vprintf, vfprintf, vsprintf, vsnprintf, vprintf_s, vfprintf_s, vsprintf_s, vsnprintf_s

Определено в заголовке <stdio.h>

`int vprintf( const char *format, va_list vlist );`	(1)	(до C99)
`int vprintf( const char *restrict format, va_list vlist );`		(с C99)
`int vfprintf( FILE *stream, const char *format, va_list vlist);`	(2)	(до C99)
`int vfprintf( FILE *restrict stream, const char *restrict format, va_list vlist );`		(с C99)
`int vsprintf( char *buffer, const char *format, va_list vlist);`	(3)	(до C99)
`int vsprintf( char *restrict buffer, const char *restrict format, va_list vlist );`		(с C99)
`int vsnprintf( char *restrict buffer, size_t bufsz, const char *restrict format, va_list vlist );`	(4)	(с C99)
`int vprintf_s( const char *restrict format, va_list arg);`	(5)	(начиная с C11)
`int vfprintf_s( FILE *restrict stream, const char *restrict format, va_list arg);`	(6)	(начиная с C11)
`int vsprintf_s( char *restrict buffer, rsize_t bufsz, const char *restrict format, va_list arg);`	(7)	(начиная с C11)
`int vsnprintf_s(char *restrict buffer, rsize_t bufsz, const char *restrict format, va_list arg);`	(8)	(начиная с C11)

Загружает данные из местоположений, определенных `vlist`, преобразует их в эквиваленты символьных строк и записывает результаты в различные приемники.

1) Записывает результаты в `stdout`.
2) Записывает результаты в поток файлов`stream`.
3) Записывает результаты в символьную строку`buffer`.
4) Записывает результаты в символьную строку`buffer`. Записывается не более `bufsz - 1`символов. Результирующая символьная строка будет заканчиваться нулевым символом, если `bufsz`только он не равен нулю. Если `bufsz`равно нулю, то ничего не записывается и `buffer`может быть нулевым указателем, однако возвращаемое значение (количество байтов, которые будут записаны без учета нулевого терминатора) все равно вычисляется и возвращается.

5-8) Такой же, как (1-4), за исключением того, что следующие ошибки обнаруживаются во время выполнения и вызывают установленную в данный момент функцию обработчика ограничений:

- спецификатор преобразования `%n`присутствует в `format`
- любой из аргументов, соответствующих `%s`нулевому указателю
- `format` или `buffer`это нулевой указатель
- `bufsz` равен нулю или больше `RSIZE_MAX`
- ошибки кодирования возникают в любом из спецификаторов преобразования строк и символов
- (`vsprintf_s`только для), строка для хранения `buffer`(включая конечный `null`)) будет превышать `bufsz`

Как и во всех функциях с проверкой границ, `vprintf_s`, `vfprintf_s`, `vsprintf_s`, и `vsnprintf_s`гарантированно доступны только в том случае, если `__STDC_LIB_EXT1__` определяется реализацией и если пользователь определяет `__STDC_WANT_LIB_EXT1__` для целочисленной константы 1 перед включением `stdio.h`.

Параметры

поток	-	поток выходного файла для записи
буфер	-	указатель на символьную строку для записи
bufsz	-	может быть записано до bufsz - 1 символов плюс нулевой терминатор
формат	-	указатель на символьную строку с нулевым окончанием, указывающую, как интерпретировать данные
vlist	-	список аргументов переменной, содержащий данные для печати.

Строка **формата** состоит из обычных многобайтовых символов (кроме `%`), которые копируются без изменений в выходной поток, и спецификаций преобразования. Каждая спецификация преобразования имеет следующий формат:

- вводный `%`символ
- (необязательно) один или несколько флагов, изменяющих поведение преобразования:

- **-**: результат преобразования выравнивается по левому краю поля (по умолчанию выравнивается по правому краю)
- **+**: знак подписанных преобразований всегда предшествует результату преобразования (по умолчанию результату предшествует минус только тогда, когда он отрицательный)
- *пробел*: если результат преобразования со знаком не начинается со знака или пуст, к результату добавляется пробел. Он игнорируется, если +флаг присутствует.
- **#** : *выполняется альтернативная форма* преобразования. Точные эффекты см. В таблице ниже, в противном случае поведение не определено.
- **0** : для преобразования целых чисел и чисел с плавающей запятой начальные нули используются для заполнения поля вместо пробелов. Для целых чисел он игнорируется, если точность указана явно. Для других преобразований использование этого флага приводит к неопределенному поведению. Он игнорируется, если -флаг присутствует.

