C++
Information
Tutorials
Reference
Articles
Forum

Reference
C library:
<cassert> (assert.h)
<cctype> (ctype.h)
<cerrno> (errno.h)
<cfenv> (fenv.h)
<cfloat> (float.h)
<cinttypes> (inttypes.h)
<ciso646> (iso646.h)
<climits> (limits.h)
<clocale> (locale.h)
<cmath> (math.h)
<csetjmp> (setjmp.h)
<csignal> (signal.h)
<cstdarg> (stdarg.h)
<cstdbool> (stdbool.h)
<cstddef> (stddef.h)
<cstdint> (stdint.h)
<cstdio> (stdio.h)
<cstdlib> (stdlib.h)
<cstring> (string.h)
<ctgmath> (tgmath.h)
<ctime> (time.h)
<cuchar> (uchar.h)
<cwchar> (wchar.h)
<cwctype> (wctype.h)
Контейнеры:
Ввод/вывод:
Многопоточность:
Другое:

<cstdio> (stdio.h)
функции:
clearerr
fclose
feof
ферrop
fflush
fgetc
fgetpos
fgets
fopen
fprintf
fputc
fputs
fread
freopen
fscanf
fseek
fsetpos
ftell
fwrite
getc
getchar
получает
perror
printf
putc
putchar
ставит
удалить
переименовать
перемотка назад
scanf
setbuf
setvbuf
snprintf
sprintf
sscanf
tmpfile
tmpnam
ungetc
vfprintf
vfscanf
vprintf
vscanf
vsnprintf
vsprintf

function

# fprintf

<cstdio>

```
int fprintf ( FILE * stream, const char * format, ... );
```

**Write formatted data to stream**

Writes the C string pointed by *format* to the *stream*. If *format* includes *format specifiers* (subsequences beginning with %), the additional arguments following *format* are formatted and inserted in the resulting string replacing their respective specifiers.

After the *format* parameter, the function expects at least as many additional arguments as specified by *format*.

## Parameters

stream
Pointer to a FILE object that identifies an output stream.

format
Строка C, содержащая текст, который будет записан в поток.
Она может дополнительно содержать встроенные *спецификаторы формата*, которые заменяются значениями, указанными в последующих дополнительных аргументах, и форматируются по запросу.

Спецификатор *формата* следует этому прототипу:

%[flags][width][.precision][length]спецификатор

, где символ *спецификатора* в конце является наиболее значимым компонентом, поскольку он определяет тип и интерпретацию соответствующего аргумента:

| *спецификатор* | Вывод | Пример |
|---|---|---|
| d *или* i | Знаковое десятичное целое число | 392 |
| u | Десятичное целое число без знака | 7235 |
| o | Восьмеричное число без знака | 610 |
| x | Шестнадцатеричное целое число без знака | 7fa |
| X | Шестнадцатеричное целое число без знака (верхний регистр) | 7FA |
| f | Десятичная плавающая точка, нижний регистр | 392.65 |
| F | Десятичная плавающая точка, верхний регистр | 392.65 |
| e | Научная нотация (мантисса/экспонента), нижний регистр | 3.9265 e+2 |
| E | Научная нотация (мантисса/экспонента), верхний регистр | 3.9265 E+2 |
| g | Используйте самое короткое представление: %e или %f | 392.65 |
| G | Используйте самое короткое представление: %E или %F | 392.65 |
| a | Шестнадцатеричная плавающая точка, нижний регистр | -0xc.90fep-2 |
| A | Шестнадцатеричная плавающая точка, верхний регистр | -0XC.90FEP-2 |
| c | Символ | a |
| s | Строка символов | пример |
| p | Адрес указателя | b8000000 |
| n | Ничего не напечатано.<br>Соответствующий аргумент должен быть указателем на знак int.<br>Количество символов, записанных до сих пор, хранится в указанном месте. | |
| % | %, за которым следует другой символ%, запишет один % в поток. | % |

Спецификатор *формата* также может содержать подспецификаторы: *flags*, *width*,. *precision* и *модификаторы* (в этом порядке), которые являются необязательными и следуют этим спецификациям:

| *флаги* | Описание |
|---|---|
| - | Выравнивание по левому краю в пределах заданной ширины поля; по умолчанию используется выравнивание по правому *краю (см.* подспецификатор ширины). |
| + | Заставляет предварять результат знаком плюс или минус (+ или -) даже для положительных чисел. По умолчанию только отрицательным числам предшествует знак -. |
| *(пробел)* | Если знак не будет записан, перед значением вставляется пустое пространство. |
| # | Используется со спецификаторами o, x или X значение предваряется 0, 0x или 0X соответственно для значений, отличных от нуля.<br>Используется с a, A, e, E, f, F, g или G это заставляет записанный вывод содержать десятичную точку, даже если больше нет цифр. По умолчанию, если нет цифр, десятичная точка не записывается. |
| 0 | При указании заполнения слева число заполняется нулями (0) вместо пробелов (см. *подспецификатор ширины*). |

| *ширина* | Описание |
|---|---|
| *(номер)* | Минимальное количество символов для печати. Если печатаемое значение короче этого числа, результат заполняется пробелами. Значение не усекается, даже если результат больше. |
| * | *Ширина* указывается не в строке *формата*, а как дополнительный аргумент целочисленного значения, предшествующий аргументу, который должен быть отформатирован. |

