Topic: 2.2.2 Arrays

An array is a data structure, which allows a set of items of identical data type to be stored together using the same identifier name.

Arrays are declared in a similar way to standard variables, except that the array size and dimensions are included. For example, the following declares an array that reserves five locations in memory and labels these as 'Names':

```
DIM Names(4) As String
```

The five individual locations are Names(0), Names(1), Names(2), Names(3), Names(4).Each data item is called an "element" of the array. To reference a particular element a programmermust use the appropriate index. For example, the following statement assigns data to the 5th element:

```
Names(4) = "Johal"
```

Arrays simplify the processing of similar data. An algorithm for getting five names from the user and storing them in the array Names is shown below:

```
Dim Names(4) As String
For i= 0 to 4
      Input Value
      Names(i)=Value
Next i
```

One-dimensional arrays

A one-dimensional array is a data structure in which the array is declared using a single index and can be visually represented as a list.

The following diagram shows the visual representation of the array `Names(4)`:

Index	Element
[0]	JONES
[1]	ERICSON
[2]	WILLIAMS
[3]	ADAMS
[4]	JOHAL

Topic: 2.2.2 Arrays

Two-dimensional arrays

A two-dimensional array is a data structure in which the array is declared using two indices and can be visually represented as a table.

The following diagram shows the visual representation of an array `Students(4,2)`:

Indexes	[0]	[1]	[2]
[0]	JONES	F	PENN'S
[1]	ERICSON	M	LAMBOURNE
[2]	WILLIAMS	M	PENN'S
[3]	ADAMS	F	CASWALL'S
[4]	JOHAL	M	SWALLOW'S

Each individual element can be referenced by its row and column indices. For example:

```
Students(0,0) is the data item "JONES"
Students(2,1) is the item "M"
Students(1,2) is the item "LAMBOURNE"
```

Initializing an array

Initializing an array is a procedure in which every value in the array is set with starter values – this starting value would typically be "" for a string array, or 0 for a numeric array.

Initialization needs to be done to ensure that the array does not contain results from a previous use, elsewhere in the program.

Algorithm for initializing a one-dimensional numeric array:

```
DIM TestScores(9) As Integer
DIM Index As Integer
FOR Index = 0 TO 9
      TestScores(Index) = 0
NEXT
```

Topic: 2.2.2 Arrays

Algorithm for initializing a two-dimensional string array:

```
DIM Students(4,2) As String
DIM RowIndex, ColumnIndexAs Integer
FOR RowIndex = 0 TO 4
      FOR ColumnIndex = 0 TO 2
            Students(RowIndex,ColumnIndex) = “”
      NEXT
NEXT
```

Serial search on an array

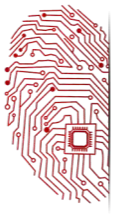
The following pseudo-code can be used to search an array to see if an item X exists:

```
01 DIM Index As Integer
02 DIM Flag As Boolean
03 Index = 0
04 Flag = False
05 Input X
06 REPEAT
07    IF TheArray(Index) = X THEN
08          Output Index
09          Flag = True
10    END IF
11    Index = Index + 1
12 UNTIL Flag = True OR Index > MaximumSizeOfTheArray
```

Note that the variable Flag (line 04 and 09) is used to indicate when the item has been found and stop the loop repeating unnecessarily (line 12 ends the loop if Flag has been set to True).

To complete the search algorithm, some lines should be added, after the loop, to detect the times when the item X was not found in the array:

```
13 IF Flag = False THEN
14 Show Message “Item not found”
15 END IF
```



Topic: 2.2.2 Arrays

Bubble Sort

A bubble sort, a sorting algorithm that continuously steps through a list, swapping items until they appear in the correct order.

Bubble sort is a simple sorting algorithm that works by repeatedly stepping through the list to be sorted, comparing each pair and swapping them if they are in the wrong order. The pass through the list is repeated until no swaps are needed, which indicates that the list is sorted. The algorithm gets its name from the way larger elements "bubble" to the top of the list. It is a very slow way of sorting data and rarely used in industry. There are much faster sorting algorithms out there such as **insertion sort** and **quick sort** which you will meet in A2.

For a step by step animation, visit:

http://en.wikipedia.org/wiki/Bubble_sort#mediaviewer/File:Bubble-sort-example-300px.gif

Step-by-step example

Let us take the array of numbers "5 1 4 2 8", and sort the array from lowest number to greatest number using bubble sort algorithm. In each step, elements written in bold are being compared.

First Pass:

(5 1 4 2 8) (1 5 4 2 8), Here, algorithm compares the first two elements, and swaps them since 5 > 1

(1 5 4 2 8) (1 4 5 2 8), It then compares the second and third items and swaps them since 5 > 4

(1 4 5 2 8) (1 4 2 5 8), Swap since 5 > 2

(1 4 2 5 8) (1 4 2 5 8), Now, since these elements are already in order (8 > 5), algorithm does not swap them.

The algorithm has reached the end of the list of numbers and the largest number, 8, has bubbled to the top. It now starts again.

Second Pass:

(1 4 2 5 8) (1 4 2 5 8), no swap needed

(1 4 2 5 8) (1 2 4 5 8), Swap since 4 > 2

(1 2 4 5 8) (1 2 4 5 8), no swap needed

(1 2 4 5 8) (1 2 4 5 8), no swap needed

Now, the array is already sorted, but our algorithm does not know if it is completed. The algorithm needs one whole pass without any swap to know it is sorted.

Third Pass:

(1 2 4 5 8) (1 2 4 5 8)

(1 2 4 5 8) (1 2 4 5 8)

(1 2 4 5 8) (1 2 4 5 8)

(1 2 4 5 8) (1 2 4 5 8)

Finally, the array is sorted, and the algorithm can terminate.

Pseudocode implementation:

The algorithm can be expressed as:

```
procedurebubbleSort( A : list of sortable items )
do
 swapped = false
 for eachiin 1 to length(A) - 1 inclusive do:
   if A[i-1] > A[i] then
        swap( A[i-1], A[i] )
        swapped = true
   end if
 end for
while swapped
end procedure
```

Topic: 2.2.2 Arrays

Exercise: Bubble Sort

We will now look at an example in Visual Basic using an array of people's heights. The following data set is being passed:

height	
1	98
2	12
3	99
4	54

```
Sub bubbleSort(ByRef height()As integer)
Dim swapped As Boolean
Dim temp As integer
'sort the elements
Do
  Swapped = False
   For Count =1 To MaxSize-1
     If height(Count +1)< height(Count)Then
       Temp = height(Count)
       height(Count)= height(Count +1)
       height(Count +1)= temp
       swapped = True
     EndIf
   Next
Loop Until swapped =False
```

```
'Print out the elements
For Count =1ToMaxSize
   Console.WriteLine(Count &": "& height(Count))
Next
EndSub
```

Topic: 2.2.2 Arrays

Linear Search

The following pseudo code describes a typical variant of linear search, where the result of the search is supposed to be either the location of the list item where the desired value was found; or an invalid location *-1*, to indicate that the desired element does not occur in the list.

For each item in the list:
if that item has the desired value,
stop the search and return the item's *location.*
Return *"-1"*

```
dim items() = {"h","g","a","d","w","n","o","q","l","b","c"}
dim searchItem as string

console.write("What are you searching for: ")
searchItem = console.readline()

For x = 0 to 10
 If items(x) = searchItem Then
  console.writeline("Found item " & searchItem & " at position " & x)
  Exit For
 EndIf
 If x = 10 Then
  console.writeline(-1)
 EndIf
Next
```