```
This program sums a series of integers.
Enter integers (0 to terminate): 8 23 71 5 0
The sum is: 107
```

Clearly we'll need a loop that uses `scanf` to read a number and then adds the number to a running total.

Letting `n` represent the number just read and `sum` the total of all numbers previously read, we end up with the following program:

*sum.c*

```
/* Sums a series of numbers */

#include <stdio.h>

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

  printf("This program sums a series of integers.\n");
  printf("Enter integers (0 to terminate): ");

  scanf("%d", &n);
  while (n != 0) {
    sum += n;
    scanf("%d", &n);
  }
  printf("The sum is: %d\n", sum);

  return 0;
}
```

Notice that the condition `n != 0` is tested just after a number is read, allowing the loop to terminate as soon as possible. Also note that there are two identical calls of `scanf`, which is often hard to avoid when using `while` loops.

## 6.2 The `do` Statement

The `do` statement is closely related to the `while` statement; in fact, the `do` statement is essentially just a `while` statement whose controlling expression is tested *after* each execution of the loop body. The `do` statement has the form

**do statement**

```
do statement while ( expression ) ;
```

As with the `while` statement, the body of a `do` statement must be one statement (possibly compound, of course) and the controlling expression must be enclosed within parentheses.

When a `do` statement is executed, the loop body is executed first, then the controlling expression is evaluated. If the value of the expression is nonzero, the loop