as its left operand. An lvalue (pronounced "L-value") represents an object stored in computer memory, not a constant or the result of a computation. Variables are lvalues; expressions such as `10` or `2 * i` are not. At this point, variables are the only lvalues that we know about; other kinds of lvalues will appear in later chapters.

Since the assignment operator requires an lvalue as its left operand, it's illegal to put any other kind of expression on the left side of an assignment expression:

```
12 = i;       /*** WRONG ***/
i + j = 0;    /*** WRONG ***/
-i = j;       /*** WRONG ***/
```

The compiler will detect errors of this nature, and you'll get an error message such as *"invalid lvalue in assignment."*

Compound Assignment

Assignments that use the old value of a variable to compute its new value are common in C programs. The following statement, for example, adds 2 to the value stored in `i`:

```
i = i + 2;
```

C's *compound assignment* operators allow us to shorten this statement and others like it. Using the += operator, we simply write:

```
i += 2;    /* same as i = i + 2; */
```

The += operator adds the value of the right operand to the variable on the left.

There are nine other compound assignment operators, including the following:

```
-=   *=   /=   %=
```

other assignment operators ➤ *20.1*

(We'll cover the remaining compound assignment operators in a later chapter.) All compound assignment operators work in much the same way:

v += *e* adds *v* to *e*, storing the result in *v*
v -= *e* subtracts *e* from *v*, storing the result in *v*
v *= *e* multiplies *v* by *e*, storing the result in *v*
v /= *e* divides *v* by *e*, storing the result in *v*
v %= *e* computes the remainder when *v* is divided by *e*, storing the result in *v*

Note that I've been careful not to say that *v* += *e* is "equivalent" to *v* = *v* + *e*. One problem is operator precedence: `i *= j + k` isn't the same as `i = i * j + k`. There are also rare cases in which *v* += *e* differs from *v* = *v* + *e* because *v* itself has a side effect. Similar remarks apply to the other compound assignment operators.

Q&A

When using the compound assignment operators, be careful not to switch the two characters that make up the operator. Switching the characters may yield an expression that is acceptable to the compiler but that doesn't have the intended meaning. For example, if you meant to write `i += j` but typed `i =+ j` instead, the