Here is the program output:

Total cookies eaten: 255

As you can see, the program works. Now let's look at why it works.

**How Pointers Enable Array-Processing Functions**

The key is that C++, like C, in most contexts treats the name of an array as if it were a pointer. Recall from Chapter 4, "Compound Types," that C++ interprets an array name as the address of its first element:

```
cookies == &cookies[0]  // array name is address of first element
```

(There are two exceptions to this rule. First, the array declaration uses the array name to label the storage. Second, applying `sizeof` to an array name yields the size of the whole array, in bytes.)

Listing 7.5 makes the following function call:

```
int sum = sum_arr(cookies, ArSize);
```

Here `cookies` is the name of an array, hence by C++ rules `cookies` is the address of its first element. The function passes an address. Because the array has type `int` elements, `cookies` must be type `pointer-to-int`, or `int *`. That suggests that the correct function heading should be this:

```
int sum_arr(int * arr, int n) // arr = array name, n = size
```

Here `int *arr` has replaced `int arr[]`. It turns out that both headings are correct, for in C++