Discussion

- Final exam scores of five students (89, 95, 77, 65, 99)
  - How can these values be stored and used in calculations?
  - How can you accomplish this?
  - How many variables do you need?

Dim score0, score1, score2, score3, score4 As Double

score0 = 89
score1 = 95
score2 = 77
score3 = 65
score4 = 99

How about 22 students?
You need 22 memory locations / variables with unique names
B100/I101
Problem Solving Using Computers

Arrays in VB.NET

< Part I >

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

Adapted from Drs. Adaikkalavan, Hakimzadeh, Zhang
What do we need to learn in order to write computer programs?

- Fundamental programming constructs:
  - Variables,
  - Arithmetic operators,
  - Input and output
  - Conditionals,
  - Loops,
  - Arrays,
  - Subroutines and functions,
  - Structures, classes and objects,
  - Files
What is an Array?

- Array is a homogeneous aggregate (group) of data elements.

- Array is a collection of objects of the **same data type** (e.g., collection of integers).

- Array is a collection of similar variables which are identified under the **same name** and different index.

- Array elements are stored contiguously in memory.
Why use an Array?

- Arrays allow the programmer to create a series of variables and reference them using a single variable name.
Declaring an Array

Syntax:

```plaintext
Dim arrayName(highestSubScript) as DataType
```

- Array starts at 0 and highestSubScript is any non-negative integer
- DataType can be any data type
Declaring an Array

- Create ‘n’ variables of type `double` under the name `scores`.

  ```
  Dim scores(5) As Double
  ```

  tells the compiler to associate 6 memory locations with the identifier `scores`.

  ![Diagram showing array declaration]

  `scores(0)` and `scores(5)`

  It creates 6 elements! Not 5
### Storing values in the Array Elements

**Dim** `score(4)` **As Double**

<table>
<thead>
<tr>
<th>Array Name</th>
<th>Array Index</th>
<th>Value to be placed in each array element</th>
<th>Number of values that needs to be stored (starting at 0)</th>
</tr>
</thead>
<tbody>
<tr>
<td>score(0)</td>
<td>0</td>
<td>89</td>
<td></td>
</tr>
<tr>
<td>score(1)</td>
<td>1</td>
<td>95</td>
<td></td>
</tr>
<tr>
<td>score(2)</td>
<td>2</td>
<td>77</td>
<td></td>
</tr>
<tr>
<td>score(3)</td>
<td>3</td>
<td>65</td>
<td></td>
</tr>
<tr>
<td>score(4)</td>
<td>4</td>
<td>99</td>
<td></td>
</tr>
</tbody>
</table>

- `score(0) = 89`
- `score(1) = 95`
- `score(2) = 77`
- `score(3) = 65`
- `score(4) = 99`
Array Properties:

- The Array’s **Length** property returns the length (number of elements) of the array.

**Example:**

```vbnet
Dim scores(4) As Double

Console.WriteLine("Array length = {0}", scores.Length)
```

**Output:**

- **Array length = ?**
Array Methods:

- The Array’s “**GetUpperBound()**” property returns the highest index of the array.

- Example:
  ```csharp
  Dim scores(4) As Double
  Console.WriteLine("Array Upper Bound = \{0\}", scores.GetUpperBound(0))
  ```

- Output:
  - **Array Upper Bound = 4**

  The argument indicates the dimension of the array.
  (0 indicates the first dimensional array)
Recap: Length & Bounds

Dim scores(4) As Double

scores(0) = 89
scores(1) = 95
scores(2) = 77
scores(3) = 65
scores(4) = 99

Console.WriteLine("Score at 3 is {0}", scores(3))

Console.WriteLine("Array Length = {0}", scores.Length)

Console.WriteLine("Array U Bound = {0}", scores.GetUpperBound(0))

Console.WriteLine("Array L Bound = {0}", scores.GetLowerBound(0))
Operations on an Array:

- Initializing the array
- Inserting data in the array
- Displaying the cell contents of the array
- Searching an Array
- Finding highest/lowest value
Operations on an Array:

- Initializing the array:

```vbnet
Dim scores(4) As Double
Dim index As Integer

For index = 1 To scores.GetUpperBound(0)
    scores(index) = 0.0
Next index
```
Operations on an Array:

- Initializing the array:

Dim scores(4) As Double
Dim index As Integer

For index = 1 To scores.GetUpperBound(0)
    scores(index) = 0.0
Next index
Operations on an Array:

- Initializing the array:

```vba
Dim scores(4) As Double
Dim index As Integer

For index = 0 To scores.GetUpperBound(0)
    scores(index) = 0.0
Next index
```
Operations on an Array:

- Initializing the array (another way):

```vbnet
Dim scores(4) As Double
Dim index As Integer

For index = scores.GetLowerBound(0) To scores.GetUpperBound(0)
    scores(index) = 0.0
Next index
```
Operations on an Array:

- This is fine as well!

Dim scores() As Double = {0, 0, 0, 0, 0, 0}

What is the length of this array?
Operations on an Array:

- Input data into the array:

```csharp
For index = 0 To scores.GetUpperBound(0)
    Console.Write("Enter Element at \{0\} : ", index)
    scores(index) = CDb1(Console.ReadLine())
Next index
```
Operations on an Array:

Displaying the cell contents of the array:

```csharp
For index = 0 To scores.GetUpperBound(0)

    Console.WriteLine("Value at \{0\} is \{1\}", index, scores(index))

Next index
```
Operations on an Array:

Displaying the cell contents of the array:

```
For index = 0 To scores.GetUpperBound(0)
    Console.WriteLine("Scores({0}) = {1}" , index, scores(index))
Next index
```
Displaying the cell contents of the array:

```csharp
For index = 0 To scores.GetUpperBound(0)
    Console.WriteLine("The No. {0} element of Scores is {1}", index+1, scores(index))
Next index
```
Dim scores(4) As Double
Dim index As Integer

For index = 0 To scores.GetUpperBound(0)
    scores(index) = 0.0
Next index

For index = 0 To scores.GetUpperBound(0)
    Console.Write("Enter Element at \{0\} : ", index)
    scores(index) = CDb1(Console.ReadLine())
Next index

For index = 0 To scores.GetUpperBound(0)
    Console.WriteLine("Value at \{0\} is \{1\}" , index, scores(index))
Next index