

If we move the `n % d == 0` condition into the loop's controlling expression, the body of the loop becomes empty:

```
for (d = 2; d < n && n % d != 0; d++)
    /* empty loop body */ ;
```

Each time through the loop, the condition `d < n` is tested first; if it's false, the loop terminates. Otherwise, the condition `n % d != 0` is tested, and if that's false, the loop terminates. (In the latter case, `n % d == 0` must be true: in other words, we've found a divisor of `n`.)

Note how we've put the null statement on a line by itself, instead of writing

```
for (d = 2; d < n && n % d != 0; d++);
```

Q&A

C programmers customarily put the null statement on a line by itself. Otherwise, someone reading the program might get confused about whether the statement after the `for` was actually its body:

```
for (d = 2; d < n && n % d != 0; d++);
if (d < n)
    printf("%d is divisible by %d\n", n, d);
```

Converting an ordinary loop into one with an empty body doesn't buy much: the new loop is often more concise but usually no more efficient. In a few cases, though, a loop with an empty body is clearly superior to the alternatives. For example, we'll find these loops to be handy for reading character data.

reading characters ► 7.3



Accidentally putting a semicolon after the parentheses in an `if`, `while`, or `for` statement creates a null statement, thus ending the `if`, `while`, or `for` prematurely.

- In an `if` statement, putting a semicolon after the parentheses creates an `if` statement that apparently performs the same action regardless of the value of its controlling expression:

```
if (d == 0);                                /**** WRONG ****/
    printf("Error: Division by zero\n");
```

The call of `printf` isn't inside the `if` statement, so it's performed regardless of whether `d` is equal to 0.

- In a `while` statement, putting a semicolon after the parentheses may create an infinite loop:

```
i = 10;
while (i > 0);                                /**** WRONG ****/
{
    printf("T minus %d and counting\n", i);
    --i;
}
```