Multi-dimensional Arrays in VB.NET

Dr. David R. Surma
Department of Computer and Information Sciences

Adapted from Drs. Adaikkalavan, Hakimzadeh, & Zhang
**Multi-dimensional Arrays**

- Arrays can have more than one dimension
- Like a table of values
- Or a cube of values

2-D

3-D
Multi-dimensional Arrays

- Syntax:

```dim ArrayName(HighestRowSubscript, HighestColumnSubscript) As DataType```

2-D
Example #1

- Two Dimensional Array

```
Dim ScoreBoard(1, 8) As Integer
```

- This declaration creates 18 storage locations (2x9) in which to put **Integer** values.

![Diagram of a 2x9 array]
Manipulating the Array

- In order to access each variable (array element), we must use two array indexes.

\[
\begin{align*}
\text{scoreBoard(0, 0)} &= 8 \\
\text{scoreBoard(1, 3)} &= 7 \\
\text{scoreBoard(1, 8)} &= 10
\end{align*}
\]
Example #2

- Two Dimensional Array

  \[ \text{Dim } \text{strName}(4, 5) \text{ As String} \]

- This declaration creates 30 storage locations (5x6) in which to put \textbf{String} values.
Example #3

- Two Dimensional Array

```
Dim gradeBook(4, 3) As Double
```

- This declaration creates 20 storage locations (5x4) in which to put **Double** values.
Operations on a 2-D Array

- Fill the array with values
- Do calculations with the array elements
- Displaying the contents of the array
' Declare a 2-D array with (4 x 3) cells to store the grades for a class
Dim gradeBook(4, 3) As Double

' Fill the array with exam grades
Dim row, col As Integer

Console.WriteLine("Please enter 15 grades: ")
For row = 0 To gradeBook.GetUpperBound(0)
    For col = 0 To gradeBook.GetUpperBound(1) - 1  ' leave the last column for average grade
        gradeBook(row, col) = CDbI(Console.ReadLine())
    Next
    gradeBook(row, gradeBook.GetUpperBound(1)) = 0  ' initialize the average to be zero
Next
Do calculations with the array elements

' Next, we calcule the average for each student
For row = 0 To gradeBook.GetUpperBound(0)
    ' Now we sum up each row then calculate the average
    For col = 0 To gradeBook.GetUpperBound(1) - 1
        gradeBook(row, gradeBook.GetUpperBound(1)) += gradeBook(row, col)
    Next
    gradeBook(row, gradeBook.GetUpperBound(1)) /= gradeBook.GetUpperBound(1)
Next

<p>| | | | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>76.5</td>
<td>84</td>
<td>88.5</td>
<td>83</td>
</tr>
<tr>
<td>82</td>
<td>86.5</td>
<td>88.5</td>
<td>85.67</td>
</tr>
<tr>
<td>92</td>
<td>94.5</td>
<td>96</td>
<td>94.17</td>
</tr>
<tr>
<td>78.5</td>
<td>72.5</td>
<td>76</td>
<td>75.67</td>
</tr>
<tr>
<td>63</td>
<td>72.5</td>
<td>58</td>
<td>64.5</td>
</tr>
</tbody>
</table>
' Now we display the contents of the 2-D array
Console.WriteLine(vbCrLf & "Here are all the grades: ")
For row = 0 To gradeBook.GetUpperBound(0)
    For col = 0 To gradeBook.GetUpperBound(1)
        Console.Write("{0, -6:.##}", gradeBook(row, col))
    Next
    Console.WriteLine()
Next
Console.WriteLine()
Example: let us play the “stars” again!

- “Store” the following image of “stars” in a 2-D array, then print the image out from the array.
Example: let us play the “stars” again!

Dim starArray1(9, 9) As Integer
Dim row, col As Integer

' Store the image of stars in the array
' 1 represents a star, 0 represents a space
For row = 0 To starArray1.GetUpperBound(0)
    For col = 0 To row
        starArray1(row, col) = 1
    Next
Next
Example: let us play the “stars” again!

' Now print the array elements, row by row
' print a star if element contains 1, a space if element contains 0
' start a new line for each array row
For row = 0 To starArray1.GetUpperBound(0)
    For col = 0 To starArray1.GetUpperBound(1)
        If starArray1(row, col) = 1 Then
            Console.Write("*")
        Else
            Console.Write(" ")
        End If
    Next
    Console.WriteLine()
Next
Now, solve the following problem!

- “Store” the following image of “stars” in a 2-D array, then print the image out from the array.