In the past, we've assumed that the minimum field width and precision were constants embedded in the format string. Putting the * character where either number would normally go allows us to specify it as an argument *after* the format string. For example, the following calls of `printf` all produce the same output:

```
printf("%6.4d", i);
printf("%*.4d", 6, i);
printf("%6.*d", 4, i);
printf("%*.*d", 6, 4, i);
```

Notice that the values to be filled in for the * come just before the value to be displayed. A major advantage of *, by the way, is that it allows us to use a macro to specify the width or precision:

```
printf("%*d", WIDTH, i);
```

We can even compute the width or precision during program execution:

```
printf("%*d", page_width / num_cols, i);
```

The most unusual specifications are `%p` and `%n`. The `%p` conversion allows us to print the value of a pointer:

```
printf("%p", (void *) ptr);   /* displays value of ptr */
```

Although `%p` is occasionally useful during debugging, it's not a feature that most programmers use on a daily basis. The C standard doesn't specify what a pointer looks like when printed using `%p`, but it's likely to be shown as an octal or hexadecimal number.

The `%n` conversion is used to find out how many characters have been printed so far by a call of ...`printf`. For example, after the call

```
printf("%d%n\n", 123, &len);
```

the value of `len` will be 3, since `printf` had written 3 characters (`123`) by the time it reached `%n`. Notice that `&` must precede `len` (because `%n` requires a pointer) and that `len` itself isn't printed.

The ...scanf Functions

```
int fscanf(FILE * restrict stream,
           const char * restrict format, ...);
int scanf(const char * restrict format, ...);
```

fscanf
scanf

`fscanf` and `scanf` read data items from an input stream, using a format string to indicate the layout of the input. After the format string, any number of pointers—each pointing to an object—follow as additional arguments. Input items are converted (according to conversion specifications in the format string) and stored in these objects.