```
%%   a percent sign
%c   a character with the given number
%s   a string
%d   a signed integer, in decimal
%u   an unsigned integer, in decimal
%o   an unsigned integer, in octal
%x   an unsigned integer, in hexadecimal
%e   a floating-point number, in scientific notation
%f   a floating-point number, in fixed decimal notation
%g   a floating-point number, in %e or %f notation
```

In addition, Perl permits the following widely-supported conversions:

```
%X   like %x, but using upper-case letters
%E   like %e, but using an upper-case "E"
%G   like %g, but with an upper-case "E" (if applicable)
%b   an unsigned integer, in binary
%p   a pointer (outputs the Perl value's address in hexadecimal)
%n   special: *stores* the number of characters output so far
     into the next variable in the parameter list
```

Finally, for backward (and we do mean "backward") compatibility, Perl permits these unnecessary but widely-supported conversions:

```
%i   a synonym for %d
%D   a synonym for %ld
%U   a synonym for %lu
%O   a synonym for %lo
%F   a synonym for %f
```

Note that the number of exponent digits in the scientific notation produced by `%e`, `%E`, `%g` and `%G` for numbers with the modulus of the exponent less than 100 is system-dependent: it may be three or less (zero-padded as necessary). In other words, 1.23 times ten to the 99th may be either "1.23e99" or "1.23e099".

Between the `%` and the format letter, you may specify a number of additional attributes controlling the interpretation of the format. In order, these are:

format parameter index

An explicit format parameter index, such as `2$`. By default sprintf will format the next unused argument in the list, but this allows you to take the arguments out of order. Eg:

```
printf '%2$d %1$d', 12, 34;      # prints "34 12"
printf '%3$d %d %1$d', 1, 2, 3;  # prints "3 1 1"
```

flags

one or more of: space prefix positive number with a space + prefix positive number with a plus sign - left-justify within the field 0 use zeros, not spaces, to right-justify # prefix non-zero octal with "0", non-zero hex with "0x", non-zero binary with "0b"

For example:

```
printf '<% d>', 12;   # prints "< 12>"
printf '<%+d>', 12;   # prints "<+12>"
printf '<%6s>', 12;   # prints "<    12>"
printf '<%-6s>', 12;  # prints "<12    >"
printf '<%06s>', 12;  # prints "<000012>"
printf '<%#x>', 12;   # prints "<0xc>"
```