A `printf` format reference page (cheat sheet)

By Alvin Alexander. Last updated: November 21, 2019

Summary: This page is a *printf* formatting cheat sheet. I originally created this cheat sheet for my own purposes, and then thought I would share it here.

A great thing about the `printf` formatting syntax is that the format specifiers you can use are very similar — if not identical — between different languages, including C, C++, Java, Perl, PHP, Ruby, Scala, and others. This means that your `printf` knowledge is reusable, which is a good thing.

printf formatting with Perl and Java

In this cheat sheet I'll show all the examples using Perl, but at first it might help to see one example using both Perl and Java. Therefore, here's a simple Perl `printf` example to get things started:

```
printf("the %s jumped over the %s, %d times", "cow", "moon", 2);
```

And here are three different Java `printf` examples, using different string formatting methods that are available to you in the Java programming language:

```
System.out.format("the %s jumped over the %s, %d times", "cow", "moon", 2);
System.err.format("the %s jumped over the %s, %d times", "cow", "moon", 2);
String result = String.format("the %s jumped over the %s, %d times", "cow", "moon"
```

As you can see in that last `String.format` example, that line of code doesn't print any output, while the first line prints to standard output, and the second line prints to standard error.

In the remainder of this document I'll use Perl examples, but again, the actual format specifier strings can be used in many different languages.

A summary of printf format specifiers

Here's a quick summary of the available `printf` format specifiers:

%c	character
%d	decimal (integer) number (base 10)
%e	exponential floating-point number
%f	floating-point number
%i	integer (base 10)
%o	octal number (base 8)
%s	a string of characters
%u	unsigned decimal (integer) number
%x	number in hexadecimal (base 16)
%%	print a percent sign
\%	print a percent sign

Controlling integer width with printf

The `%3d` specifier is used with integers, and means a minimum width of three spaces, which, by default, will be right-justified:

printf("%3d", 0);	`0`
printf("%3d", 123456789);	`123456789`
printf("%3d", -10);	`-10`
printf("%3d", -123456789);	`-123456789`

Left-justifying printf integer output

To left-justify integer output with `printf`, just add a minus sign (`-`) after the `%` symbol, like this:

printf("%-3d", 0);	`0`
printf("%-3d", 123456789);	`123456789`
printf("%-3d", -10);	`-10`
printf("%-3d", -123456789);	`-123456789`

The printf integer zero-fill option

To zero-fill your `printf` integer output, just add a zero (`0`) after the `%` symbol, like this:

printf("%03d", 0);	000
printf("%03d", 1);	001
printf("%03d", 123456789);	123456789
printf("%03d", -10);	-10
printf("%03d", -123456789);	-123456789

printf integer formatting

As a summary of `printf` integer formatting, here's a little collection of integer formatting examples. Several different options are shown, including a minimum width specification, left-justified, zero-filled, and also a plus sign for positive numbers.

Description	Code	Result
At least five wide	printf("'%5d'", 10);	' 10'
At least five-wide, left-justified	printf("'%-5d'", 10);	'10 '
At least five-wide, zero-filled	printf("'%05d'", 10);	'00010'
At least five-wide, with a plus sign	printf("'%+5d'", 10);	' +10'
Five-wide, plus sign, left-justified	printf("'%-+5d'", 10);	'+10 '

formatting floating point numbers with printf

Here are several examples showing how to format floating-point numbers with `printf`:

Description	Code	Result
Print one position after the decimal	printf("'%.1f'", 10.3456);	'10.3'
Two positions after the decimal	printf("'%.2f'", 10.3456);	'10.35'
Eight-wide, two positions after the decimal	printf("'%8.2f'", 10.3456);	' 10.35'
Eight-wide, four positions after the decimal	printf("'%8.4f'", 10.3456);	' 10.3456'
Eight-wide, two positions after the decimal, zero-filled	printf("'%08.2f'", 10.3456);	'00010.35'
Eight-wide, two positions after the decimal, left-justified	printf("'%-8.2f'", 10.3456);	'10.35 '
Printing a much larger number with that same format	printf("'%-8.2f'", 101234567.3456);	'101234567.35'

printf string formatting

Here are several examples that show how to format string output with `printf`:

Description	Code	Result
A simple string	printf("'%s'", "Hello");	`'Hello'`
A string with a minimum length	printf("'%10s'", "Hello");	`'     Hello'`
Minimum length, left-justified	printf("'%-10s'", "Hello");	`'Hello     '`

printf special characters

The following character sequences have a special meaning when used as `printf` format specifiers:

\a	audible alert
\b	backspace
\f	form feed
\n	newline, or linefeed
\r	carriage return
\t	tab
\v	vertical tab
\\	backslash

As you can see from that last example, because the backslash character itself is treated specially, you have to print two backslash characters in a row to get one backslash character to appear in your output.

Here are a few examples of how to use these special characters:

Description	Code	Result
Insert a tab character in a string	printf("Hello\tworld");	`Hello world`
Insert a newline character in a string	printf("Hello\nworld");	`Hello` `world`
Typical use of the newline character	printf("Hello world\n");	`Hello world`
A DOS/Windows path with backslash characters	printf("C:\\Windows\\System32\\");	`C:\Windows\System32\`