3 Formatted Input/Output

In seeking the unattainable, simplicity only gets in the way.

`scanf` and `printf`, which support formatted reading and writing, are two of the most frequently used functions in C. As this chapter shows, both are powerful but tricky to use properly. Section 3.1 describes `printf`, and Section 3.2 covers `scanf`. Neither section gives complete details, which will have to wait until Chapter 22.

3.1 The `printf` Function

The `printf` function is designed to display the contents of a string, known as the *format string*, with values possibly inserted at specified points in the string. When it's called, `printf` must be supplied with the format string, followed by any values that are to be inserted into the string during printing:

`printf(`*string*`,` *expr*$_1$`,` *expr*$_2$`, ...);`

The values displayed can be constants, variables, or more complicated expressions. There's no limit on the number of values that can be printed by a single call of `printf`.

The format string may contain both ordinary characters and *conversion specifications*, which begin with the `%` character. A conversion specification is a placeholder representing a value to be filled in during printing. The information that follows the `%` character *specifies* how the value is *converted* from its internal form (binary) to printed form (characters)—that's where the term "conversion specification" comes from. For example, the conversion specification `%d` specifies that `printf` is to convert an `int` value from binary to a string of decimal digits, while `%f` does the same for a `float` value.