- (необязательно) целочисленное значение или *задающее минимальную ширину поля. Результат дополняется пробелами (по умолчанию), если требуется, слева при выравнивании по правому краю или справа при выравнивании по левому краю. В случае, когда *используется, ширина задается дополнительным аргументом типа `int`, который появляется перед преобразуемым аргументом и аргументом, обеспечивающим точность, если он указан. Если значение аргумента отрицательное, то результат с указанным -флагом и положительной шириной поля. (Примечание: это минимальная ширина: значение никогда не усекается.)

- (необязательно) . за которым следует целое число или *, или ни то, ни другое, указывающее *точность* преобразования. В случае, когда *используется, *точность* задается дополнительным аргументом типа `int`, который появляется перед преобразуемым аргументом, но после аргумента, указывающего минимальную ширину поля, если он указан. Если значение этого аргумента отрицательное, оно игнорируется. Если ни число, ни *используется, точность принимается равной нулю. См. Таблицу ниже для точных эффектов *точности*.

- (необязательно) *модификатор длины*, задающий размер аргумента (в сочетании со спецификатором формата преобразования задает тип соответствующего аргумента)

- спецификатор формата преобразования

Доступны следующие спецификаторы формата:

Спецификатор преобразования	Объяснение	Ожидаемый тип аргумента								
	Модификатор длины →	hh (C99)	h	(нет)	l	ll (C99)	j (C99)	z (C99)	t (C99)	L
%	пишет литерал %. Полная спецификация преобразования должна быть%%.	N/A	N/A	N/A	N/A	N/A	N/A	N/A	N/A	N/A
c	записывает **один символ.** Аргумент сначала преобразуется в `unsigned char`. Если используется модификатор l, аргумент сначала преобразуется в символьную строку, как если **бы %ls** с `аргументом wchar_t[2]`.	N/A	N/A	`int`	`wint_t`	N/A	N/A	N/A	N/A	N/A
s	записывает **символьную строку** Аргумент должен быть указателем на начальный элемент массива символов. *Precision* указывает максимальное количество записываемых байтов. Если *точность* не указана, записывает каждый байт до первого нулевого терминатора, не включая его. Если используется спецификатор l, аргумент должен быть указателем на начальный элемент массива `wchar_t`, который преобразуется в массив char как бы вызовом wcrtomb с нулевым инициализированным состоянием преобразования.	N/A	N/A	`char*`	`wchar_t*`	N/A	N/A	N/A	N/A	N/A
d i	преобразует целое **число со знаком** в десятичное представление *[-]dddd*. *Точность* указывает минимальное количество цифр, которые должны отображаться. Точность по умолчанию `равна 1`. Если преобразованное значение и точность равны `0`, преобразование не приводит к символам.	`подписанный символ`	`короткое`	`int`	`длинный`	`долго долго`	`intmax_t`	подписанный `size_t`	`ptrdiff_t`	N/A
o	преобразует целое **число без знака** в восьмеричное представление *oooo*. *Точность* указывает минимальное количество цифр, которые должны отображаться. Точность по умолчанию `равна 1`. Если преобразованное значение и точность равны `0`, преобразование не приводит к символам. В *альтернативной реализации* точность увеличивается, если необходимо записать один ведущий ноль. В этом случае, если и преобразованное значение, и точность равны `0`, записывается одиночный `0`.	`символ без знака`	`без знака короткий`	`unsigned int`	`без знака долго`	`без знака long long`	`uintmax_t`	`size_t`	неподписанная версия `ptrdiff_t`	N/A
x X	преобразует целое число без знака в шестнадцатеричное представление *hhhh*. Для **x**преобразования abcdefиспользуются буквы. Для **X**преобразования ABCDEFиспользуются буквы. *Точность* указывает минимальное количество цифр, которые должны отображаться. Точность по умолчанию `равна 1`. Если преобразованное значение и точность равны `0`, преобразование не приводит к символам. В *альтернативной реализации* **0x** или **0X**имеет префикс results , если преобразованное значение не равно нулю.									N/A
u	преобразует целое **число без знака** в десятичное представление *dddd*. *Точность* указывает минимальное количество цифр, которые должны отображаться. Точность по умолчанию `равна 1`. Если преобразованное значение и точность равны `0`, преобразование не приводит к символам.									N/A
f F	преобразует **число с плавающей запятой** в десятичную систему счисления в стиле *[-]ddd.ddd*. *Точность* указывает точное количество цифр, которые будут отображаться после символа десятичной точки. Точность по умолчанию `равна 6`. В *альтернативной реализации* символ десятичной точки записывается, даже если за ним не следуют цифры. Стиль преобразования бесконечности и не-числа см. В примечаниях.	N/A	N/A	`двойной`	`двойной` (C99)	N/A	N/A	N/A	N/A	`длинный двойной`
e E	преобразует **число** с плавающей запятой в десятичную систему счисления. Для **e**преобразования используется стиль *[-]d.ddd***e**±*dd*. Для **E**преобразования используется стиль *[-]d.ddd***E**±*dd*. Показатель степени содержит не менее двух цифр, больше цифр используется только в случае необходимости. Если значение равно `0`, показатель степени также `равен 0`. *Точность* указывает точное количество цифр, которые будут отображаться после символа десятичной точки. Точность по умолчанию `равна 6`. В *альтернативной реализации*	N/A	N/A			N/A	N/A	N/A	N/A	

