**Chapter 7 Inserts**

**Insert 410-A**

**Using Structured Binding Declarations with Arrays**

In some situations, you might find it necessary to retrieve the individual values that are stored in an array and assign those values to regular variables. In C++ 17 a new feature known as *structured* *binding* *declarations* allows you to write one statement that defines a set of variables and initializes those variables with the values that are stored in an array. This process is known as *unpacking* the array. Here is the general format of a structured binding declaration:

auto [*variable1*, *variable2*, *etc*…] = *array*;

The statement begins with the auto key word, followed by a list of variable names enclosed in square brackets ([]). Next, an assignment operator appears, followed by the name of an array. When the statement executes, the variables that are named inside the square brackets will be defined and initialized with the values that are stored in the array. The first variable listed inside the brackets will be assigned the value of element 0, the second variable listed inside the brackets will be assigned the value of element 1, and so on. The number of variables that are listed inside the square brackets must be the same as the number of elements in the array. Otherwise, a compiler error will occur.

Let's look at an example. Suppose we have three values stored in an array, as shown here:

int testScores[] = {80, 90, 100};

The following statement is a structured binding declaration that unpacks the array and assigns its values to variables:

auto [score1, score2, score3] = testScores;

This statement defines three int variables named score1, score2, and score3. The variables are defined as ints because the elements of the testScores array are ints. After the statement executes, the score1 variable will be assigned 80, the score2 variable will be assigned 90, and the score3 variable will be assigned 100.

The following code snippet shows how the array can be unpacked into variables that are used in a calculation:

int testScores[] = {80, 90, 100};

auto [score1, score2, score3] = testScores;

double average = (score1 + score2 + score3) / 3.0;

**Checkpoint**

Assume the following array has been defined:

int numbers[] = {1, 2, 3};

Write a structured binding declaration that defines three variables named a, b, and c. The statement should unpack the numbers array and assign the values of its elements to the variables a, b, and c.

**Insert 418-A**

**Structured Binding Declarations and Array Arguments**

If an array has been passed as an argument into a function, you cannot use a structured binding declaration in the receiving function to unpack the array. Inside the receiving function, the compiler only has the beginning memory address of the array argument and does not have enough information to determine the size the array. You can only use a structured binding declaration to unpack an array that has a known size, such as one that has been defined in the same function as where the structured binding declaration is used.