

C LIBRARY FUNCTION - VSPRINTF

Description

The C library function **int vsprintf(char * str, const char * format, va_list arg)** sends formatted output to a string using an argument list passed to it.

Declaration

Following is the declaration for vsprintf function.

```
int vsprintf(char * str, const char * format, va_list arg)
```

Parameters

- **str** – This is the array of char elements where the resulting string is to be stored.
- **format** – This is the C string that contains the text to be written to the str. It can optionally contain embedded format tags that are replaced by the values specified in subsequent additional arguments and are formatted as requested. Format tags prototype – **%[flags][width][.precision][length]specifier**, as explained below –

specifier Output

c	Character
d or i	Signed decimal integer
e	Scientific notation <i>mantissa/exponent</i> using e character
E	Scientific notation <i>mantissa/exponent</i> using E character
f	Decimal floating point
g	Uses the shorter of %e or %f.
G	Uses the shorter of %E or %f
o	Signed octal
s	String of characters
u	Unsigned decimal integer
x	Unsigned hexadecimal integer
X	Unsigned hexadecimal integer <i>capitalletters</i>
p	Pointer address
n	Nothing printed
%	Character

flags Description

-	Left-justify within the given field width; Right justification is the default <i>seewidthsub-specifier</i> .
+	Forces to precede the result with a plus or minus sign +or- even for positive

numbers. By default, only negative numbers are preceded with a -ve sign.

space If no sign is going to be written, a blank space is inserted before the value.

Used with o, x or X specifiers the value is preceded with 0, 0x or 0X respectively for values different than zero. Used with e, E and f, it forces the written output to contain a decimal point even if no digits would follow. By default, if no digits follow, no decimal point is written. Used with g or G the result is the same as with e or E but trailing zeros are not removed.

0 Left-pads the number with zeroes 0 instead of spaces, where padding is specified *seewidthsub-specifier*.

width	Description
<i>number</i>	Minimum number of characters to be printed. If the value to be printed is shorter than this number, the result is padded with blank spaces. The value is not truncated even if the result is larger.
*	The width is not specified in the format string, but as an additional integer value argument preceding the argument that has to be formatted.

.precision	Description
. <i>number</i>	For integer specifiers d, i, o, u, x, X – precision specifies the minimum number of digits to be written. If the value to be written is shorter than this number, the result is padded with leading zeros. The value is not truncated even if the result is longer. A precision of 0 means that no character is written for the value 0. For e, E and f specifiers – this is the number of digits to be printed after the decimal point. For g and G specifiers – This is the maximum number of significant digits to be printed. For s – this is the maximum number of characters to be printed. By default all characters are printed until the ending null character is encountered. For c type – it has no effect. When no precision is specified, the default is 1. If the period is specified without an explicit value for precision, 0 is assumed.
.*	The precision is not specified in the format string, but as an additional integer value argument preceding the argument that has to be formatted.

length	Description
h	The argument is interpreted as a short int or unsigned short int <i>only applies to integer specifiers – i, d, o, u, x and X.</i>
l	The argument is interpreted as a long int or unsigned long int for integer specifiers i, d, o, u, x and X, and as a wide character or wide character string for specifiers c and s.
L	The argument is interpreted as a long double <i>only applies to floating point specifiers – e, E, f, g and G.</i>

- **arg** – An object representing the variable arguments list. This should be initialized by the `va_start` macro defined in `<stdarg.h>`.

Return Value

If successful, the total number of characters written is returned, otherwise a negative number is returned.

Example

The following example shows the usage of vsprintf function.

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

char buffer[80];
int vspfunc(char *format, ...)
{
    va_list aptr;
    int ret;

    va_start(aptr, format);
    ret = vsprintf(buffer, format, aptr);
    va_end(aptr);

    return(ret);
}

int main()
{
    int i = 5;
    float f = 27.0;
    char str[50] = "tutoriaspoint.com";

    vspfunc("%d %f %s", i, f, str);
    printf("%s\n", buffer);

    return(0);
}
```

Let us compile and run the above program, this will produce the following result –

```
5 27 000000 tutoriaspoint.com
Loading [MathJax]/jax/output/HTML-CSS/jax.js
```