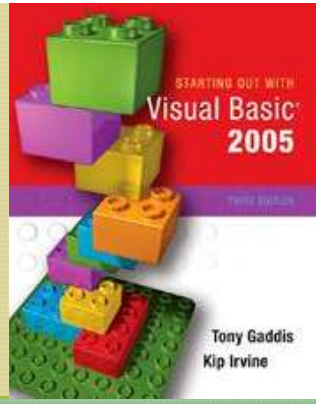
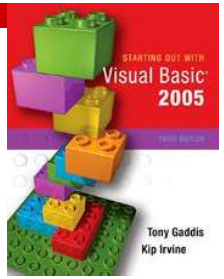


8.2 More About Array Processing

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

For Example, Arrays Are Used to Total Values or Search for Data

Determining Number of Elements



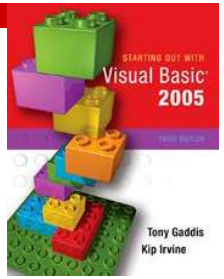
- Arrays have a *Length* property that holds the number of elements in the array

```
Dim values(25) As Integer

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

- Note that length is number of array elements, not the upper subscript of the array
- *Length* property always 1 greater than the upper subscript

Total the Values in an Array

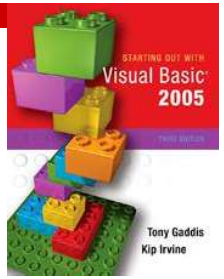


- Variable to hold sum (**`intTotal`**) is initialized to zero prior to loop
- Iterate over all elements, adding each element to **`intTotal`**

```
Dim intUnits(24) as Integer  'Declare array
Dim intTotal As Integer = 0  'Initialize accumulator
Dim intCount as Integer      'Declare loop counter

'Find total of all values held in array
For intCount = 0 To (intUnits.Length - 1)
   intTotal += intUnits(intCount)
Next intCount
```

Average the Values in an Array



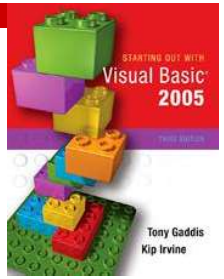
- Similar to previous example but then divides total by number of elements to get average

```
Dim intUnits(24) as Integer  'Declare array
Dim intTotal As Integer = 0  'Initialize accumulator
Dim dblAverage as Double     'Declare average var
Dim intCount as Integer      'Declare loop counter

'Find total of all values held in array
For intCount = 0 To (intUnits.Length - 1)
   intTotal += intUnits(intCount)
Next intCount

'Use floating-point division to compute average
dblAverage = intTotal / intUnits.Length
```

Find Highest & Lowest Array Values



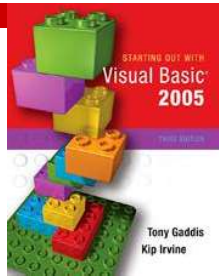
- Pick first element as the highest
- Search rest of array for higher values
- If a higher value is located, save the value

```
Dim intNumbers(24) as Integer
Dim intCount as Integer
Dim intHighest as Integer = intNumbers(0)

