- *Counting up from* **0** *to* **n–1***:*

idiom

```
for (i = 0; i < n; i++) ...
```

- *Counting up from* **1** *to* **n***:*

idiom

```
for (i = 1; i <= n; i++) ...
```

- *Counting down from* **n–1** *to* **0***:*

idiom

```
for (i = n - 1; i >= 0; i--) ...
```

- *Counting down from* **n** *to* **1***:*

idiom

```
for (i = n; i > 0; i--) ...
```

Imitating these patterns will help you avoid some of the following errors, which beginning C programmers often make:

- Using < instead of > (or vice versa) in the controlling expression. Notice that "counting up" loops use the < or <= operator, while "counting down" loops rely on > or >=.
- Using == in the controlling expression instead of <, <=, >, or >=. A controlling expression needs to be true at the beginning of the loop, then later become false so that the loop can terminate. A test such as `i == n` doesn't make much sense, because it won't be true initially.
- "Off-by-one" errors such as writing the controlling expression as `i <= n` instead of `i < n`.

Omitting Expressions in a `for` Statement

The `for` statement is even more flexible than we've seen so far. Some `for` loops may not need all three of the expressions that normally control the loop, so C allows us to omit any or all of the expressions.

If the *first* expression is omitted, no initialization is performed before the loop is executed:

```
i = 10;
for (; i > 0; --i)
  printf("T minus %d and counting\n", i);
```

In this example, `i` has been initialized by a separate assignment, so we've omitted the first expression in the `for` statement. (Notice that the semicolon between the first and second expressions remains. The two semicolons must always be present, even when we've omitted some of the expressions.)

If we omit the *third* expression in a `for` statement, the loop body is responsible for ensuring that the value of the second expression eventually becomes false. Our `for` statement example could be written like this:

```
for (i = 10; i > 0;)
  printf("T minus %d and counting\n", i--);
```