

fgetc	
fgetpos	
fgets	
fopen	
fprintf	
fputc	
··· fputs	
fread	
freopen	
fscanf	
fseek	
fsetpos	
···· ftell	
···· fwrite	
getc getc	
getchar	
gets gets	
perror	
··· printf	
putc	
putchar	
puts	
remove	
rename	
··· rewind	
··· scanf	
··· setbuf	
··· setvbuf	
···· snprintf	C++II
sprintf	
··· sscanf	
··· tmpfile	
tmpnam	
ungetc	
···· vfprintf	
···· vfscanf	C++II
···· vprintf	
1000	C++II
To the state of th	C++II
···· vsprintf	
e objects:	
stderr	
stdin	
stdout	
types:	
FILE	
fpos_t	
size_t	
macro constants:	
BUFSIZ	
EOF	
FILENAME_MAX	

		Decimal digits assumed by default (0-9), but a 0 prefix introduces octal digits (0-7), and $0 \times$ hexadecimal digits (0-f). Signed argument.	
d <i>or</i> u	Decimal integer	Any number of decimal digits (0-9), optionally preceded by a sign (+ or -). d is for a signed argument, and u for an unsigned.	
0	Octal Any number of octal digits (0-7), optionally preceded by a sign (+ or -). Unsigned argument.		
x	Hexadecimal integer	Any number of hexadecimal digits (0-9, a-f, A-F), optionally preceded by $0x$ or $0x$, and all optionally preceded by a sign (+ or -). Unsigned argument.	
f, e, g	Floating point	A series of decimal digits, optionally containing a decimal point, optionally preceded by a sign (+ or -) and optionally followed by the e or E character and a decimal integer (or some of the other sequences supported by strtod).	
a	number	Implementations complying with C99 also support hexadecimal floating-point format when preceded by $0x$ or $0x$.	
С	Character	The next character. If a <i>width</i> other than 1 is specified, the function reads exactly <i>width</i> characters and stores them in the successive locations of the array passed as argument. No null character is appended at the end.	
s	String of characters	Any number of non-whitespace characters, stopping at the first whitespace character found. A terminating null character is automatically added at the end of the stored sequence.	
р	Pointer address	A sequence of characters representing a pointer. The particular format used depends on the system and library implementation, but it is the same as the one used to format p in fprintf.	
[characters]	Scanset	Any number of the characters specified between the brackets. A dash (-) that is not the first character may produce non-portable behavior in some library implementations.	
[^characters]	Negated scanset	Any number of characters none of them specified as <i>characters</i> between the brackets.	
n	Count	No input is consumed. The number of characters read so far from stdin is stored in the pointed location.	
8	8	A % followed by another % matches a single %.	

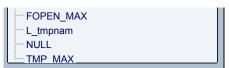
Except for n, at least one character shall be consumed by any specifier. Otherwise the match fails, and the scan ends there.

The format specifier can also contain sub-specifiers: asterisk (\star), width and length (in that order), which are optional and follow these specifications:

sub- specifier	description
*	An optional starting asterisk indicates that the data is to be read from the stream but ignored (i.e. it is not stored in the location pointed by an argument).
width	Specifies the maximum number of characters to be read in the current reading operation (optional).
uenatn	One of hh, h, 1, 11, j, z, t, L (optional). This alters the expected type of the storage pointed by the corresponding argument (see below).

This is a chart showing the types expected for the corresponding arguments where input is stored (both with and without a *length* sub-specifier):

	specifiers					
length	di	иож	fega	c s [] [^]	р	n
(none)	int*	unsigned int*	float*	char*	void**	int*



hh	signed char*	unsigned char*			signed char*
h	short int*	unsigned short int*			short int*
1	long int*	unsigned long int*	double*	wchar_t*	long int*
11	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			long double*		

Note: Yellow rows indicate specifiers and sub-specifiers introduced by C99.

... (additional arguments)

Depending on the *format* string, the function may expect a sequence of additional arguments, each containing a pointer to allocated storage where the interpretation of the extracted characters is stored with the appropriate type.

There should be at least as many of these arguments as the number of values stored by the *format specifiers*. Additional arguments are ignored by the function.

These arguments are expected to be pointers: to store the result of a scanf operation on a regular variable, its name should be preceded by the *reference operator* (&) (see example).

Return Value

On success, the function returns the number of items of the argument list successfully filled. This count can match the expected number of items or be less (even zero) due to a matching failure, a reading error, or the reach of the *end-of-file*.

If a reading error happens or the *end-of-file* is reached while reading, the proper indicator is set (feof or ferror). And, if either happens before any data could be successfully read, EOF is returned.

If an encoding error happens interpreting wide characters, the function sets errno to EILSEQ.

₹

Example

```
1 /* scanf example */
 2 #include <stdio.h>
 4 int main ()
 5 {
 6 char str [80];
    int i;
   printf ("Enter your family name: ");
10 scanf ("%79s", str);
                                               Bedit & Run
printf ("Enter your age: ");
12 scanf ("%d",&i);
printf ("Mr. %s , %d years old.\n", str,i);
printf ("Enter a hexadecimal number: ");
15 scanf ("%x",&i);
    printf ("You have entered %#x (%d).\n",i,i);
18 return 0;
19 }
```

This example demonstrates some of the types that can be read with scanf:

Enter your family name: Soulie Enter your age: 29 Mr. Soulie , 29 years old. Enter a hexadecimal number: ff You have entered 0xff (255).

Compatibility

Particular library implementations may support additional specifiers and sub-specifiers.

Those listed here are supported by the latest C and C++ standards (both published in 2011), but those in yellow were introduced by C99 (only required for C++ implementations since C++11), and may not be supported by libraries that comply with older standards.

₱ See also

fscanf	Read formatted data from stream (function)
printf	Print formatted data to stdout (function)
gets	Get string from stdin (function)
fopen	Open file (function)

Home page | Privacy policy © cplusplus.com, 2000-2020 - All rights reserved - v3.2 Spotted an error? contact us