Remember

To indicate the kind of array and the number of elements to an array-processing function, pass the information as two separate arguments:

```c
void fillArray(int arr[], int size);  // prototype
```

Don't try to pass the array size by using brackets notation:

```c
void fillArray(int arr[size]);         // NO -- bad prototype
```

More Array Function Examples

When you choose to use an array to represent data, you are making a design decision. But design decisions should go beyond how data is stored; they also should involve how the data is used. Often, you'll find it profitable to write specific functions to handle specific data operations. (The profits here are increased program reliability, ease of modification, and ease of debugging.) Also, when you begin integrating storage properties with operations when you think about a program, you are taking an important step toward the OOP mind-set; that, too, might prove profitable in the future.

Let's examine a simple case. Suppose you want to use an array to keep track of the dollar values of your real estate. (If necessary, suppose you have real estate.) You have to decide what type to use. Certainly, `double` is less restrictive in its range than `int` or `long`, and it provides enough significant digits to represent the values precisely. Next, you have to decide on the number of array elements. (With dynamic arrays created with `new`, you can put off that decision, but let's keep things simple.) Let's say that you have no more than five properties, so you can use an array of five `doubles`.

Now consider the possible operations you might want to execute with the real estate array. Two very basic ones are reading values into the array and displaying the array contents. Let's add one more operation to the list: reassessing the value of the properties. For simplicity, assume that all your properties increase or decrease in value at the same rate. (Remember, this is a book on C++, not on real estate management.) Next, fit a function to each operation and then write the code accordingly. We go through these steps next.