【C】Day4(3)

Course	Advanced C
Study Date	@April 4, 2022

【Ch7】Input and Output

7.2 Formatted Output-Printf

The output function `printf` translates internal values to characters.

```
int printf(char *format, arg1, arg2, ...)
```

`printf` converts, formats, and prints its arguments on the standard output under control of the format. It returns the number of characters printed.

The format string contains two types of objects:

- Ordinary characters, which are copied to the output stream
- Conversion specifications, each of which causes conversion and printing of the next successive argument to `printf`.

  Each conversion specification begins with a `%` and ends with a conversion character.

Between the `%` and the conversion character there may be, in order:

- A minus sign, which specifies left adjustment of the converted argument
- A number that specifies the minimum field width. The converted argument will be printed in a field at least this wide.

  If necessary it will be padded on the left(or right, if left adjustment is called for) to make up the field width.
- A period, which separates the field width from the precision.

- A number, the precision, that specifies the maximum number of characters to be printed from a string, or the number of digits after the decimal point of a floating-point value, or the minimum number of digits for an integer.
- An `h` if the integer is to be printed as a short, or `l` if as a long.

Conversion characters are shown below:

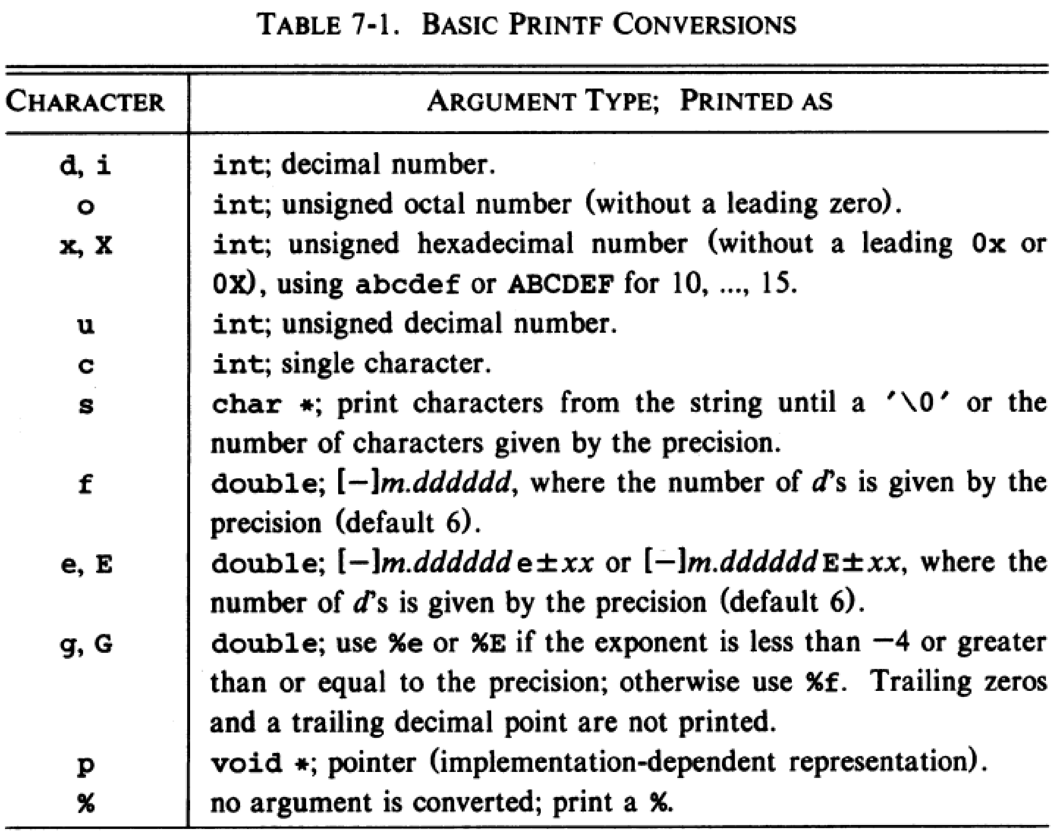


TABLE 7-1. BASIC PRINTF CONVERSIONS

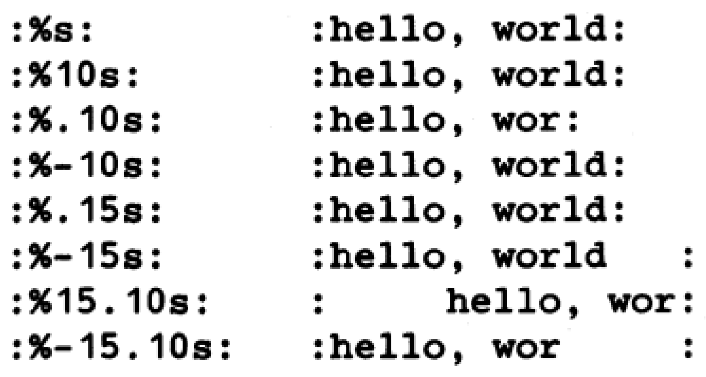
CHARACTER	ARGUMENT TYPE; PRINTED AS
d, i	int; decimal number.
o	int; unsigned octal number (without a leading zero).
x, X	int; unsigned hexadecimal number (without a leading 0x or 0X), using abcdef or ABCDEF for 10, ..., 15.
u	int; unsigned decimal number.
c	int; single character.
s	char *; print characters from the string until a '\0' or the number of characters given by the precision.
f	double; [−]*m.dddddd*, where the number of *d*'s is given by the precision (default 6).
e, E	double; [−]*m.dddddd*e±*xx* or [−]*m.dddddd*E±*xx*, where the number of *d*'s is given by the precision (default 6).
g, G	double; use %e or %E if the exponent is less than −4 or greater than or equal to the precision; otherwise use %f. Trailing zeros and a trailing decimal point are not printed.
p	void *; pointer (implementation-dependent representation).
%	no argument is converted; print a %.

A width or precision may be specified as `*`, in which case the value is computed by converting the next argument.

For example, to print at most max characters from a string s,

```
printf("%.*s", max, s);
```

The following table shows the effect of a variety of specifications in printing “hello, world”.



```
:%s:          :hello, world:
:%10s:        :hello, world:
:%.10s:       :hello, wor:
:%-10s:       :hello, world:
:%.15s:       :hello, world:
:%-15s:       :hello, world   :
:%15.10s:     :     hello, wor:
:%-15.10s:    :hello, wor     :
```

The function `sprintf` does the same conversions as `printf` does, but stores the output in a string:

```
int sprintf(char *string, char *format, arg1, arg2, ...)
```