with three digits after the decimal point. Since x requires only seven characters (three before the decimal point, three after the decimal point, and one for the decimal point itself), three spaces precede x.

- %10.3e — Displays x in exponential form, using 10 characters overall, with three digits after the decimal point. x requires nine characters altogether (including the exponent), so one space precedes x.
- %-10g — Displays x in either fixed decimal form or exponential form, using 10 characters overall. In this case, printf chose to display x in fixed decimal form. The presence of the minus sign forces left justification, so x is followed by four spaces.

Escape Sequences

The \n code that we often use in format strings is called an ***escape sequence***. Escape sequences enable strings to contain characters that would otherwise cause problems for the compiler, including nonprinting (control) characters and characters that have a special meaning to the compiler (such as "). We'll provide a complete list of escape sequences later; for now, here's a sample:

escape sequences ➤7.3

Alert (bell)	\a
Backspace	\b
New line	\n
Horizontal tab	\t

When they appear in printf format strings, these escape sequences represent actions to perform upon printing. Printing \a causes an audible beep on most machines. Printing \b moves the cursor back one position. Printing \n advances the cursor to the beginning of the next line. Printing \t moves the cursor to the next tab stop.

Q&A

A string may contain any number of escape sequences. Consider the following printf example, in which the format string contains six escape sequences:

```
printf("Item\tUnit\tPurchase\n\tPrice\tDate\n");
```

Executing this statement prints a two-line heading:

```
Item    Unit    Purchase
        Price   Date
```

Another common escape sequence is \". which represents the " character. Since the " character marks the beginning and end of a string, it can`t appear within a string without the use of this escape sequence. Here's an example:

```
printf("\"Hello!\"");
```

This statement produces the following output:

```
"Hello!"
```