{  
    for (int i = 0; i < n; i++)  
        cout << "Cheers! ";  
        cout << "\n";  
}

double cube(double x)  
{  
    return x * x * x;  
}

Here's a sample run:

Cheers! Cheers! Cheers! Cheers! Cheers!
Give me a number: 5
A 5-foot cube has a volume of 125 cubic feet.
Cheers! Cheers! Cheers! Cheers! Cheers! Cheers! Cheers! Cheers!

Note that main() calls the type void function cheers() by using the function name and arguments followed by a semicolon: cheers(5);. That's an example of a function call statement. But because cube() has a return value, main() can use it as part of an assignment statement:

double volume = cube(side);

But we said you should concentrate on the prototypes. What should you know about prototypes? First, you should understand why C++ requires prototypes. Then, because C++ requires prototypes, you should know the proper syntax. Finally, you should appreciate what the prototype does for you. Let's look at these points in turn, using Listing 7.2 as a basis for discussion.

**Why Prototypes?**

The prototype describes the function interface to the compiler. That is, it tells the compiler what type of return value, if any, the function has, and it tells the compiler the number and type of function arguments. Consider, for example, how a prototype affects this function call from Listing 7.2: