Chapter 6
Arrays

At a Glance

- Overview
- Objectives
- Quick Quizzes
- Additional Resources
- Key Terms
Overview

Chapter 6 provides an introduction to arrays and subscripts. Students will learn the basic steps involved in declaring, initializing, loading, and searching arrays. They will learn about using parallel arrays to solve problems. Students will become familiar with using flags to indicate success in searching and to trigger early exits from search loops. Finally, students will learn how to search for values within ranges.

Chapter Objectives

In this chapter, you will learn about:

- Arrays and how they occupy computer memory
- Manipulating an array to replace nested decisions
- Using constants with arrays
- Searching an array
- Using parallel arrays
- Searching an array for a range match
- Remaining within array bounds
- Using a for loop to process arrays

Understanding Arrays and How They Occupy Computer Memory

1. Introduce the concept of an array.

As an introduction, an array is like one row in an Excel spreadsheet. Each cell has a unique identifier, and each cell contains a value.

2. Understand the concept of a subscript (or index).

How Arrays Occupy Computer Memory

1. All array items have the same name and the same data type but are uniquely identified by their index position.

2. The array elements are stored in contiguous locations in memory.

A good conceptual example for arrays is the addresses of several people who all live in the same apartment building. Their addresses all have the same street name, but they have different apartment numbers. Relate the row of mailboxes to an array, as holding places for (mail) values.

3. The measure of size for an array is the count of its elements.
4. Understand how subscripts are denoted in programming languages.

5. Understand the concept of zero-based counting for array subscripts.

The highest subscript value will always be one less than the size of the array (the number of elements) because subscripts are zero-relative.

Manipulating an Array to Replace Nested Decisions

1. Walk through Figure 6-3, noting that this approach does not use arrays.

2. Using the flowcharts and pseudocode in Figures 6-4 and 6-5, explain that arrays can be used in the program but do not necessarily simplify the processing.

3. Using Figure 6-6, explain how to use an array to simplify the processing. The entire program is shown in Figure 6-7.

Quick Quiz 1

1. Each element of an array has the same _____ and the same _____.
   Answer: name, data type

2. Each element of an array has a unique ______.
   Answer: subscript or index

3. True or False: Array elements are stored sequentially in computer memory.
   Answer: True

Using Constants with Arrays

1. Describe the three ways to use constants with arrays, listed on page 240.

Using a Constant as the Size of an Array

1. Understand the benefits of using a constant for the array size rather than a magic number.

Using Constants as Array Element Values

1. Review the example of using constants for array element values on page 241.
Using a Constant as an Array Subscript

1. Review the example on page 241 of using a constant for an array subscript.

Searching an Array

1. Describe the use of the array to hold nonconsecutive item numbers, and understand the concept of searching for an item in an array using a linear search.

   Using a loop with an array can replace a series of nested if statements.

2. Understand the concept of a flag.

3. Describe the steps involved in searching an array and how a flag can be used to indicate the item has been found.

4. Review the sample program, using Figure 6-8 and focusing on the array search portion. Understand the use of the if statement nested within the while loop.

Using Parallel Arrays

1. Understand the concept of parallel arrays.

   This is a difficult concept for new programmers. The key to successful parallel arrays is ensuring that related items are in the same position in their arrays. Then, the same subscript value used with each array will retrieve the related items.

2. Review the sample program logic, using Figure 6-10. Understand how simple the code becomes when using parallel arrays.

Improving Search Efficiency

1. Understand the improvement in efficiency that can be realized by exiting a search loop when a match is found.

2. Understand the concept of exiting a loop early, and understand that its benefit increases with the size of the array.

3. Review the sample application, using Figure 6-11 and noting the early exit code.

Searching an Array for a Range Match
1. Understand the concept of searching for a match within a range of values.

2. Review the sample application input, using Figure 6-12, and walk through the drawbacks of the initial approach listed on page 254.

3. Understand the solution of creating four discount array elements and using parallel arrays to search for appropriate discount level.

4. Each array element holds a single value, not a range of values.

5. The array holds the low end of each quantity range.

6. Understand why you should start with the last array element and work backwards.

7. Review the sample program logic, using Figure 6-15.

Remaining within Array Bounds

1. Arrays are composed of elements of the same data type, which occupy the same number of bytes in memory.

2. Walk through Figure 6-16, which uses an array to determine a month string from a numeric entry, discussing what happens if the user enters a value that is too small or too large.

3. If the number entered by the user is too small or too large, the subscript is out of bounds.

Different programming languages report this error differently. In addition to “subscript out of bounds” or “subscript out of range,” you may also see messages such as “invalid index.”

Array processing is prone to “off by one” errors, because loops are used with zero-relative subscripts.

4. the program can check the validity of the data before using the user’s input to index the array.

5. Review the use of zero-based arrays again, and review how to determine the array bounds based on the size (number of elements) of the array.

Using a for Loop to Process Arrays

1. Review the for loop, and it is particularly convenient when working with arrays.
2. The highest usable subscript is one less than the size of the array.

The last valid subscript is one less than the size of the array. Emphasize that if an array has 12 items, its valid subscripts are 0 through 11.

3. Walk through Figures 6-17 and 6-18 to illustrate how to use a for loop to output array elements, and discuss why 6-17 is less efficient than 6-18.

Quick Quiz 2

1. What programming construct should be used to search through an array for a matching item?
   Answer: a loop

2. How are the items in two parallel arrays related?
   Answer: by having the same relative position in the array

3. The first element in an array has a subscript of ____.
   Answer: zero

4. The last element in an array has a subscript of ____.
   Answer: one less than the size of the array

Additional Resources

1. Javascript arrays:
   www.devarticles.com/c/a/JavaScript/JavaScript-Arrays/

2. Tutorial on arrays in C and C++:
   www.augustcouncil.com/~tgibson/tutorial/arr.html

3. Tutorial on arrays in Java:
   http://java.sun.com/docs/books/tutorial/java/nutsandbolts/arrays.html

4. A free online book chapter on arrays:
   http://webster.cs.ucr.edu/AoA/Windows/HTML/Arrays.html

Key Terms

- **Array** – a series or list of values in computer memory, all of which have the same name but are differentiated with special numbers called subscripts.

- **Binary search** – one that starts in the middle of a sorted list and then determines whether it should continue higher or lower to find a target value.
- **Element** – a single data item in an array.
- **Flag** – a variable that you set to indicate whether some event has occurred.
- **Indirect relationship** – describes the relationship between parallel arrays in which an element in the first array does not directly access its corresponding value in the second array.
- **Linear search** – a search through a list from one end to the other.
- **Out of bounds** – describes an array subscript that is not within the range of acceptable subscripts for the array.
- **Parallel arrays** – each element in one array is associated with the element in the same relative position in the other array(s).
- **Populating an array** – the act of assigning values to the array elements.
- **Size of the array** – the number of elements it can hold.
- **Subscript**, also called an **index** – a number that indicates the position of a particular item within an array.