Summary

The array, the structure, and the pointer are three C++ compound types. The array can hold several values all of the same type in a single data object. By using an index, or subscript, you can access the individual elements in an array.

The structure can hold several values of different types in a single data object, and you can use the membership operator (.) to access individual members. The first step in using structures is creating a structure template defining what members the structure holds. The name, or tag, for this template then becomes a new type identifier. You then can declare structure variables of that type.

A union can hold a single value, but it can be of a variety of types, with the member name indicating which mode is being used.

Pointers are variables designed to hold addresses. We say a pointer points to the address it holds. The pointer declaration always states to what type of object a pointer points. Applying the dereferencing operator (*) to a pointer yields the value at the location to which the pointer points.

A string is a series of characters terminated by a null character. A string can be represented by a quoted string constant, in which case the null character is implicitly understood. You can store a string in an array of char, and you can represent a string by a pointer-to-char that is initialized to point to the string. The strlen() function returns the length of a string, not counting the null character. The strcpy() function copies a string from one location to another. When using these functions, include the cstring or the string.h header file.

The new operator lets you request memory for a data object while a program is running. The operator returns the address of the memory it obtains, and you can assign that address to a pointer. The only means to access that memory is to use the pointer. If the data object is a simple variable, you can use the dereferencing operator (*) to indicate a value. If the data object is an array, you can use the pointer as if it were an array name to access the elements. If the data object is a structure, you can use the pointer dereferencing operator (->) to access structure members.

Pointers and arrays are closely connected. If ar is an array name, then the expression ar[i] is interpreted as *(ar + i), with the array name interpreted as the address of the first