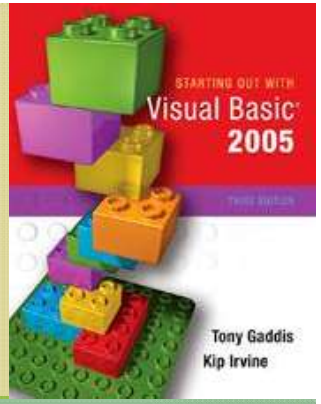
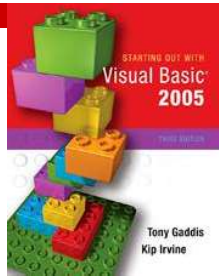


8.4 Multidimensional Arrays

You May Create Arrays With More Than Two Subscripts to Hold Complex Sets of Data

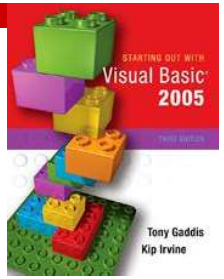
A Two Dimensional Array Picture



- Thus far, arrays have been *one-dimensional*
- However, arrays can also be *two-dimensional*
- Picture a two-dimensional array like a spreadsheet with rows and columns

	Column 0	Column 1	Column 2	Column 3
Row 0				
Row 1				
Row 2				

Two Dimensional Array Syntax



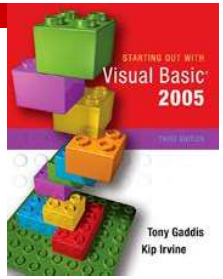

```
Dim ArrayName(UpperRow, UpperColumn) As DataType
```

- UpperRow and UpperColumn give the highest subscript for the row and column indices of the array
- The array on the previous slide could be:

```
Dim array(2,3) As Single
```

  - Defines three rows; 0, 1, and 2
  - And four columns; 0, 1, 2, and 3

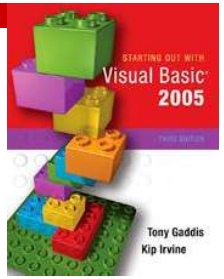
Two Dimensional Array Subscripts




```
Dim array(2,3) As Single
```

	Column 0	Column 1	Column 2	Column 3
Row 0	array(0,0)	array(0,1)	array(0,2)	array(0,3)
Row 1	array(1,0)	array(1,1)	array(1,2)	array(1,3)
Row 2	array(2,0)	array(2,1)	array(2,2)	array(2,3)

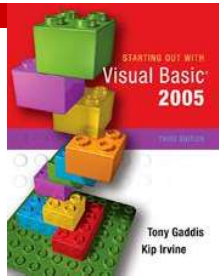
Nested Loops And Two Dimensions



- Nested loops are often used in processing two-dimensional arrays
- In the example, a nested loop is used to insert a value into every element of array *scores*

```
For row = 0 To 2
      For col = 0 To 3
            num = Val(InputBox("Enter a score."))
            scores(row, col) = num
      Next col
Next row
```

Implicit Sizing and Initialization



- Can be used with multi-dimensional arrays:

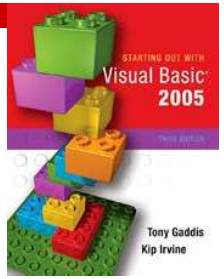
```
Dim numbers(,) As Integer = _
        { {1, 2, 3}, {4, 5, 6}, {7, 8, 9} }
```

- Row 0 values
- Row 1 values
- Row 2 values

- Initializes array *numbers* with the following values

	Col 0	Col 1	Col 2
Row 0	1	2	3
Row 1	4	5	6
Row 2	7	8	9

For Each Loop With Two Dimensions

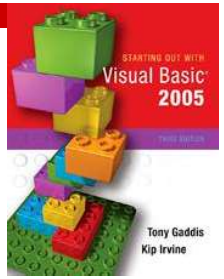


- A For Each Loop will process all elements of an array without requiring nested loops
- The example below computes the sum of all elements
- Total has the value 45 when loop is complete

```
Dim numbers(,) As Integer = {{1, 2, 3}, _
                             {4, 5, 6}, _
                             {7, 8, 9}}

For Each element In numbers
      total += element
Next element
```

Sum Two-Dimensional Array Columns

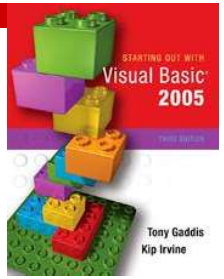


- Outer loop controls column subscript
- Inner loop controls row subscript

```
' Process each column in turn
For col = 0 To 2
    ' Initialize column accumulator
    total = 0
    ' Sum the column values from each row
    For row = 0 To 4
        total += values(row, col)
    Next row

    ' Display the sum of the column.
    MessageBox.Show("Sum of column " & _
        col.ToString & " is " & total.ToString)
Next col
```

Three-Dimensional Arrays & Beyond



- VB allows arrays of up to 32 dimensions
- Beyond three dimensions, they are difficult to visualize
- But, all one needs to do is to be consistent in the use of the different indices