For updated version, please click on
http://ocw.ump.edu.my

COMPUTER PROGRAMMING

ARRAY

by
LIM KAR SING

FACULTY OF CIVIL ENGINEERING & EARTH RESOURCES
UNIVERSITI MALAYSIA PAHANG

limks@ump.edu.my

Creating and Using Arrays

- Declaring an Array Variable
- The Load Event Procedure
- Implicit Array Sizing and Initialization
- Text Files
- Array Methods
- Calculating an Array Value with a Loop

Simple and Array Variables

- A **variable** (or simple variable) is a name to which Visual Basic can assign a single value.
- An **array variable** is a collection of simple variables of the same type to which Visual Basic can efficiently assign a list of values.

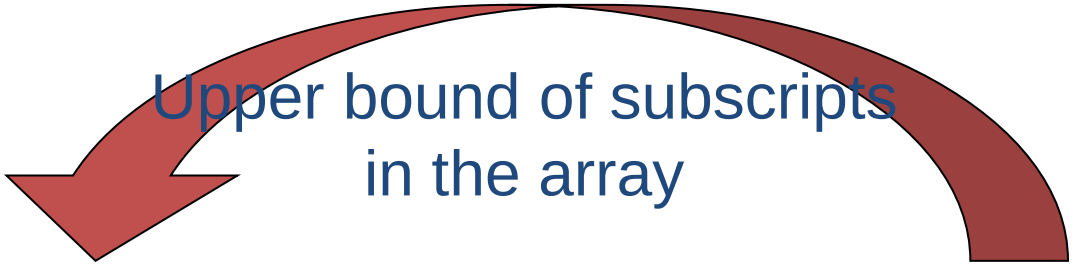
Example

Assuming you want to determine the final marks for 50 students and to display the names of the students with their respective marks.

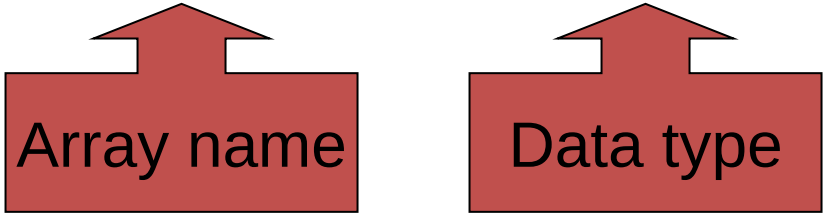
```
Private Sub BtnShow_Click(...) _
                    Handles btnDisplay.Click

   Dim student0 As String, mark0 As Double
   Dim student1 As String, mark1 As Double
   Dim student2 As String, mark2 As Double
   Dim student3 As String, mark3 As Double
   Dim student4 As String, mark4 As Double
   Dim student5 As String, mark5 As Double
   Dim student6 As String, mark6 As Double
    .
    .
```

Using Arrays




```
Dim students(49) As String
Dim marks(49) As Double
```



Putting Values into an Array

- students(0) = “Zakaria"

Read: "students sub zero equals Zakaria"

Which means that the string “Zakaria" is being stored at the first location in the array called *students* because all arrays begin counting at 0.

Array Terminology

- Dim arrayName(n) As *DataType*
- 0 is the **lower bound** of the array
- n is the **upper bound** of the array–the last available subscript in this array
- The number of elements, *n* + 1, is the **size** of the array.

Load Event Procedure

Occurs as the Form loads in memory

```
Private Sub frmMain_Load(...) _
                Handles MyBase.Load
```

The keyword MyBase refers to the form being loaded. This event procedure is a good place to assign values to an array.

Initializing Arrays

Arrays may be initialized when created:

```
Dim arrayName() As DataType =
            {value0, value1, value2, ..., valueN}
```

declares an array having upper bound *N* and assigns *value0* to *arrayName*(0), *value1* to *arrayName*(1), ..., and *valueN* to *arrayName*(*N*).

Example:
```
Dim UniNames() As String =
        {"UniMap", "UMP", "UTHM", "UTeM"}
```

Text Files

- Hold data to be processed by programs.
- Can be created, viewed, and managed by word processors or by the Visual Basic IDE.
- Have the extension txt
- Normally placed in the *bin\Debug* folder in the Solution Explorer.

A Text File Displayed in the Visual Basic IDE

The file (written in notepad) contains the score of Test1.

Test1 Mark.txt | frmArray.vb | frmArray.vb [Design]

```
25
25
15
18
21
13
18
17
15
15
19
20
21
22
24
16
17
18
12
20
```

Array Methods

arrayName.Count	number of elements
arrayName.Max	highest value
arrayName.Min	lowest value
arrayName.First	first element
arrayName.Last	last element

Array Methods (continued)

- The upper bound of *arrayName* is

```
arrayName.Count - 1
```

- ***arrayName.First*** is the same as

```
arrayName(0)
```

Methods for Numeric Arrays

numArrayName.Average	average value of elements
numArrayName.Sum	sum of values of elements

Using Loops Instead of Methods

- In Exercise 29 the greatest and lowest value in a numeric array *ages* is determined.
- The value of the variable *max* and *min* are set to the first element of the array.
- Then a For…Next loop successively examines each element of the array and resets the value of *max* and *min* when appropriate.

For Each Loops

```
For i As Integer = 1 To (ages.Count - 1)
   If ages(i) > max Then
      max = ages(i)
   End If
Next
```

can be replaced with

```
For Each age As Integer In ages
   If age > max Then
      max = age
   End If
Next
```

For Each Loops (continued)

- In the For…Next loop, the counter variable *i* can have any name.
- In the For Each loop, the looping variable *age* can have any name.
- The primary difference between the two types of loops is that in a For Each loop no changes can be made in the values of elements of the array.

Split Method

- Facilitates working with text files.
- Split can convert a string containing comma-separated data into a string array.
- The 0^{th} element of the array contains the text preceding the first comma, the 1^{st} element contains the text between the first and second commas, ..., and the last element contains the text following the last comma.

Split Example

For instance, suppose the string array *employees* has been declared without an upper bound, and the string variable *line* has the value "Bob,23.50,45".

```
employees = line.Split(","c)
```

- sets the size of *employees* to 3
- sets employees(0) = "Bob"
- sets employees(1) = "23.50"
- sets employees(2) = "45"

Split Comments

```
Consider employees = line.Split(","c)
```

- In this statement, the character comma is called the **delimiter** for the Split method, and the letter *c* specifies that the comma has data type Character instead of String
- Any character can be used as a delimiter. If no character is specified, the space character will be used as the delimiter.

Join Function

The reverse of the Split method is the Join function. Join concatenates the elements of a string array into a string containing the elements separated by a specified delimiter.

```
Dim greatLakes() As String = {"Huron",
      "Ontario",
  "Michigan","Erie","Superior"}
Dim lakes As String
lakes = Join(greatLakes, ",")
txtOutput.Text = lakes
```

Output: `Huron,Ontario,Michigan,Erie,Superior`

Passing Arrays to Procedures

- An array declared in a procedure is local to that procedure.
- An entire array can be passed to a Sub or Function procedure.
- The calling statement uses the name of the array without parentheses.
- The header of the Sub or Function procedure uses the name with an empty set of parentheses.