- Don't forget to declare the variable used in `for`'s parentheses. This common mistake is another one made by just about everyone. Refer to Chapter 8 for more information about declaring variables.
- Here's a handy plug-in you can use for loops. Just substitute the big *X* in the following line for the number of times you want the loop to work:

```
for(i=1 ; i<=X ; i=i+1)
```

  You must declare `i` to be an integer variable. It starts out equal to `1` and ends up equal to the value of *X*. To repeat a loop 100 times, for example, you use this command:

```
for(i=1 ; i<=100 ; i=i+1)
```

Having fun whilst counting to 100

This section has the source code for a program named 100.C. This program uses a `for` loop to count to 100 and display each number on the screen. Indeed, it's a major achievement: Early computers could count up to only 50 before they began making wild and often inaccurate guesses about what number came next. (Refer to your phone bill to see what I mean.)

The core of 100.C is similar to OUCH.C. In fact, the only reason I have tossed it in here is that `for` loops are so odd to some folks that you need two program examples to drive home the point:

```
#include <stdio.h>

int main()
{
    int i;

    for(i=1 ; i<=100 ; i=i+1)
        printf("%d\t",i);
    return(0);
}
```

Type this source code into your editor. Watch your indentations; it's traditional in the C language to indent one statement belonging to another (as shown in the example), even when the curly braces are omitted.

In the `for` statement, the `i` variable starts out equal to 1. The `while_true` condition is `i<=100` — which means that the loop works, although the value of variable `i` is less than or equal to 100. The final part of the statement increments the value of `i` by 1 each time the loop works.