# vscanf, vfscanf, vsscanf, vscanf_s, vfscanf_s, vsscanf_s

Определено в заголовке <stdio.h>

| | | |
|---|---|---|
| `int vscanf( const char *restrict format, va_list vlist );` | (1) | (начиная с C99) |
| `int vfscanf( FILE *restrict stream, const char *restrict format, va_list vlist );` | (2) | (начиная с C99) |
| `int vsscanf( const char *restrict buffer, const char *restrict format, va_list vlist );` | (3) | (начиная с C99) |
| `int vscanf_s(const char *restrict format, va_list vlist);` | (4) | (начиная с C11) |
| `int vfscanf_s( FILE *restrict stream, const char *restrict format, va_list vlist);` | (5) | (начиная с C11) |
| `int vsscanf_s( const char *restrict buffer, const char *restrict format, va_list vlist);` | (6) | (начиная с C11) |

Считывает данные из различных источников, интерпретирует их в соответствии с `format` и сохраняет результаты в местах, определенных `vlist`.

1) Считывает данные из `stdin`
2) Считывает данные из потока файлов `stream`
3) Считывает данные из символьной строки с нулевым окончанием `buffer`. Достижение конца строки эквивалентно достижению условия конца файла для `fscanf`

4-6) То же, что и (1-3), за исключением того, что `%c`, `%s` и `%[` спецификаторы преобразования ожидают два аргумента (обычный указатель и значение типа `rsize_t`, указывающее размер принимающего массива, который может быть равен 1 при чтении с `%c` в один символ) и за исключением что следующие ошибки обнаруживаются во время выполнения и вызывают установленную в данный момент функцию обработчика ограничений:

- любой из аргументов типа указателя является нулевым указателем
- `format`, `stream`, или `buffer` является нулевым указателем
- количество символов, которые будут записаны с помощью `%c`, `%s` или `%[` , плюс завершающий нулевой символ, превысит второй аргумент (`rsize_t`), предоставленный для каждого из этих спецификаторов преобразования
- опционально, любая другая обнаруживаемая ошибка, такая как неизвестный спецификатор преобразования

Как и во всех функциях с проверкой границ, `vscanf_s`, `vfscanf_s`, и `vsscanf_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 входного файла для чтения
**буфер** - указатель на символьную строку с нулевым окончанием для чтения
**формат** - указатель на символьную строку с нулевым окончанием, указывающую, как читать входные данные
**vlist** - список аргументов переменной, содержащий аргументы получателя.

Строка **формата** состоит из

- небельные многобайтовые символы, за исключением `%`: каждый такой символ в строке формата потребляет ровно один идентичный символ из входного потока или вызывает сбой функции, если следующий символ в потоке не сравнивается равным.
- символы пробела: любой отдельный символ пробела в строке формата потребляет все доступные последовательные символы пробела из входных данных (определяется как вызов `isspace` в цикле). Обратите внимание, что нет никакой разницы между `"\ n"`, `" "`, `"\ t\ t"` или другими пробелами в строке формата.
- технические характеристики преобразования. Каждая спецификация преобразования имеет следующий формат:
  - вводный `%` символ
  - (необязательно) присваивание-подавляющий символ `*`. Если этот параметр присутствует, функция не назначает результат преобразования ни одному получающему аргументу.
  - (необязательно) целое число (больше нуля), задающее *максимальную ширину поля*, то есть максимальное количество символов, которые функция может использовать при выполнении преобразования, указанного в текущей спецификации преобразования. Обратите внимание, что `%s` и `%[` могут привести к переполнению буфера, если ширина не указана.

