numdigits.c

```
/* Calculates the number of digits in an integer */

#include <stdio.h>

int main(void)
{
  int digits = 0, n;

  printf("Enter a nonnegative integer: ");
  scanf("%d", &n);

  do {
    n /= 10;
    digits++;
  } while (n > 0);

  printf("The number has %d digit(s).\n", digits);

  return 0;
}
```

To see why the do statement is the right choice, let's see what would happen if we were to replace the do loop by a similar `while` loop:

```
while (n > 0) {
  n /= 10;
  digits++;
}
```

If `n` is 0 initially, this loop won't execute at all, and the program would print

```
The number has 0 digit(s).
```

6.3 The `for` Statement

We now come to the last of C's loops: the `for` statement. Don't be discouraged by the `for` statement's apparent complexity; it's actually the best way to write many loops. The `for` statement is ideal for loops that have a "counting" variable, but it's versatile enough to be used for other kinds of loops as well.

The `for` statement has the form

`for` statement

```
for ( expr1 ; expr2 ; expr3 ) statement
```

where *expr1*, *expr2*, and *expr3* are expressions. Here's an example:

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

When this `for` statement is executed, the variable `i` is initialized to 10, then `i` is tested to see if it's greater than 0. Since it is, the message `T minus 10 and`