Format Specifiers in C

There is a loooong list of codes for specifying output/input format in C! It can get very overwhelming.

We should try to come up with a system of rules to organize our thoughts....

① Step 1: Start with the variable type!

char →	%c	(actually char array!)
string →	%s	
int →	%d	(signed integer)
unsigned int →	%u	
float →	%f	(decimal)
float →	%e	(scientific with e)
float →	%E	(scientific with E)
float →	%g	(e, if f does not work)
float →	%G	(E, if f does not work)
(signed) short (int) →	%hi	
unsigned short (int) →	%hu	
(signed) long (int) →	%l or %ld or %li	
unsigned long (int) →	%lu	
(signed) long long (int) →	%lli or %lld	
unsigned long long (int) →	%llu	
double →	%lf	
long double →	%Lf	
pointer →	%p	
octal →	%o	
hexidecimal →	%x or %X	
"nothing" →	%n	
% character →	%%	

Step 2: Think about

(i) field width

(ii) field precision (i.e decimal places)

(iii) field alignment

Field width: a number, immediately after the % sign, specifies minimum field width.

If less than this → fill with spaces

```
int i = 45, j = 145, k = 2145;
printf("%3d\n", i);
printf("%3d\n", j);
printf("%3d\n", k);
```

```
␣45
145
2145
```

Field Precision: A period (.) is used to separate field width and field precision.

Floats and Doubles ⇨ If there is no "." provided, the precision is assumed to be six figures.

I.E. "%9f" ≡ "%9.6f"

⇨ The field width includes the decimal point.

I.E.

```
float x = 12.345678;
printf("%9.6f\n", x);
float y = 12.3456789;
printf("%9.6f\n", y);
float z = 12.34567;
printf("%9.6f\n", z);
```

```
12.345678
12.345679
12.345670
```

```
float t = 123.45678;
printf("%9.6f\n", t);
```

(Expect 123.456780

```
123.456779
```

↑ Egads!

exceeds

What happened ?? 123.45678

already the precision of float! Great example of why we should always use double !!

Field Alignment :

A minus sign indicates left alignment.

```
float x = 1.23;
printf("%9.2f|\n", x);
printf("%-9.2f|\n", x);
```

```
 1 2 3 4 5 6 7 8 9
 ␣␣␣␣␣1.23|
 1.23␣␣␣␣␣|
```

: A plus sign indicates that we should explicitly print a + sign for positive numbers... the + sign is not incuded in the field width count.

```
float x = 12.345678;
float y = -12.345678;
printf("%9.6f\n", x);
printf("%9.6f\n", y);
```

```
12.345678
-12.345678
```

```
printf("%+9.6f\n", x);
printf("%+9.6f\n", y);
```

```
+12.345678
-12.345678
```

} nicely aligned !!

```
printf("%10.6f\n", x);
printf("%10.6f\n", y);
```

```
␣12.345678
-12.345678
```

} also, nicely aligned !!

: Adding a "0" in front of the field width

specifier will pad the
number with zeroes
in front.

```
float xpad = 1.234567;
printf("%012.6f\n", xpad
```

```
| 00001.234567
  1 2 3 4 5 6 7 8 9 10 11 12
```

```
printf("%-012.6f\n", xpad)
```

```
| 1.234567␣␣␣␣
```

i.e "0" is ignored when - is present!

Weird inconsistency !! ⇨ in
padding negative numbers with zeros,
the minus sign is included in the
field count !!

```
int ip = 713;
int in = -713;
printf("%08d\n", ip);
printf("%08d\n", in);
```

```
  1 2 3 4 5 6 7 8
| 00000713
| -0000713
```