- An ***abstract object*** is a collection of functions that operate on a hidden data structure. (In this chapter, the term "object" has a different meaning than in the rest of the book. In C terminology, an object is simply a block of memory that can store a value. In this chapter, however, an object is a collection of data bundled with operations on the data. If the data is hidden, the object is "abstract.") The stack module we've been discussing belongs to this category.
- An ***abstract data type (ADT)*** is a type whose representation is hidden. Client modules can use the type to declare variables, but have no knowledge of the structure of those variables. For a client module to perform an operation on such a variable, it must call a function provided by the abstract data type module. Abstract data types play a significant role in modern programming; we'll return to them in Sections 19.3–19.5.

19.2 Information Hiding

A well-designed module often keeps some information secret from its clients. Clients of our stack module, for example, have no need to know whether the stack is stored in an array, in a linked list, or in some other form. Deliberately concealing information from the clients of a module is known as ***information hiding***. Information hiding has two primary advantages:

- ***Security.*** If clients don't know how the stack is stored, they won't be able to corrupt it by tampering with its internal workings. To perform operations on the stack, they'll have to call functions that are provided by the module itself—functions that we've written and tested.
- ***Flexibility.*** Making changes—no matter how large—to a module's internal workings won't be difficult. For example, we could implement the stack as an array at first, then later switch to a linked list or other representation. We'll have to rewrite the implementation of the module, of course, but—if the module was designed properly—we won't have to alter the module's interface.

`static` storage class ➤ *18.2*

In C, the major tool for enforcing information hiding is the `static` storage class. Declaring a variable with file scope to be `static` gives it internal linkage, thus preventing it from being accessed from other files, including clients of the module. (Declaring a function to be `static` is also useful—the function can be directly called only by other functions in the same file.)

A Stack Module

To see the benefits of information hiding, let's look at two implementations of a stack module, one using an array and the other a linked list. The module's header file will have the following appearance: