use the keyword `typename` instead of `class`. Also, you must use the angle brackets. The type name (Any in the example) is your choice, as long as you follow the usual C++ naming rules; many programmers use simple names like T. The rest of the code describes the algorithm for swapping two values of type Any. The template does not create any functions. Instead, it provides the compiler with directions about how to define a function. If you want a function to swap ints, then the compiler creates a function following the template pattern, substituting int for Any. Similarly, if you need a function to swap doubles, the compiler follows the template, substituting the double type for Any.

The keyword `typename` is a recent addition to C++. You can use instead of the keyword `class` in this particular context. That is, you can write the template definition this way:

```cpp
template <typename Any>
void Swap(Any &a, Any &b)
{
    Any temp;
    temp = a;
    a = b;
    b = temp;
}
```

The `typename` keyword makes it a bit more obvious that the parameter Any represents a type; however, large libraries of code already have been developed by using the older keyword `class`. The C++ Standard treats the two keywords identically when they are used in this context.

**Tip**

Use templates if you need functions that apply the same algorithm to a variety of types. If you aren't concerned with backward compatibility and can put up with the effort of typing a longer word, use the keyword `typename` rather than `class` when you declare type parameters.

To let the compiler know that you need a particular form of swap function, just use a function called `Swap()` in your program. The compiler checks the argument types you use and then generates the corresponding function. Listing 8.9 shows how this works. The