

8

Arrays

OBJECTIVES

In this chapter you will learn:

- What arrays are.
- To declare arrays, initialize arrays and refer to individual elements of arrays.
- To use the foreach statement to iterate through arrays.
- To pass arrays to methods.
- To declare and manipulate multidimensional arrays.
- To write methods that use variable-length argument lists.
- To read command-line arguments into an application.

8.2 Arrays

• Arrays

- Group of variables
 - Have same type
- Reference type
- Remain same size once created
 - Fixed-length entries

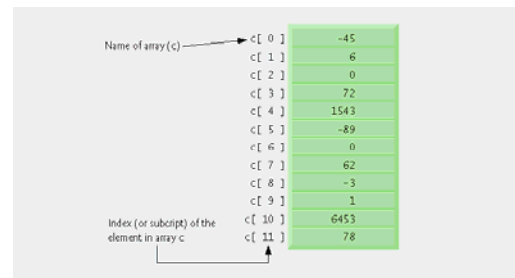


Fig. 8.1 | A 12-element array.

8.2 Arrays (Cont.)

• Index

- Position number in square brackets
- Must be positive integer or integer expression
- First element has index zero

• Examine array C

- **c** is the array name
- **c.Length** accesses array C's length
- **c** has 12 elements (**c[0]**, **c[1]**, ... **c[11]**)

8.3 Declaring and Creating Arrays

• Declaring and creating arrays

- Arrays are objects that occupy memory
- Created dynamically with keyword **new**

```
int[] c = new int[ 12 ];
```

 - Equivalent to

```
int[] c; // declare array variable
c = new int[ 12 ]; // create array
```
- We can create arrays of objects too

```
String[] b = new string[ 100 ];
```

8.4 Examples Using Arrays

- Declaring arrays
- Creating arrays
- Initializing arrays
- Manipulating array elements

© 2006 Pearson Education, Inc. All rights reserved.

```
1 // Fig. 8.2: InitArray.cs
2 // Creating an array.
3 using System;
4
5 public class InitArray
6 {
7     public static void Main( string[] args )
8     {
9         int[] array; // declare array named array
10
11         // create the space for array and initialize to default zeros
12         array = new int[ 10 ]; // 10 int elements
13
14         Console.WriteLine( "{0}{1,8}", "Index", "Value" ); // headings
15
16         // output each array element's value
17         for ( int counter = 0; counter < array.Length; counter++ )
18             Console.WriteLine( "{0,5}{1,8}", counter, array[ counter ] );
19     } // end Main
20 } // end class InitArray
```

Index	Value
0	0
1	0
2	0
3	0
4	0
5	0
6	0
7	0
8	0
9	0

© 2006 Pearson Education, Inc. All rights reserved.

8.4 Examples Using Arrays (Cont.)

- Using an array initializer
 - Use *initializer list*
 - Items enclosed in braces ({})
 - Items in list separated by commas
- ```
int[] n = { 10, 20, 30, 40, 50 };
```
- Creates a five-element array
  - Index values of 0, 1, 2, 3, 4
- Do not need keyword `new`

© 2006 Pearson Education, Inc. All rights reserved.

```
1 // Fig. 8.3: InitArray.cs
2 // Initializing the elements of an array with an array initializer.
3 using System;
4
5 public class InitArray
6 {
7 public static void Main(string[] args)
8 {
9 // initializer list specifies the value for each element
10 int[] array = { 32, 27, 64, 18, 95, 14, 90, 70, 60, 37 };
11
12 Console.WriteLine("{0}{1,8}", "Index", "Value"); // headings
13
14 // output each array element's value
15 for (int counter = 0; counter < array.Length; counter++)
16 Console.WriteLine("{0,5}{1,8}", counter, array[counter]);
17 } // end Main
18 } // end class InitArray
```

| Index | Value |
|-------|-------|
| 0     | 32    |
| 1     | 27    |
| 2     | 64    |
| 3     | 18    |
| 4     | 95    |
| 5     | 14    |
| 6     | 90    |
| 7     | 70    |
| 8     | 60    |
| 9     | 37    |

