Filling the Array

Because a function with an array name argument accesses the original array, not a copy, you can use a function call to assign values to array elements. One argument to the function will be the name of the array to be filled. In general, a program might manage more than one person's investments, hence more than one array, so you won't want to build the array size into the function. Instead, pass the array size as a second argument, as in the previous example. Also, it's possible that you might want to quit reading data before filling the array, so you want to build that feature into the function. Because you might enter fewer than the maximum number of elements, it makes sense to have the function return the actual number of values entered. These considerations suggest the following function prototype:

```c
int fill_array(double ar[], int limit);
```

The function takes an array name argument and an argument specifying the maximum number of items to be read, and the function returns the actual number of items read. For example, if you use this function with an array of five elements, you pass 5 as the second argument. If you enter only three values, the function returns 3.

You can use a loop to read successive values into the array, but how can you terminate the loop early? One way is to use a special value to indicate the end of input. Because no property should have a negative value, you can use a negative number to indicate the end of input. Also, the function should do something about bad input, such as terminating further input. Given this, you can code the function as follows:

```c
int fill_array(double ar[], int limit)
{
    double temp;
    int i;
    for (i = 0; i < limit; i++)
    {
        cout << "Enter value #" << (i + 1) << ": ";
        cin >> temp;
        if (!cin)  // bad input
        {
            cin.clear();
            while (cin.get() != '\n')
```