For intCount = 1 To (intNumbers.Length - 1)
      If intNumbers(intCount) > intHighest Then
            intHighest = intNumbers(intCount)
      End If
Next intCount
```

- Use similar logic to find lowest value

Copy Values in One Array to Another

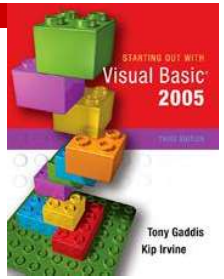


- Done by copying elements one at a time

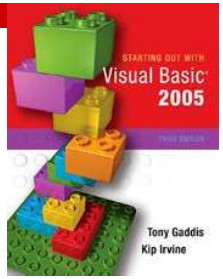
```
For intCount = 0 To 100
      newValues(intCount) = oldValues(intCount)
Next intCount
```

- Note that this cannot be done by a simple assignment **`newValues = oldValues`**
  - Causes **`newValues`** variable to reference the same memory location as **`oldValues`**
  - Thus any change to **`oldValues`** will affect **`newValues`** and vice versa

Parallel Arrays



- Sometimes useful to store related data in two or more arrays called *parallel arrays*
  - Causes the i^{th} element of one array to be related to the i^{th} element of another
- Allows related data to be accessed using the same subscript on both arrays

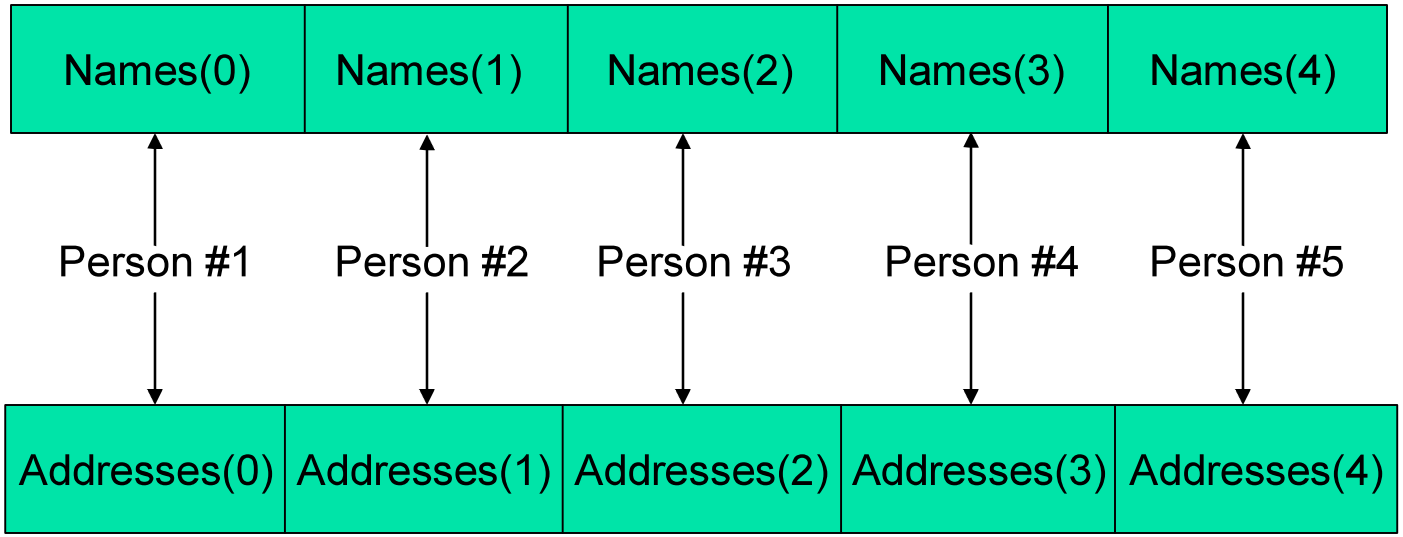


Parallel Arrays Example

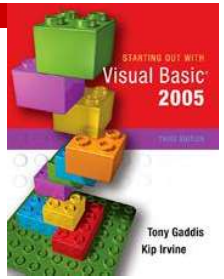
- Assume the following array declarations

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

- Element 1 refers to the same person with name in one array and address in the other



Parallel Arrays Processing



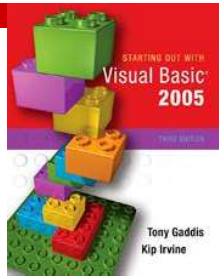
- Use parallel array processing to add name and address of each person to a list box

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

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

- Tutorial 8-2 has an example of parallel arrays

Array & List Box Parallel Processing

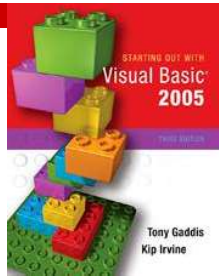


- A list box selection used as an array index

```
' 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
```

Searching Arrays, The Search

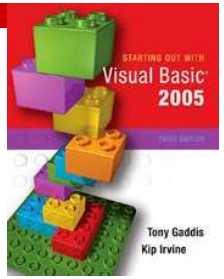


- Find 1st instance of value 100 in array *scores*
- intPosition gives position of this value

```
Dim blnFound as Boolean = False
Dim intCount as Integer = 0
Dim intPosition as Integer = -1

' Search for a 100 in the array
Do While Not blnFound And intCount < scores.Length
      If scores(intCount) = 100 Then
            blnFound = True
            intPosition = intCount
      End If
intCount += 1
Loop
```

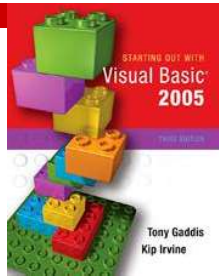
Searching Arrays, The Result



- Indicates whether the value was found
- If found, position is given as a test number

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

Sorting an Array



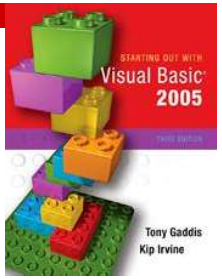
- Arrays have a *Sort* method
- Arranges elements in ascending order (lowest to highest)
- Sorted so that numbers = {1, 3, 6, 7, 12}

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

- Sorted so that names = {Alan, Bill, Kim, Sue}

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

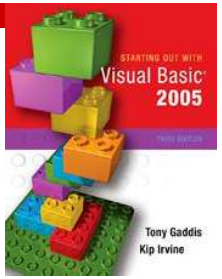
Resizing an Array




```
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
- Can declare an array with no subscript and state number of elements later with *ReDim*

Resizing Example



- Array *scores* declared with no elements
- User prompted for number of elements
- ReDim resizes array based on user input

```
Dim scores() As Single      ' Declared with no elements
Dim numScores as Integer

' Obtain number of elements from the user
numScores = CInt(InputBox("Enter number of test scores"))
If numScores > 0 Then
       ReDim scores(numScores - 1)
Else
       MessageBox.Show("You must enter 1 or greater.")
End If
```