allows a character constant to contain the ' character, while the \ " escape allows a string to contain the " character. The \? escape is rarely used.

Character escapes are handy, but they have a problem: the list of character escapes doesn't include all the nonprinting ASCII characters, just the most common. Character escapes are also useless for representing characters beyond the basic 128 ASCII characters. Numeric escapes, which can represent *any* character, are the solution to this problem.

To write a numeric escape for a particular character, first look up the character's octal or hexadecimal value in a table like the one in Appendix E. For example, the ASCII escape character (decimal value: 27) has the value 33 in octal and 1B in hex. Either of these codes can be used to write an escape sequence:

■ An *octal escape sequence* consists of the \ character followed by an octal number with at most three digits. (This number must be representable as an unsigned character, so its maximum value is normally 377 octal.) For example, the escape character could be written \33 or \033. Octal numbers in escape sequences—unlike octal constants—don't have to begin with 0.

■ A *hexadecimal escape sequence* consists of \x followed by a hexadecimal number. Although C places no limit on the number of digits in the hexadecimal number, it must be representable as an unsigned character (hence it can't exceed FF if characters are eight bits long). Using this notation, the escape character would be written \x1b or \x1B. The x must be in lower case, but the hex digits (such as b) can be upper or lower case.

When used as a character constant, an escape sequence must be enclosed in single quotes. For example, a constant representing the escape character would be written '\33' (or '\x1b'). Escape sequences tend to get a bit cryptic, so it's often a good idea to give them names using #define:

```
#define ESC '\33'    /* ASCII escape character */
```

Escape sequences can be embedded in strings as well, as we saw in Section 3.1.

Escape sequences aren't the only special notations for representing characters. Trigraph sequences provide a way to represent the characters #, [, \,], ^, {, |, }, and ~, which may not be available on keyboards in some countries. C99 adds universal character names, which resemble escape sequences. Unlike escape sequences, however, universal character names are allowed in identifiers.

Character-Handling Functions

Earlier in this section, we saw how to write an if statement that converts a lower-case letter to upper-case:

```
if ('a' <= ch && ch <= 'z')
  ch = ch - 'a' + 'A';
```

This isn't the best method, though. A faster—and more portable—way to convert case is to call C's toupper library function: