PROGRAMMING LANGUAGE 2 (SPM 3112)

ARRAYS

NOOR AZEAN ATAN
MULTIMEDIA EDUCATIONAL DEPARTMENT
UNIVERSITI TEKNOLOGI MALAYSIA

Topics

- Array Definition
- Array Declaration
- Array Bound & Option Base
- Dynamic Array
- Array Initialisation
- Assessing an array
- Two-dimensional array

Introduction to Arrays

- When we work with more than one item, we will declare more than one variable for each item.
- For example:
- Dim name1, name2, name3 As String
  - name1="Ali"
  - name2="Tan"
  - name3="Samy"

Problem:

- Difficult to process complex data

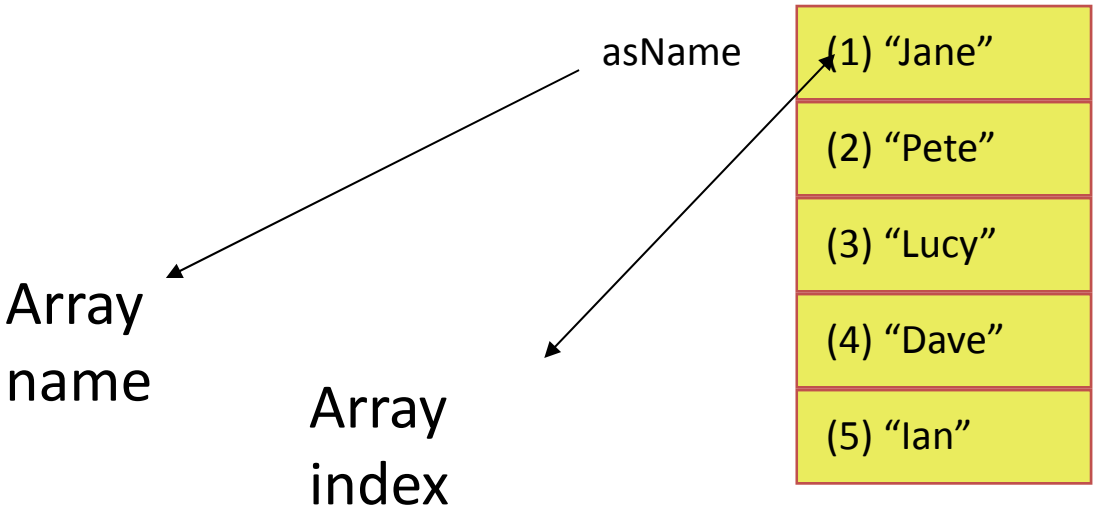
To overcome:

- We can use array to represent these value together under a single name.

For example: Dim name(3) As String

Array Definition

- By definition, array → a list of variables,
  - all with the same data type and name.
- Allows you to refer to these related values → same name with an index or subscript.
- Index → an integer number that identifies array element's position.



Array Declaration

- The general format to declare an array as follow:
  Dim arrayName(bounds) As dataType

- We use the standard keywords used to declare variables:

  Dim - at module, form or procedure level
  Global - at module level
  Static - at procedure level

- We need to say what the size of the array is when we declare it

```
Dim aiCounters(14) As Integer
Dim asNames(5) As String
Static acTicketPrices(5) As Double
Global gafMeasurements(99) As Single
```

- An array variable can be declared in a code module, a form, or a procedure

Why use arrays?

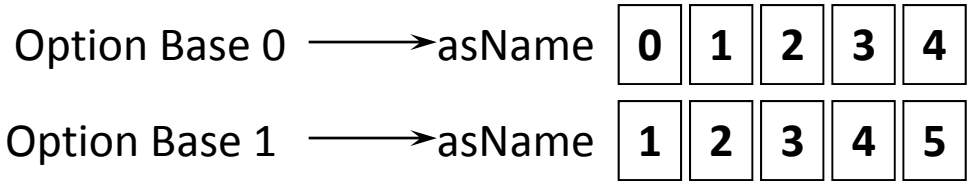
- To store & process LISTS or TABLES of data
- To provide easy access via loops
- To allow processing of groups of data, where the groups must be “remembered”

Array Bound & Option Base

- The bounds are the 'size' of an array
- Arrays have a lower address or lower bound, and an upper address or upper bound

- Option Base can be set to either 0 or 1
  - Option Base 0 'sets the lower bound to 0
  - Option Base 1 'sets the lower bound to 1

- Dim asName(5) As String



Array Bound & Option Base

- declare how many elements we want in our array → VB will set them up using the Option Base setting:

```
Dim asName(5) As String
```

- state explicitly the lower and upper bounds that we want for the array

```
Dim asName(5 To 9) As String
```

Array Bound & Option Base

DIM statement with no lower bound is assumed to start at zero

Dim sTotal(24) as Single

SAME AS

Dim sTotal(0 to 24) as Single

Clearing Array

- To clear a complete array you can use the Erase command:

```
Erase asNames
```

- This resets all fields to their ‘null’ values

Dynamic Array

- **Array size is allocated on demand**
- **We can use ReDim statement to resize array.**
- **ReDim cannot be used to change datatype, the number of dimension and new initialisation value.**
- **Only appear at procedure level.**

In the declaration Section:

```
Dim asNames() As String
```

Within a procedure:

```
ReDim asNames(10)
'--ReDim releases the existing array and creates a new array with the same rank.
or
ReDim Preserve asNames(15)
'—ReDim Preserve, the elements from the existing array arecopied to the new array.
```

Initialisation of Arrays

arrayname[index] = initialiser

Example:

```
Dim sGrade(6) as Integer
Option Base 1
sGrade(1) = 90
sGrade(5) = 92
for index = 2 to 4
   sGrade(index) = 100
next index
sGrade(6) = (sGrade(1) + sGrade(2) )/2
```

Initialisation of Arrays

- If number of initialisers is less than the size of the array
  - the remaining elements are initialised to zero.
- If Number of initialisers > size of array
  - Syntax error

```
Dim x(10) As Integer
x(15) = 10
x(-1) = 25
x(10) = 3
x(0) = 0
```

Invalid element accessed!

Be careful of option base. Only one of these will be valid!

Assessing the value of an array

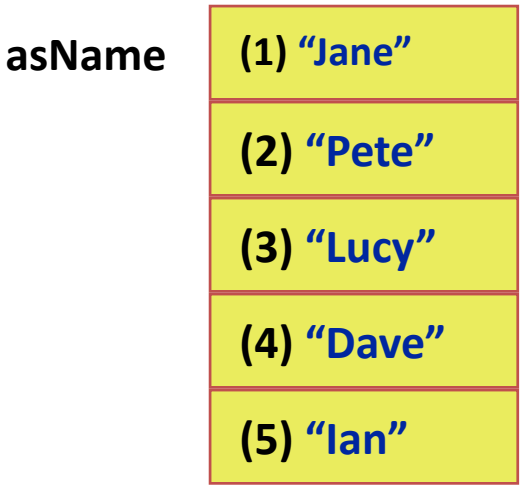
- We need to be able to address the individual elements → in an array
- We use the array name and the element number to access it

Example:

```
Dim asName(5) As String
Option Base 1
    asName(2) = "Pete"

    asName(3) = "Lucy"

    Text1.Text = asName(4)
```



Array Example

```
min = x(0)
For i = 0 To ARRAY_SIZE - 1
    If x(i) < min Then min = x(i)
Next I
```

- Can use code with any number of values
  - Easier to read

Array Example

- Sum 100 numbers

```
For i = 0 to 99
        x(i) = Get_Number()
        sum = sum + x(i)
Next i
MsgBox sum
```

Array Example

- Sum up to 100 numbers or until a 0 is returned.

```
For i = 0 to 99
    x(i) = Get_Number()
    If x(i) = 0 Then
        Exit For
    Else
        sum = sum + x(i)
    End If
Next i
```

Array Example

- Average up to 100 numbers or until a 0 is returned:

```
For i = 0 to 99
        x(i) = Get_Number()
 If x(i) = 0 Then
                Exit For
        Else
                sum = sum + x(i)
        End If
Next i
average = sum / i
```

Finding Array Boundaries

- To find the bounds of a single dimension array we can use:

```
iLowerBound = LBound( asNames )

iUpperBound = UBound( asNames )
```

- To find multi-dimensional bounds:

```
iUpperBound =  UBound( asAddress, 2 )
```

Multi-Dimensional Array

- Visual basic will allow us up to 60 dimensions!
- Do not to use too many dimensions, otherwise you will become confused
- Best idea is only to use multi-dimensional arrays where they map clearly to the real-world

Multi-Dimensional Array

- 1-Dimensional array – has only one subscript
- 2-Dimensional arrays – have two subscripts
  - Analogous to a table
    - 1st subscript represents the "row"
    - 2nd subscript represents the "column"
- All elements in a given array must be of the same type, regardless of dimension

Two-Dimensional Array

Dim x(10, 2) As Integer

- Declares 10 x 2 integers, all referred to by x
- Individual items (*elements*) are accessed as x(*m, n*)
  - x(2, 1), x(7, 0)

X(0, 0)	X(0, 1)
X(1, 0)	X(1, 1)
X(2, 0)	X(2, 1)
X(3, 0)	X(3, 1)
X(4, 0)	X(4, 1)
X(5, 0)	X(5, 1)
X(6, 0)	X(6, 1)
X(7, 0)	X(7, 1)
X(8, 0)	X(8, 1)
X(9, 0)	X(9, 1)

Declaring 2D Array

- Dim iArray (1 to 2, 1 to 5) as Integer
- Dim sGrades (1 to 30, 1 to 10) as Single

☑ Either, an array of numeric grades on 10 items for each of 30 students

OR

☑ An array of grades on 30 items for each of 10 students

Assesing 2D Array

Dim iArray(2,5) as Integer

Option Base 1

- **iArray (1, 2) = 10**
- iArray (2, 5) = iArray(1, 4) + iArray(1, 5)

Dim sGrades (1 to 30, 1 to 10) as Single

- I = 7
  J = 5
  sGrades (I, J) = 93

- x = 1
  y = 2
  iArray (1, x + y) = 15