More recently, C++ has introduced a second way of defining a constant within a class—using the keyword `static`:

```cpp
class Stock
{
private:
    static const int Len = 30; // declare a constant! WORKS
    char company[Len];

    ...  
```

This creates a single constant called `Len` that is stored with other static variables rather than in an object. Thus, there is only one `Len` constant shared by all `Stock` objects.

Chapter 12, "Classes and Dynamic Memory Allocation," looks further into static class members. You only can use this technique for declaring static constants with integral and enumeration values. You can't store a `double` constant this way.

**An Abstract Data Type**

The `Stock` class is pretty specific. Often, however, programmers define classes to represent more general concepts. For example, classes are a good way to implement what computer scientists describe as *abstract data types*, or ADTs, for short. As the name suggests, an ADT describes a data type in a general fashion, without bringing in language or implementation details. Consider, for example, the stack. The stack is a way of storing data in which data is always added to or deleted from the top of the stack. C++ programs, for example, use a stack to manage automatic variables. As new automatic variables are generated, they are added to the top of the stack. When they expire, they are removed from a stack.

Let's describe the properties of a stack in a general, abstract way. First, a stack holds several items. (That property makes it a *container*, an even more general abstraction.) Next, a stack is characterized by the operations you can perform on one.

- You can create an empty stack.
- You can add an item to the top of a stack (*push* an item).
- You can remove an item from the top (*pop* an item).