Função printf

Int printf (cons char * format, …);

Imprimir dados formatados para stdout

Escreve uma string C apontada pelo **format** na saída padrão (stdout). Se o **format** incluir **especificadores de formato** (subsequências começando com %), os argumentos adicionais após o **format** serão formatados e inseridos na string resultante substituindo seus respectivos especificadores.

Parâmetros

format

String C que contém o texto a ser escrito em stdout.

Ele pode, opcionalmente, conter **especificadores de formato** integrados que são substituídos pelos valores especificados em argumentos adicionais subsequentes e formatados conforme solicitado.

Um **especificador de formato** segue o seguinte protótipo:

%[flags][width][.precision][lenght] especificador

Onde o **caractere especificador** no final é o componente mais significativo, pois define o tipo e a interpretação de seu argumento correspondente:

Especificador	Saída	Exemplo
d ou i	Inteiro decimal sinalizado	123
u	Inteiro decimal não sinalizado	1234
o	Inteiro octal não sinalizado	765
x	Inteiro hexadecimal não sinalizado (minúsculo)	8fb
X	Inteiro hexadecimal não sinalizado (maiúsculo)	8FB
f	Ponto decimal flutuante (minúsculo)	123.45
F	Ponto decimal flutuante (maiúsculo)	123.45
e	Notação cientifica (minúsculo)	1.2345e+3
E	Notação cientifica (maiúsculo)	1.2345E+3
g	Usar a representação curta: %e ou %f	123.45
G	Usar a representação curta: %E ou %F	123.45
a	Ponto hexadecimal flutuante (minúsculo)	-0xc.90fep-2
A	Ponto hexaecimal flutuante (maiúsculo)	-0CX.90FEP-2
c	Caractér	a
s	String de caracteres	amostra
p	Ponteiro de endereço	b400000
n	1) Não imprime nada 2) O argumento correspondente deve ser um ponteiro para um **signed int** 3) O número de caracteres escritos até agora é armazenado no local apontado	
%	Um % seguido por outro caracter % escreverá um único caracter % para o stream	%

O **especificador de formato** também pode conter sub-especificadores: **flags**, **width**, **.precision** e **modifiers** (nessa ordem), que são opcionais e seguem as seguintes especificações:

flags	descrições
-	Justificar à esquerda dentro da largura de campo dado; A justificação à direita é o padrão (consulte o sub-especificador **width**).
+	Força o preceder do resultado com um sinal de mais ou menos (+ ou -) mesmo para números positivos. Por padrão, apenas números negativos são precedidos por um sinal -
(space)	Se nenhum sinal for escrito, um espaço em branco é inserido antes do valor.
#	- Usado com os especificadores o, x ou X, o valor é precedido por 0, 0x ou 0X, respectivamente, para valores diferentes de zero. - Usado com a, A, e, E, f, F, g ou G, força a saída escrita a conter um ponto decimal, mesmo que não haja mais dígitos a seguir. Por padrão, se nenhum dígito for seguido, nenhum ponto decimal será escrito.
0	Preenche à esquerda o número com zeros (0) em vez de espaços quando o preenchimento é especificado (consulte o subespecificador **width**).

width	descrições
(number)	Número mínimo de caracteres a serem impressos. Se o valor a ser impresso for menor que esse número, o resultado será preenchido com espaços em branco. O valor não é truncado mesmo se o resultado for maior.
*	A largura (width) não é especificada na string de formato, mas como um argumento de valor inteiro adicional precedendo o argumento que deve ser formatado.

.precision	descrições
.number	- Para especificadores inteiros (d, i, o, u, x, X): precisão especifica o número mínimo de dígitos a serem escritos. Se o valor a ser escrito for menor que esse número, o resultado será preenchido com zeros à esquerda. O valor não é truncado, mesmo que o resultado seja mais longo. Uma precisão de 0 significa que nenhum caractere é escrito para o valor 0. - Para especificadores a, A, e, E, f e F: este é o número de dígitos a serem impressos após o ponto decimal (por padrão, é 6). - Para especificadores g e G: Este é o número máximo de dígitos significativos a serem impressos. - Para s: este é o número máximo de caracteres a serem impressos. Por padrão, todos os caracteres são impressos até que o caractere nulo final seja encontrado. - Se o período for especificado sem um valor explícito para precisão, 0 será assumido.
.*	O **precision** não é especificado na string de formato, mas como um argumento de valor inteiro adicional precedendo o argumento que deve ser formatado.

O sub-especificador **length** modifica o comprimento do tipo de dados. Segue a tabela que mostra os tipos usados para interpretar os argumentos correspondentes com e sem especificador **length** (se um tipo diferente for usado, a promoção ou conversão de tipo apropriada é executada, se permitido):

length	**Especificadores**						
	d i	**u 0 x X**	**f F e E g G a A**	**c**	**s**	**p**	**n**
(none)	int	unsigned int	double	int		void*	int*
hh	signed char	Unsigned char					signed char*
h	short int	Unsigned short int					short 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			long double				

Observação sobre o **especificador** c: ele usa um int (ou wint_t) como argumento, mas, executa a conversão adequada para um valor char (ou um wchar_t) antes de formatá-lo para saída.

Dependendo da string **format**, a função pode esperar uma sequência de argumentos adicionais, cada um contendo um valor a ser usado para substituir um **especificador de formato** na string **format** (ou um ponteiro para um local de armazenamento, para n).

Deve haver pelo menos tantos desses argumentos quanto o número de valores especificados nos **especificadores de formato**. Argumentos adicionais são ignorados pela função.

Valor de retorno

Em caso de sucesso, o número total de caracteres escritos é retornado.

Se ocorrer um erro de escrita, o indicador de erro (**ferror**) é definido e um número negativo é retornado.

Se ocorrer um erro de codificação de caracteres multibyte durante a escrita de caracteres largos, **errno** será definido como EILSEQ e um número negativo será retornado.