	символ десятичной точки записывается, даже если за ним не следуют цифры. Стиль преобразования бесконечности и не-числа см. В примечаниях.										
a **A** (C99)	преобразует **число** с плавающей запятой в шестнадцатеричную экспоненту. Для **a**преобразования используется стиль *[-]***0x***h.hhh***p±d***.* Для **A**преобразования используется стиль *[-]***0X***h.hhh***P±d***.* Первая шестнадцатеричная цифра не 0является, если аргумент является нормализованным значением с плавающей запятой. Если значение равно `0`, показатель степени также `равен 0`. *Точность* указывает точное количество цифр, которые будут отображаться после шестнадцатеричного символа точки. Точность по умолчанию достаточна для точного представления значения. В *альтернативной реализации* символ десятичной точки записывается, даже если за ним не следуют цифры. Стиль преобразования бесконечности и не-числа см. В примечаниях.	N/A	N/A				N/A	N/A	N/A	N/A	
g **G**	преобразует **число с плавающей запятой** в десятичную или десятичную экспоненту в зависимости от значения и *точности*. Для **g**стиля преобразования **ef**будет выполнено преобразование со стилем или. Для **G**стиля преобразования **EF**будет выполнено преобразование со стилем или. Пусть Pравна точности, если ненулевая, `6`, если точность не указана, или `1`, если точность равна `0`. Тогда, если преобразование со стилем Ебудет иметь показательX: ▪ если *P> X ≥ -4*, преобразование выполняется со стилем **f**или **F**и точностью *P - 1 - X.* ▪ в противном случае преобразование выполняется со стилем **e**или **E**и точностью *P - 1*. Если *не запрашивается альтернативное представление*, конечные нули удаляются, а также символ десятичной точки удаляется, если не осталось дробной части. Стиль преобразования бесконечности и не-числа см. В примечаниях.	N/A	N/A				N/A	N/A	N/A	N/A	
n	возвращает **количество символов, записанных** до сих пор этим вызовом функции. Результат *записывается* в значение, на которое указывает аргумент. Спецификация может не содержать никакого *флага*, *ширины поля*или *точности*.	`подписанный символ*`	`короткое*`	`инт*`	`длинный*`	`долго долго*`	`intmax_t*`	подписанный `size_t*`	`ptrdiff_t*`	N/A	
p	записывает определенную реализацией последовательность символов, определяющую **указатель**.	N/A	N/A	`пустота*`	N/A	N/A	N/A	N/A	N/A	N/A	

Функции преобразования с плавающей запятой преобразуют бесконечность в `inf`или `infinity`. Какой из них используется, определяется реализацией.

Not-a-number преобразуется в `nan`or `nan(`*`char_sequence`*`)`. Какой из них используется, определяется реализацией.

Вместо этого преобразования **F**, **E**, **G**, **A**выводят `INF`, `INFINITY`, `NAN`.

Несмотря `%c`на то, что ожидает `int`аргумент, безопасно передавать a `char`из-за целочисленного продвижения, которое происходит при вызове переменной функции.

Правильные спецификации преобразования для типов символов фиксированной ширины (`int8_t`и т. Д.) Определены в заголовке `<inttypes.h>`(хотя `PRIdMAX`, `PRIuMAX` ит. Д. Являются синонимами `%jd`, `%ju`и т. Д.).

Спецификатор преобразования для записи в память `%n` является общей целью эксплойтов безопасности, где строки формата зависят от пользовательского ввода и не поддерживаются семейством функций с проверкой границ`printf_s`.

После действия каждого спецификатора преобразования существует точка последовательности; это позволяет хранить несколько результатов `%n` в одной и той же переменной или, в крайнем случае, печатать строку, измененную более ранним `%n` в том же вызове.

Если спецификация преобразования недопустима, то поведение не определено.

Возвращаемое значение

1-3) Количество символов, записанных в случае успеха или отрицательное значение, если произошла ошибка.