© 2006 Pearson Education, Inc. All rights reserved.

```
1 // Fig. 8.4: InitArray.cs
2 // Calculating values to be placed into the elements of an array.
3 using System;
4
5 public class InitArray
6 {
7 public static void Main(string[] args)
8 {
9 const int ARRAY_LENGTH = 10; // create a named constant
10 int[] array = new int[ARRAY_LENGTH]; // create array
11
12 // calculate value for each array element
13 for (int counter = 0; counter < array.Length; counter++)
14 array[counter] = 2 + 2 * counter;
15
16 Console.WriteLine("{0}{1,8}", "Index", "Value"); // headings
17
18 // output each array element's value
19 for (int counter = 0; counter < array.Length; counter++)
20 Console.WriteLine("{0,5}{1,8}", counter, array[counter]);
21 } // end Main
22 } // end class InitArray
```

| Index | Value |
|-------|-------|
| 0     | 2     |
| 1     | 4     |
| 2     | 6     |
| 3     | 8     |
| 4     | 10    |
| 5     | 12    |
| 6     | 14    |
| 7     | 16    |
| 8     | 18    |
| 9     | 20    |

© 2006 Pearson Education, Inc. All rights reserved.

| Index | Value |
|-------|-------|
| 0     | 2     |
| 1     | 4     |
| 2     | 6     |
| 3     | 8     |
| 4     | 10    |
| 5     | 12    |
| 6     | 14    |
| 7     | 16    |
| 8     | 18    |
| 9     | 20    |

[Outline](#)

InitArray.cs  
(2 of 2)

© 2006 Pearson Education, Inc. All rights reserved.

```

1 // Fig. 8.5: SumArray.cs
2 // Computing the sum of the elements of an array.
3 using System;
4
5 public class SumArray
6 {
7 public static void Main(string[] args)
8 {
9 int[] array = { 87, 68, 94, 100, 83, 78, 85, 91, 76, 87 };
10 int total = 0;
11
12 // add each element's value to total
13 for (int counter = 0; counter < array.Length; counter++)
14 total += array[counter];
15
16 Console.WriteLine("Total of array elements: {0}", total);
17 } // end Main
18 } // end class SumArray

```

Total of array elements: 849

© 2006 Pearson Education, Inc. All rights reserved.

```

1 // Fig. 8.6: BarChart.cs
2 // Bar chart printing application.
3 using System;
4
5 public class BarChart
6 {
7 public static void Main(string[] args)
8 {
9 int[] array = { 0, 0, 0, 0, 0, 0, 1, 2, 4, 2, 1 };
10
11 Console.WriteLine("Grade distribution: ");
12
13 // for each array element, output a bar of the chart
14 for (int counter = 0; counter < array.Length; counter++)
15 {
16 // output bar labels ("00-09: ", ..., "90-99: ", "100: ")
17 if (counter == 10)
18 Console.WriteLine(" 100: ");
19 else
20 Console.WriteLine("{0:D2}-{1:D2}: ",
21 counter * 10, counter * 10 + 9);
22
23 // print bar of asterisks
24 for (int stars = 0; stars < array[counter]; stars++)
25 Console.WriteLine(" * ");
26 }
27 }
28 }

```

© 2006 Pearson Education, Inc. All rights reserved.

```

26 Console.WriteLine(); // start a new line of output
27 } // end outer for
28 } // end Main
29 } // end class BarChart
30 } // end class BarChart

```

Grade distribution:

|        |       |
|--------|-------|
| 00-09: |       |
| 10-19: |       |
| 20-29: |       |
| 30-39: |       |
| 40-49: |       |
| 50-59: | *     |
| 60-69: | **    |
| 70-79: | ***   |
| 80-89: | ****  |
| 90-99: | ***** |
| 100:   |       |

