• If the data object is an array, use your only choice, a pointer.

• If the data object is a structure, use a reference or a pointer.

• If the data object is a class object, use a reference.

Of course, these are just guidelines, and there might be reasons for making different choices. For example, `cin` uses references for basic types so that you can use `cin >> n` instead of `cin >> &n`.

**Default Arguments**

Let's look at another topic from C++’s bag of new tricks—the default argument. A default argument is a value that's used automatically if you omit the corresponding actual argument from a function call. For example, if you set up `void wow(int n)` so that `n` has a default value of 1, then the function call `wow()` is the same as `wow(1)`. This gives you greater flexibility in how you use a function. Suppose you have a function called `left()` that returns the first `n` characters of a string, with the string and `n` as arguments. More precisely, the function returns a pointer to a new string consisting of the selected portion of the original string. For example, the call `left("theory", 3)` constructs a new string "the" and returns a pointer to it. Now suppose you establish a default value of 1 for the second argument. The call `left("theory", 3)` would work as before, with your choice of 3 overriding the default. But the call `left("theory")`, instead of being an error, would assume a second argument of 1 and return a pointer to the string "t". This kind of default is helpful if your program often needs to extract a one-character string but occasionally needs to extract longer strings.

How do you establish a default value? You must use the function prototype. Because the compiler looks at the prototype to see how many arguments a function uses, the function prototype also has to alert the program to the possibility of default arguments. The method is to assign a value to the argument in the prototype. For example, here's the prototype fitting this description of `left()`:

```
char * left(const char * str, int n = 1);
```

We want the function to return a new string, so its type is `char*`, or pointer-to-char. We want to leave the original string unaltered, so we use the `const` qualifier for the first