стандартного вывода `stdin` (т.е. «с клавиатуры», если ввод не перенаправлен).
Возвращает код прочитанного символа.
В случае ошибки возвращает константу `EOF`.
Обратите внимание, что из-за необходимости сигнализировать об ошибке функция возвращает значение типа `int`, а не `char`.
По умолчанию функция `getchar` будет ждать нажатия Enter, т. е. её нельзя использовать для обработки «Press any key to continue». Вообще говоря, эта задача переносимым (работающим в любой операционной системе) способом не решается.

`int fgetc(FILE *f);`
Действие: читает один символ из файла f. При `f == stdin` эквивалентна функции `getchar`.
Возвращает код прочитанного символа.
В случае ошибки возвращает константу `EOF`.

`char *fgets(char *s, int size, FILE *f);`
Действие: читает из файла `f` строку размером не более чем `size` символов, включая символ с кодом 0 в конце, и помещает её по адресу `s`. Чтение завершается в одном из трёх случаев:

1. прочитано `(size-1)` символов, которые были помещены в строку `s` от 0-го до `(size-2)`-го символов включительно; в `s[size-1]` будет записан 0;
2. (наиболее частый случай) прочитан символ перевода строки, при этом сам он также помещается в `s`, и `s` дополняется символом с кодом 0;
3. файл `f` закончен; `s` дополняется символом с кодом 0.

Таким образом, строка `s` всегда оказывается завершённой, и последним символом её является символ перевода строки, кроме случаев, когда чтение завершилось по пунктам 1 или 3.
Возвращает `s`.
В случае ошибки возвращает `NULL`.

`int scanf(const char *fmt, ...);`
Действие: читает из потока стандартного ввода данные в соответствии с форматом, заданным в строке `fmt`. Второй и последующие параметры должны хранить адреса ячеек памяти, в которые нужно поместить прочитанную информацию.
Возвращает количество успешно прочитанных значений.
В случае ошибки возвращает константу `EOF`.
Обратите внимание, что если достигнут конец файла, функция `scanf` может в разных случаях вернуть как `EOF`, так и неотрицательное число. Если необходимо проверить, что все запрошенные параметры прочитаны, нужно сравнивать значение, которое вернула функция `scanf`, с количеством запрашиваемых параметров.

`int fscanf(FILE *f, const char *fmt, ...);`
Действие: аналогично функции `scanf`, но чтение производится из файла `f`.
Возвращает количество успешно прочитанных значений.
В случае ошибки возвращает константу `EOF`.

`int sscanf(const char *str, const char *fmt, ...);`
Действие: аналогично функции `scanf`, но чтение производится из строки `str`.
Возвращает количество успешно прочитанных значений.
В случае ошибки возвращает константу `EOF`.

Примеры чтения до конца файла

Посимвольное чтение

```
int c; /* важно: не char! */
while (EOF != (c = fgetc(f)))
{
   /* ... */
}
```

```
int c;
while (1 == fscanf(f, "%c", &c))
{
    /* ... */
}
```

Построчное чтение

```
char s[80];
while (fgets(s, 80, f))
{
    /* ... */
}
```

Чтение пар целых чисел

```
int a, b;
while (2 == fscanf(f, "%d%d", &a, &b))
{
    /* ... */
}
/* файл либо закончился, либо далее идёт не целое число */
```

Все эти примеры можно использовать и для чтения из стандартного ввода («с клавиатуры»), если указать вместо файла переменную `stdin`.

Перенаправление ввода и вывода

При запуске программы можно указать, что ввод/вывод необходимо производить в файл/из файла. Для перенаправления ввода используется запись `< filename`, для перенаправления вывода — запись `> filename`. Например, чтобы программа `test`, находящаяся в текущем каталоге, читала данные из файла `input.txt` вместо чтения «с клавиатуры», нужно запустить её следующим образом:

```
./test < input.txt
```