

Chapter 8

Arrays, Timers, and More

Starting Out with Visual Basic .NET 2nd Edition

8.3

More About Array Processing

There Are Many Uses of Arrays and Many Programming Techniques That Involve Them

Arrays May Be Used to Total Values and Search for Data

Related Information May Be Stored in Multiple Parallel Arrays

In Addition, Arrays May Be Resized at Run Time

Starting Out with Visual Basic .NET 2nd Edition

Determining the Number of Elements in an Array

- Array's have a Length property that holds the number of elements in the array

```
Dim values(25) As Integer

For count = 0 to (values.Length - 1)
    MessageBox.Show(values(count).ToString)
Next count
```

- Note that the length is the number of elements on the array, not the largest subscript

Starting Out with Visual Basic .NET 2nd Edition

How to Total the Values in an Array

- The values are summed up in a variable that is initialized to zero beforehand:

```
Dim total As Integer = 0 ' Initialize accumulator
For count = 0 To (units.Length - 1)
    total += units(count)
Next count
```

Starting Out with Visual Basic .NET 2nd Edition

Getting the Average of the Values in a Numeric Array

- This algorithm first totals the values, then calculates the average:

```
Dim total As Integer = 0 ' Initialize accumulator
Dim average As Single

For count = 0 To (units.Length - 1)
    total += units(count)
Next count
average = total / units.Length
```

Starting Out with Visual Basic .NET 2nd Edition

Finding the Highest and Lowest Values in a Numeric Array

- Pick the first element as the highest, then look through the rest of the array for even higher ones, always saving the highest value

```
highest = numbers(0)
For count = 1 To (numbers.Length - 1)
    If numbers(count) > highest Then
        highest = numbers(count)
    End If
Next count
```

- Proceed analogously to find the lowest value

Starting Out with Visual Basic .NET 2nd Edition

Copying One Array's Contents to Another

- This is done by copying the elements one at a time, thusly

```
For count = 0 To 100
    newValues(count) = oldValues(count)
Next count
```

- Note that newvalues = oldvalues will not work for this use because the value of oldvalues is the storage location for that array (not all of the contents of oldvalues)

Starting Out with Visual Basic .NET 2nd Edition

Parallel Arrays

- Sometimes it is useful to store related data in two or more arrays
 - Hence the i^{th} element of one array is related to the i^{th} element of another
- Then the program can access this related information by using the same subscript on both arrays

```
Dim names(4) As String
Dim addresses(4) As String
```

Starting Out with Visual Basic .NET 2nd Edition

Parallel Arrays Example

```
Dim names(4) As String
Dim addresses(4) As String

For count = 0 To 4
    lstPeople.Items.Add( "Name: " & names(count) & _
        " Address: " & addresses(count))
Next count
```

Starting Out with Visual Basic .NET 2nd Edition

Parallelism Between Arrays and List Boxes and Combo Boxes

```
' Initialize a List Box with names
lstPeople.Items.Add("Jean James") ' Subscript 0
lstPeople.Items.Add("Kevin Smith") ' Subscript 1
lstPeople.Items.Add("Joe Harrison") ' Subscript 2

' Initialize an Array with corresponding phone numbers
phoneNumbers(0) = "555-2987"
phoneNumbers(1) = "555-5656"
phoneNumbers(2) = "555-8897"

' Process a selection
If lstPeople.SelectedIndex > -1 And _
    lstPeople.SelectedIndex < phoneNumbers.Length Then
    MessageBox.Show(phoneNumbers(lstPeople.SelectedIndex))
Else
    MessageBox.Show("That is not a valid selection.")
End If
```

Starting Out with Visual Basic .NET 2nd Edition

Searching Arrays, The Search

- This code hunts for the value 100 in the array scores

```
' Search for a 100 in the array.
found = False
count = 0
Do While Not found And count < scores.Length
    If scores(count) = 100 Then
        found = True
        position = count
    End If
    count += 1
Loop
```

Starting Out with Visual Basic .NET 2nd Edition

Searching Arrays, Acting on the Result

- This code indicates whether or not the value was found or not

```
' Was 100 found in the array?
If found Then
    MessageBox.Show( _
        "Congratulations! You made a 100 on test " & _
        (position + 1).ToString, "Test Results")
Else
    MessageBox.Show( _
        "You didn't score a 100, but keep trying!", _
        "Test Results")
End If
```

