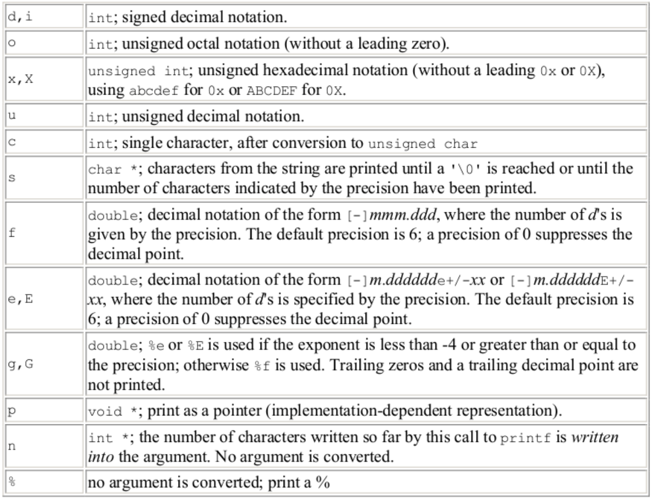
CSD101: Introduction to computing and programming (ICP)

`printf` structure

- The `printf` function call structure is below.
  `printf("<format spec.>",<arguments>)`
- The `<format spec.>` contains zero or more conversion specifiers signalled by the % sign. For each conversion specifier there should be an argument in `<arguments>` of the corresponding type.
- Till now the conversion specifiers have been simple indicators of the type. For example, `%d` for `int` , `%f` for `float` , `%c` for char etc.
- The conversion specifier can have a much more complex structure with optional elements that allow one to control the formatting. This is discussed in the following slides.

`printf` conversion char description[1]



`d,i`	`int`; signed decimal notation.
`o`	`int`; unsigned octal notation (without a leading zero).
`x,X`	`unsigned int`; unsigned hexadecimal notation (without a leading `0x` or `0X`), using `abcdef` for `0x` or `ABCDEF` for `0X`.
`u`	`int`; unsigned decimal notation.
`c`	`int`; single character, after conversion to `unsigned char`
`s`	`char *`; characters from the string are printed until a `'\0'` is reached or until the number of characters indicated by the precision have been printed.
`f`	`double`; decimal notation of the form `[-]`*mmm.ddd*, where the number of *d*'s is given by the precision. The default precision is 6; a precision of 0 suppresses the decimal point.
`e,E`	`double`; decimal notation of the form `[-]`*m.dddddd*`e+/-`*xx* or `[-]`*m.dddddd*`E+/-`*xx*, where the number of *d*'s is specified by the precision. The default precision is 6; a precision of 0 suppresses the decimal point.
`g,G`	`double`; `%e` or `%E` is used if the exponent is less than -4 or greater than or equal to the precision; otherwise `%f` is used. Trailing zeros and a trailing decimal point are not printed.
`p`	`void *`; print as a pointer (implementation-dependent representation).
`n`	`int *`; the number of characters written so far by this call to `printf` is *written into* the argument. No argument is converted.
`%`	no argument is converted; print a %

[1]From Kernighan, Ritchie

Format specification of `printf` I

The structure of the conversion specifier in the format specification is given below.

%	[<flags>]	[<Min. width>]	[<precision>]	[<size>]	<conv. char>

- Only % and `<conv. char>` are necessary. Others are optional.
- `<Min. width>` gives the minimum width in characters of the entire field. For a string (%s) it is right justifed.
- The `<flags>` are given below.

Flag	**Meaning**
$-$	Left justify
$+$	Print $+/-$ sign of numeric value
space	Print space if no sign
0	Pad with leading 0s
#	Special print spec. for float, octal, hex

The behaviour for the # flag is given below.

Format specification of `printf` II

Flag	Meaning
%#0	Adds leading 0 to octal number
%#*x* or *X*	Adds leading 0*x* or 0*X* to hex
%#*f* or *e*	Always prints decimal point
%#*g* or *G*	Prints trailing 0s and decimal point

- `<precision>` gives the number of digits after the decimal point for `float` values. If present it should be preceded by a decimal point. For string values printed with %s only that many characters will be printed
- The `<size>` modifier behaviour is given below.

Flag	Conv. char	Meaning
l	d,i,o,u,x	`long int`
h	d,i,o,u,x	`short int`
l	e,f	`double`
L	e,f	`long double`