What's a Type?

Let's think a little more about what constitutes a type. For example, what is a nerd? If you subscribe to the popular stereotype, you may think of a nerd in visual terms—thick, black-rimmed glasses, pocket protector full of pens, and so on. After a little reflection, you might conclude that a nerd is better defined operationally, for example, in how he or she responds to an awkward social situation. We have a similar situation, if you don't mind stretched analogies, with a procedural language like C. At first, you tend to think of a data type in terms of its appearance—how it is stored in memory. A char, for example, is one byte of memory, and a double often is eight bytes of memory. But a little reflection leads us to conclude that a data type also is defined in terms of the operations that can be performed upon it. For example, the int type can use all the arithmetic operations. You can add, subtract, multiply, and divide integers. You also can use the modulus operator (%) with them.

On the other hand, consider pointers. A pointer might very well require the same amount of memory as an int. It might even be represented internally as an integer. But a pointer doesn't allow the same operations that an integer does. You can't, for example, multiply two pointers by each other. The concept makes no sense, so C++ doesn't implement it. Thus, when you declare a variable as an int or as a pointer-to-float, you're not just allocating memory—you also are establishing which operations can be performed with the variable. In short, specifying a basic type does two things:

- It determines how much memory is needed for a data object.
- It determines what operations, or methods, can be performed using the data object.

For built-in types, this information is built in to the compiler. But when you define a user-defined type in C++, you have to provide the same kind of information yourself. In exchange for this extra work, you gain the power and flexibility to custom fit new data types to match real-world requirements.

The Class

The class is the C++ vehicle for translating an abstraction to a user-defined type. It combines data representation and methods for manipulating that data into one neat package. Let's look at a class that represents stocks.