4) Количество символов, записанных в случае успеха или отрицательное значение, если произошла ошибка. Если результирующая строка усекается из-за `buf_size`ограничения, функция возвращает общее количество символов (не включая завершающий нулевой байт), которые были бы записаны, если бы ограничение не было наложено.

5,6) количество символов, передаваемых в выходной поток, или отрицательное значение, если произошла ошибка вывода, ошибка нарушения констант времени выполнения или ошибка кодирования.

7) количество символов, записанных в buffer, не считая нулевого символа (который всегда записывается до тех пор, пока bufferне является нулевым указателем и bufszне равен нулю и не большеRSIZE_MAX), или ноль при нарушениях ограничений времени выполнения и отрицательное значение при ошибках кодирования

8) количество символов, не включающих завершающий нулевой символ (который всегда записывается до тех пор, пока bufferне является нулевым указателем и bufszне равен нулю и не большеRSIZE_MAX), который был бы записанbuffer, если bufszбы был проигнорирован, или отрицательное значение, если бы произошло нарушение ограничений времени выполнения или ошибка кодирования

Все эти функции вызывают va_arg по крайней мере один раз, значение argнеопределенно после возврата. Эти функции не вызывают va_end, и это должно быть сделано вызывающим.

vsnprintf_s, в отличие vsprintf_sот , будет усекать результат, чтобы он помещался в массив, на который указывает by buffer.

Запустите этот код

```
#include <stdio.h>
#include <stdarg.h>
#include <time.h>

void debug_log(const char *fmt, ...)
{
    struct timespec ts;
 timespec_get(&ts, TIME_UTC);
    char time_buf[100];
    size_t rc = strftime(time_buf, sizeof time_buf, "%D %T", gmtime(&ts.tv_sec));
    snprintf(time_buf + rc, sizeof time_buf - rc, ".%06ld UTC", ts.tv_nsec / 1000);

 va_list args1;
 va_start(args1, fmt);
 va_list args2;
 va_copy(args2, args1);
    char buf[1+vsnprintf(NULL, 0, fmt, args1)];
 va_end(args1);
 vsnprintf(buf, sizeof buf, fmt, args2);
 va_end(args2);

    printf("%s [debug]: %s\n", time_buf, buf);
}

int main(void)
{
 debug_log("Ведение журнала, %d, %d, %d", 1, 2, 3);
}
```

Возможный выход:

```
20.02.15 21:58:09.072683 UTC [debug]: Logging, 1, 2, 3
```

Ссылки

- Стандарт C11 (ISO/IEC 9899:2011):
  - 7.21.6.8 Функция vfprintf (p: 326-327)
  - 7.21.6.10 Функция vprintf (p: 328)
  - 7.21.6.12 Функция vsnprintf (p: 329)
  - 7.21.6.13 Функция vsprintf (p: 329)
  - K.3.5.3.8 Функция vfprintf_s (p: 597)
  - K.3.5.3.10 Функция vprintf_s (p: 598-599)
  - K.3.5.3.12 Функция vsnprintf_s (p: 600)
  - K.3.5.3.13 Функция vsprintf_s (p: 601)
- Стандарт C99 (ISO/IEC 9899:1999):

- 7.19.6.8 Функция vfprintf (p: 292)
- 7.19.6.10 Функция vprintf (p: 293)
- 7.19.6.12 Функция vsnprintf (p: 294)
- 7.19.6.13 Функция vsprintf (p: 295)

- Стандарт C89/C90 (ISO /IEC 9899:1990):
  - 4.9.6.7 Функция vfprintf
  - 4.9.6.8 Функция vprintf
  - 4.9.6.9 Функция vsprintf

См. Также

vwprintf (C95) **vfwprintf** (C95) **vswprintf** (C95) **vwprintf_s** (C11) **vfwprintf_s** (C11) **vswprintf_s** (C11) **vsnwprintf_s** (C11)	печатает форматированный вывод широкого символа в stdout, поток файла или буфер с использованием списка аргументов переменной (функции)
printf **fprintf** **sprintf** **snprintf** (C99) **printf_s** (C11) **fprintf_s** (C11) **sprintf_s** (C11) **snprintf_s** (C11)	печатает форматированный вывод в stdout, поток файла или буфер (функция)
vscanf (C99) **vfscanf** (C99) **vsscanf** (C99) **vscanf_s** (C11) **vfscanf_s** (C11) **vsscanf_s** (C11)	считывает форматированные входные данные из stdin, потока файлов или буфера с помощью списка аргументов переменной (функции)
C ++ документация для **vprintf**, **vfprintf**, **vsprintf**, **vsnprintf**	