- (необязательно) *модификатор длины*, определяющий размер принимающего аргумента, то есть фактический тип назначения. Это влияет на точность преобразования и правила переполнения. Тип назначения по умолчанию отличается для каждого типа преобразования (см. Таблицу ниже).
- спецификатор формата преобразования

Доступны следующие спецификаторы формата:

| Спецификатор преобразования | Объяснение | Тип аргумента | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|
| | **Модификатор длины →** | **hh** (C99) | **h** | (нет) | **l** | **ll** (C99) | **j** (C99) | **z** (C99) | **t** (C99) | **L** |
| **%** | соответствует буквальному % | N/A | N/A | N/A | N/A | N/A | N/A | N/A | N/A | N/A |
| **c** | соответствует **символу** или последовательности **символов**<br>Если используется спецификатор ширины, он точно соответствует символам ширины (аргумент должен быть указателем на массив с достаточным пространством). В отличие от %s и %[ , не добавляет нулевой символ в массив. | N/A | N/A | голец* | wchar_t* | N/A | N/A | N/A | N/A | N/A |
| **s** | соответствует последовательности символов без пробелов (**строка**)<br>Если используется спецификатор ширины, соответствует *ширине* или до первого пробельного символа, в зависимости от того, что появится первым. Всегда хранит нулевой символ в дополнение к совпадающим символам (поэтому массив аргументов должен иметь место как минимум для *символов width + 1*) | | | | | | | | | |
| **[*набор*]** | соответствует непустой последовательности символов из *набора* символов.<br>Если первый символ набора равен^, то все символы, не входящие в набор, совпадают. Если набор начинается с ]или^], то ]символ также включается в набор. Определяется реализацией, может ли символ -в начальной позиции в наборе развертки указывать диапазон, как в [0-9]. Если используется спецификатор ширины, соответствует только *ширине*. Всегда хранит нулевой символ в дополнение к совпадающим символам (поэтому массив аргументов должен иметь место как минимум для *символов width + 1*) | | | | | | | | | |
| **d** | соответствует **десятичному целому**числу.<br>Формат числа такой же, как и ожидалось strtol() со значением 10 для baseаргумента | символ со знаком* или символ без знака* | короткое со знаком* или короткое без знака* | signed int* или unsigned int* | signed long* или unsigned long* | signed long long* или unsigned long long* | intmax_t* или uintmax_t* | size_t* | ptrdiff_t* | N/A |
| **i** | соответствует целому **числу**.<br>Формат числа такой же, как и ожидалось strtol() со значением 0 для baseаргумента (база определяется первыми проанализированными символами). | | | | | | | | | |
| **u** | соответствует **десятичному целому**числу без знака.<br>Формат числа такой же, как и ожидалось strtoul() со значением 10 для baseаргумента. | | | | | | | | | |
| **o** | соответствует **восьмеричному целому числу без знака.**<br>Формат числа такой же, как и ожидалось strtoul() со значением 8 для baseаргумента | | | | | | | | | |
| **x, X** | соответствует **шестнадцатеричному целому**числу без знака.<br>Формат числа такой же, как и ожидалось strtoul() со значением 16 для baseаргумента | | | | | | | | | |
| **n** | возвращает **количество прочитанных символов.**<br>Вход не потребляется. Не увеличивает количество назначений. Если в спецификаторе определен оператор подавления присваивания, поведение не определено | | | | | | | | | |
| **a, A**(C99)<br>**e, E**<br>**f,F**<br>**g, G** | соответствует **числу с плавающей запятой.**<br>Формат числа такой же, как и ожидалось strtof() | N/A | N/A | поплавок* | двойной* | N/A | N/A | N/A | N/A | длинный двойной* |
| **p** | соответствует определенной реализацией последовательности символов, определяющей **указатель.**<br>printf семейство функций должно выдавать одну и ту же последовательность с использованием **%p**спецификатора формата | N/A | N/A | пустота** | N/A | N/A | N/A | N/A | N/A | N/A |

Для каждого спецификатора преобразования, отличного от n, самая длинная последовательность входных символов, которая не превышает заданной ширины поля и которая либо является именно тем, что ожидает спецификатор преобразования, либо является префиксом последовательности, которую он ожидает, – это то, что потребляется из потока. Первый символ, если таковой имеется, после этой используемой последовательности остается непрочитанным. Если

потребляемая последовательность имеет нулевую длину или если потребляемая последовательность не может быть преобразована, как указано выше, сбой сопоставления происходит, если только конец файла, ошибка кодирования или ошибка чтения не предотвратили ввод из потока, и в этом случае это сбой ввода.

Все спецификаторы преобразования, отличные от [, c, и nпотребляют и отбрасывают все ведущие символы пробела (определяемые как бы вызовом isspace) перед попыткой анализа входных данных. Эти потребляемые символы не учитываются при указанной максимальной ширине поля.