Starting Out with Visual Basic .NET 2nd Edition

Sorting an Array

- There is an Array.Sort method that sorts arrays in ascending order, examples:

```
Dim numbers() As Integer = { 7, 12, 1, 6, 3 }
Array.Sort(numbers)

' OR

Dim names() As String = { "Sue", "Kim", "Alan", "Bill" }
Array.Sort(names)
```

Starting Out with Visual Basic .NET 2nd Edition

Resizing an Array

```
ReDim [Preserve] Arrayname (UpperSubscript)
```

- ReDim is a new keyword
- If Preserve is specified, the existing contents of the array are preserved
- Arrayname names the existing array
- UpperSubscript specifies the new highest subscript value

Starting Out with Visual Basic .NET 2nd Edition

Resizing Example

```
Dim scores() As Single ' Start array as having "Nothing"
                          ' Now obtain a size from the user
numScores = Val(TextBox("Enter the number of test scores."))
If numScores > 0 Then
    ReDim scores(numScores - 1)
Else
    MessageBox.Show("You must enter 1 or greater.")
End If
```

Starting Out with Visual Basic .NET 2nd Edition

8.6 Enabled Property, Timer Control, Splash Screens

You Disable Controls by Setting Their Enabled Property to False

The Timer Control Allows Your Application to Execute a Procedure at Regular Time Intervals

Splash Screens Are Forms That Appear As an Application Begins Executing

Starting Out with Visual Basic .NET 2nd Edition

Enabled Property, I

- Most controls have a Boolean property named Enabled
- When a control's *Enabled property* is set to false, it is considered disabled, which means:
 - It cannot receive the focus and cannot respond to events generated by the user
 - In addition, it will appear dimmed, or grayed out

Starting Out with Visual Basic .NET 2nd Edition

Enabled Property, II

- This property defaults to true (Enabled)
- Under program control this property can be set to whatever the application logic dictates

```
radBlue.Enabled = False
```

Starting Out with Visual Basic .NET 2nd Edition

Timer Control, I

- This control, when placed on a form, generates Tick events on a regular interval
- If there is a corresponding Tick event procedure, it executes at these intervals
- Hence, your form can perform needed operations on a regular interval

Starting Out with Visual Basic .NET 2nd Edition

Timer Control, II

- The timer control has two important properties:
 - Enabled - if set to True, it generates the Tick events, otherwise not
 - Interval - holds the interval between Ticks in milliseconds (thousandths of a second)

Starting Out with Visual Basic .NET 2nd Edition

Splash Screens

- This is a form, typically with the application's logo, that is often displayed while an application is loading, it should:
 - Have its Topmost property set to True so that it will be seen in preference to the applications other forms
 - Disappear shortly (using a Timer)
 - Be modeless

Starting Out with Visual Basic .NET 2nd Edition

8.6 Anchoring and Docking Controls

Controls Have Two Properties, Anchor and Dock,
That Allow You to Determine the Control's
Position on the Form When the Form Is Resized at
Run Time

Starting Out with Visual Basic .NET 2nd Edition

Anchor Property

- A control may be Anchored to two adjacent sides of the form
 - This situation will maintain the same distances from those edges whenever the form is resized by the user (control does not change size)
- A control may be anchored to opposite sides of a form
 - This situation will again maintain those distances, but by changing the size of the control

Starting Out with Visual Basic .NET 2nd Edition

Dock Property

- A control can be docked against any side of a form
- Whenever the form is resized, the control will change in size also to fill up the edge that it is docked to

Starting Out with Visual Basic .NET 2nd Edition

8.7 Random Numbers

Visual Basic .NET Provides the Tools to Generate Random Numbers and Initialize the Sequence of Random Numbers With a Seed Value

Starting Out with Visual Basic .NET 2nd Edition

Random Number Generation, Step 1

- Random number generation is initiated with the selection of a "seed" number

```
Randomize [Number]
```

- If you wish to repeat the same sequence of random numbers, then use the above statement with the same argument Number
- If you do not want to repeat a sequence, omit the argument

Starting Out with Visual Basic .NET 2nd Edition

Random Number Generation, Step 2

- Obtain the next random number in the sequence

```
randomNumber = Rnd
```

- Repeatedly use a statement like the above to generate additional random numbers

Starting Out with Visual Basic .NET 2nd Edition

Random Number Range

- Rnd creates random numbers in the range of 0.0 to 1.0
- To scale these to a range between two integers that may be needed in your program:

```
randomNumber = Int(LowerNumber + Rnd * _  
                  (UpperNumber - LowerNumber))
```

Starting Out with Visual Basic .NET 2nd Edition
