- **Functions that produce only output:** These functions generate a value or string, returning it to the program — for example, a function that may tell you where the *Enterprise* is in the Klingon Empire. You call the `whereEnt()` function, and it returns some galactic coordinates.

Any function can fall into any category. It all depends on what you want the function to do. After you know that, you build the function accordingly.

How to send a value to a function

Sending a value to a function is as easy as heaving Grandma through a plate glass window. Just follow these steps:

1. **Know what kind of value you're going to send to the function.**

   It can be a constant value, a number or string, or it can be a C language variable. Either way, you must declare that value as the proper type so that the function knows exactly what type of value it's receiving: int, char, or float, for example.

2. **Declare the value as a variable in the function's parentheses.**

   Suppose that your function eats an integer value. If so, you need to declare that value as a variable that the function will use. It works like declaring any variable in the C language, though the declaration is made inside the function's parentheses following the function's name:

   ```
   void jerk(int repeat)
   ```

   

   Don't follow the variable declaration with a semicolon! In the preceding example, the integer variable `repeat` is declared, which means that the `jerk()` function requires an integer value. Internally, the function refers to the value by using the `repeat` variable.

3. **Somehow use the value in your function.**

   The compiler doesn't like it when you declare a variable and then that variable isn't used. (It's a waste of memory.) The error message reads something like `jerk is passed a value that is not used` or `Parameter 'repeat' is never used in function jerk`. It's a warning error — and, heck, it may not even show up — but it's a good point to make: Use your variables!

4. **Properly prototype the function.**

   You must do this or else you get a host of warning errors. My advice: Select the line that starts your function; mark it as a block. Then, copy it to up above the `main()` function. After pasting it in, add a semicolon:

   ```
   void jerk(int repeat);
   ```

   No sweat.