Спецификаторы преобразования lc, ls, и l[выполняют многобайтовое преобразование символов, как если бы вызывали mbrtowc() с объектом mbstate_t, инициализированным до нуля перед преобразованием первого символа.

Спецификаторы преобразования s и [всегда сохраняют нулевой терминатор в дополнение к соответствующим символам. Размер целевого массива должен быть как минимум на единицу больше указанной ширины поля. Использование `%s` или `%[` без указания размера целевого массива так же небезопасно, как и gets

Правильные спецификации преобразования для целочисленных типов с фиксированной шириной (int8_t, etc) Определены в заголовке <inttypes.h>(хотя SCNdMAX, SCNuMAX ит. Д. Являются синонимами %jd, %ju, etc) .

После действия каждого спецификатора преобразования существует точка последовательности; это позволяет хранить несколько полей в одной переменной "приемник".

При анализе неполного значения с плавающей запятой, которое заканчивается показателем без цифр, например при анализе `"100er"` со спецификатором преобразования `%f`, последовательность `"100e"` (самый длинный префикс возможно допустимого числа с плавающей запятой) потребляется, что приводит к ошибке сопоставления (потребляемая последовательностьне может быть преобразовано в число с плавающей запятой), при `этом остается "r"`. Некоторые существующие реализации не следуют этому правилу и откатываются, чтобы потреблять только `"100"`, оставляя `"er"`, например ошибка glibc 1765 (https://sourceware.org/bugzilla/show_bug.cgi?id=1765)

## Возвращаемое значение

1-3) Количество успешно назначенных аргументов приема или EOF, если сбой чтения произошел до назначения первого аргумента приема.

4-6) То же, что и (1-3), за исключением того, что EOF также возвращается, если есть нарушение ограничений времени выполнения.

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 абонентом.

Запустите этот код

```
#включить <stdio.h>
#включить <stdbool.h>
#включить <stdarg.h>

bool checked_sscanf(int count, const char* buf, const char *fmt, ...)
{
 va_list ap;
 va_start(ap, fmt);
    int rc = vsscanf(buf, fmt, ap);
 va_end(ap);
    return rc == count;
}

int main(void)
{
    int n, m;

    printf("Разбор '1 2'...");
    if(checked_sscanf(2, "1 2", "% d %d", &n, &m))
        puts("success");
    else
        puts("failure");

    printf("Parsing '1 a'...");
    if(checked_sscanf(2, "1 a", "%d %d", &n, &m))
        puts("success");
    elseputs
        ("сбой");
}
```

Выход:

```
Разбор '1 2' ... успех
Разбор '1 a' ... сбой
```

- Стандарт C11 (ISO/IEC 9899:2011):
  - 7.21.6.9 Функция vfscanf (p: 327)
  - 7.21.6.11 Функция vscanf (p: 328)
  - 7.21.6.14 Функция vsscanf (p: 330)
  - K.3.5.3.9 Функция vfscanf_s (p: 597-598)
  - K.3.5.3.11 Функция vscanf_s (p: 599)
  - K.3.5.3.14 Функция vsscanf_s (p: 602)
- Стандарт C99 (ISO/IEC 9899:1999):
  - 7.19.6.9 Функция vfscanf (p: 293)
  - 7.19.6.11 Функция vscanf (p: 294)
  - 7.19.6.14 Функция vsscanf (p: 295)

## См. также

| | |
|---|---|
| **scanf**<br>**fscanf**<br>**sscanf**<br>**scanf_s** (C11)<br>**fscanf_s** (C11)<br>**sscanf_s** (C11) | считывает форматированные входные данные из stdin, потока файлов или буфера<br>(функции) |
| **vprintf**<br>**vfprintf**<br>**vsprintf**<br>**vsnprintf** (C99)<br>**vprintf_s** (C11)<br>**vfprintf_s** (C11)<br>**vsprintf_s** (C11)<br>**vsnprintf_s** (C11) | печатает форматированный вывод в stdout, поток файла или буфер<br>с использованием списка аргументов переменной<br>(функции) |
| **C ++ документация** для **vscanf**, **vfscanf**, **vsscanf** | |