© 2006 Pearson Education, Inc. All rights reserved.

```

1 // Fig. 8.7: RollDie.cs
2 // Roll a six-sided die 6000 times.
3 using System;
4
5 public class RollDie
6 {
7 public static void Main(string[] args)
8 {
9 Random randomNumbers = new Random(); // random number generator
10 int[] frequency = new int[7]; // array of frequency counters
11
12 // roll die 6000 times; use die value as frequency index
13 for (int roll = 1; roll <= 6000; roll++)
14 ++frequency[randomNumbers.Next(1, 7)];
15
16 Console.WriteLine("{0}(1,10)", "Face", "Frequency");
17
18 // output each array element's value
19 for (int face = 1; face < frequency.Length; face++)
20 Console.WriteLine("{0,4}(1,10)", face, frequency[face]);
21 } // end Main
22 } // end class RollDie

```

| Face | Frequency |
|------|-----------|
| 1    | 956       |
| 2    | 981       |
| 3    | 1001      |
| 4    | 1030      |
| 5    | 1035      |
| 6    | 997       |

© 2006 Pearson Education, Inc. All rights reserved.

```

1 // Fig. 8.8: StudentPoll.cs
2 // Poll analysis application.
3 using System;
4
5 public class StudentPoll
6 {
7 public static void Main(string[] args)
8 {
9 // array of survey responses
10 int[] responses = { 1, 2, 6, 4, 8, 5, 9, 7, 8, 10, 1, 6, 3, 8, 6,
11 10, 3, 8, 2, 7, 6, 5, 7, 6, 8, 6, 7, 5, 6, 6, 5, 6, 7, 5, 6,
12 4, 8, 6, 8, 10 };
13 int[] frequency = new int[11]; // array of frequency counters
14
15 // for each answer, select responses element and use that value
16 // as frequency index to determine element to increment
17 for (int answer = 0; answer < responses.Length; answer++)
18 ++frequency[responses[answer]];
19
20 Console.WriteLine("{0}(1,10)", "Rating", "Frequency");
21 }
22 }

```

© 2006 Pearson Education, Inc. All rights reserved.

```

21
22 // output each array element's value
23 for (int rating = 1; rating < frequency.Length; rating++)
24 Console.WriteLine("{0,6}(1,10)", rating, frequency[rating]);
25 } // end Main
26 } // end class StudentPoll

```

| Rating | Frequency |
|--------|-----------|
| 1      | 2         |
| 2      | 2         |
| 3      | 2         |
| 4      | 2         |
| 5      | 5         |
| 6      | 11        |
| 7      | 5         |
| 8      | 7         |
| 9      | 1         |
| 10     | 3         |

© 2006 Pearson Education, Inc. All rights reserved.

## Error-Prevention Tip 8.2

When writing code to loop through an array, ensure that the array index remains greater than or equal to 0 and less than the length of the array. The loop-continuation condition should prevent the accessing of elements outside this range.

© 2006 Pearson Education, Inc. All rights reserved.

## 8.6 foreach Statement

### • foreach

- Iterates through the elements of an entire array or collection without using counter
  - Can access array elements
  - Cannot access the counter indicating the index
  - Cannot modify array elements

### - Syntax

**foreach** ( *type identifier* | *n arrayName* )  
*statement*

© 2006 Pearson Education, Inc. All rights reserved.

```
1 // Fig. 8.12: ForEachTest.cs
2 // Using foreach statement to total integers in an array. Outline
3 using System;
4
5 public class ForEachTest ForEachTest.cs
6 {
7 public static void Main(string[] args)
8 {
9 int[] array = { 87, 68, 94, 100, 83, 78, 85, 91, 76, 87 };
10 int total = 0;
11
12 // add each element's value to total
13 foreach (int number in array)
14 total += number;
15
16 Console.WriteLine("Total of array elements: {0}", total);
17 } // end Main
18 } // end class ForEachTest

```

Total of array elements: 849

Lines 13-14 equivalent to:

```
for (int counter = 0; counter < array.Length; counter++)
 total += array[counter];
```

© 2006 Pearson Education, Inc. All rights reserved.