| *точность* | Описание |
|---|---|
| .*номер* | Для целочисленных спецификаторов (d, i, o, u, x, X): *precision* задает минимальное количество записываемых цифр. Если записываемое значение короче этого числа, результат дополняется начальными нулями. Значение не усекается, даже если результат длиннее. *Точность* 0 означает, что для значения 0 не записывается ни один символ.<br>Для спецификаторов a, A, e, E, f и F: это количество цифр, которые будут напечатаны **после** десятичной точки (по умолчанию это 6).<br>Для спецификаторов g и G: этомаксимальное количество значащих цифр для печати. |

vsscanf
***Объекты:***
stderr
stdin
stdout
***типы:***
ФАЙЛ
fpos_t
size_t
***макроконстанты:***
BUFSIZ
EOF
FILENAME_MAX
FOPEN_MAX
L_tmpnam
NULL
TMP_MAX

| | |
|---|---|
| | Для s: это максимальное количество символов для печати. По умолчанию все символы печатаются до тех пор, пока не будет найден конечный нулевой символ.<br>Если период указан без явного значения *точности*, предполагается 0. |
| .* | *Точность* указывается не в строке *формата*, а как дополнительный аргумент целочисленного значения, предшествующий аргументу, который должен быть отформатирован. |

Подспецификатор length изменяет длину типа данных. Это диаграмма, показывающая типы, используемые для интерпретации соответствующих аргументов со *спецификатором длины и без* него (если используется другой тип, выполняется правильное продвижение или преобразование типа, если это разрешено):

| | **спецификаторы** | | | | | | |
|---|---|---|---|---|---|---|---|
| ***длина*** | **d i** | **u o x X** | **f F e E g G a A** | **c** | **s** | **p** | **n** |
| *(нет)* | int | unsigned int | двойной | int | char* | void* | int* |
| hh | подписанный символ | unsigned char | | | | | подписанный символ* |
| h | короткий int | unsigned short int | | | | | короткий int* |
| l | long int | unsigned long int | | wint_t | wchar_t* | | long int* |
| ll | long long int | unsigned long long int | | | | | long long int* |
| j | intmax_t | uintmax_t | | | | | intmax_t* |
| z | size_t | size_t | | | | | size_t* |
| t | ptrdiff_t | ptrdiff_t | | | | | ptrdiff_t* |
| L | | | длинный двойной | | | | |

Обратите внимание, что спецификатор c принимает int (или wint_t) в качестве аргумента, но выполняет правильное преобразование в значение char (или wchar_t) перед форматированием его для вывода.

**Примечание:** Желтые строки указывают спецификаторы и подспецификаторы, введенные C99. См<cinttypes>. спецификаторы для расширенных типов.

*... (дополнительные аргументы)*

*В зависимости от строки* формата функция может ожидать последовательность дополнительных аргументов, каждый из которых содержит значение, используемое для замены спецификатора *формата* в строке *формата* (или указателя на место хранения, например n).
Этих аргументов должно быть как минимум столько же, сколько и значенийуказывается в спецификаторах *формата*.
Дополнительные аргументы игнорируются функцией.

## Возвращаемое значение

При успешном выполнении возвращается общее количество записанных символов.

При возникновении ошибки записи *устанавливается индикатор* ошибки (ferror) и возвращается отрицательное число.

Если при записи широких символов возникает ошибка многобайтовой кодировки символов, errno устанавливается в EILSEQ и возвращается отрицательное число.

## Пример

```
1  /* fprintf example */
2  #include <stdio.h>
3
4  int main ()
5  {
6     FILE * pFile;
7     int n;
8     char name [100];
9
10    pFile = fopen ("myfile.txt","w");
11    for (n=0 ; n<3 ; n++)
12    {
13      puts ("please, enter a name: ");
14      gets (name);
15      fprintf (pFile, "Name %d [%-10.10s]\n",n+1,name);
16    }
17    fclose (pFile);
18
19    return 0;
20 }
```

Редактировать и запускать

В этом примере пользователь 3 раза запрашивает имя, а затем записывает его в myfile.txt каждый из них в строке фиксированной длины (всего 19 символов + новая строка).
Используются два тега формата:
%d : десятичное целое со знаком
%-10.10 s : выравнивание по левому краю (-), минимум десять символов (10), максимум десять символов (.10), строка (ы).
Предположим, что мы ввели Джона, Жан-Франсуа и Йоко в качестве имен 3, myfile.txt будет содержать:

```
Name 1 [John      ]
Name 2 [Jean-Franc]
Name 3 [Yoko      ]
```

Дополнительные примеры форматирования см. в разделе printf.

## Совместимость

Конкретные реализации библиотек могут поддерживать дополнительные *спецификаторы* и подспецификаторы.
Перечисленные здесь поддерживаются последними стандартами C и C++ (оба опубликованы в 2011 году), но те, которые выделены желтым цветом, были введены в C99 (требуются только для реализаций C++ начиная с C++11) и могут не поддерживаться библиотекамикоторые соответствуют старым стандартам.

## См. Также

| | |
|---|---|
| **printf** | Печать отформатированных данных в stdout (функция ) |
| **fscanf** | Чтение отформатированных данных из потока (функция ) |
| **fwrite** | Запись блока данных в поток (функция ) |
| **fputs** | Write string to stream (function ) |