



Syllabus Content: 10.2 Arrays

- use the technical terms associated with arrays including upper and lower bound
- select a suitable data structure (1D or 2D array) to use for a given task
- use pseudocode for 1D and 2D arrays (pseudocode will use square brackets to contain the array subscript, for example a 1D array as A[1:n] and a 2D array as C[1:m, 1:n])
- write program code using 1D and 2D arrays
- write algorithms/program code to process array data including:
 - Sort using a bubble sort
 - Search using a linear search

An array is a special variable that has one name, but can store multiple values. Each value is stored in an element pointed to by an index.

The first element in the array has index value 0, the second has index 1, etc

One Dimensional Arrays

A one dimensional array can be thought as a list. An array with 10 elements, called names, can store 10 names and could be visualized as this: **Lower bound of ARRAY can start from 0 or 1**

| Index | Name |
|-------|----------|
| 0 | Majid |
| 1 | Tahir |
| 2 | Naila |
| 3 | Hassan |
| 4 | Adil |
| 5 | Ali |
| 6 | Osman |
| 7 | Abdullah |
| 8 | Haider |
| 9 | Hamza |

| Index | Name |
|-------|----------|
| 1 | Majid |
| 2 | Tahir |
| 3 | Naila |
| 4 | Hassan |
| 5 | Adil |
| 6 | Ali |
| 7 | Osman |
| 8 | Abdullah |
| 9 | Haider |
| 10 | Hamza |

Arrays (One-dimensional arrays)

In order to use a one-dimensional array in a computer program, you need to consider:

- What the array is going to be used for, so it can be given a meaningful name
- How many items are going to be stored, so the size of the array can be determined.
- What sort of data is to be stored, so that the array can be the appropriate data type.

DECLARATION of Blank Array with 10 slots:

Pseudocode:

```
DECLARE names[10]: STRING
```

PYTHON Code

```
name [ ]
```

VB code example:

```
Dim names(9) As String
```





Entering Values in One-Dimension Array

```

BEGIN
DECLARE count : Integer
DECLARE name [5] : String      // for declaring 5 elements in ARRAY
DECLARE marks [5] : Integer

    FOR count = 1 to 5          // for inputting 5 names and grades
        PRINT ("Enter Name "& count)
        INPUT name (count)
        PRINT ("Enter grade for "& name(count))
        INPUT marks (count)
    NEXT count

    FOR count 1 to 5 // for displaying 5 names and grades
        PRINT (name (count) & "has marks " & marks(count))
    NEXT count
END

```

PYTHON Code:

```

name = []
marks = []
for count in range(5):
    name.append (str(input("Enter name: ")))
    marks.append (int(input("Enter marks ")))
print("Name ", name, " scored ", marks)

```

1D Array.py - C:/Users/majid/AppData/Local/Programs/Python/

File Edit Format Run Options Window Help

```

name = []
marks = []
for count in range(5):
    name.append (str(input("Enter name: ")))
    marks.append (int(input("Enter marks ")))
print("Name ", name, " scored ", marks)

```

OUTPUT screen

IDLE Shell 3.12.5

File Edit Shell Debug Options Window Help

Python 3.12.5 (tags/v3.12.5:ff3bc82, Aug 6 2024, 20:45:27) [MSC v.1940 64 bit (AMD64)] Type "help", "copyright", "credits" or "license()" for more information.

```

>>>
== RESTART: C:/Users/majid/AppData/Local/Programs/Python/Python312/1D Array.py =
Enter name: Ali
Enter marks 11
Enter name: Hassan
Enter marks 22
Enter name: Naila
Enter marks 33
Enter name: Majid
Enter marks 44
Enter name: Jimmy
Enter marks 55
Name ['Ali', 'Hassan', 'Naila', 'Majid', 'Jimmy'] scored [11, 22, 33, 44, 55]
>>>

```

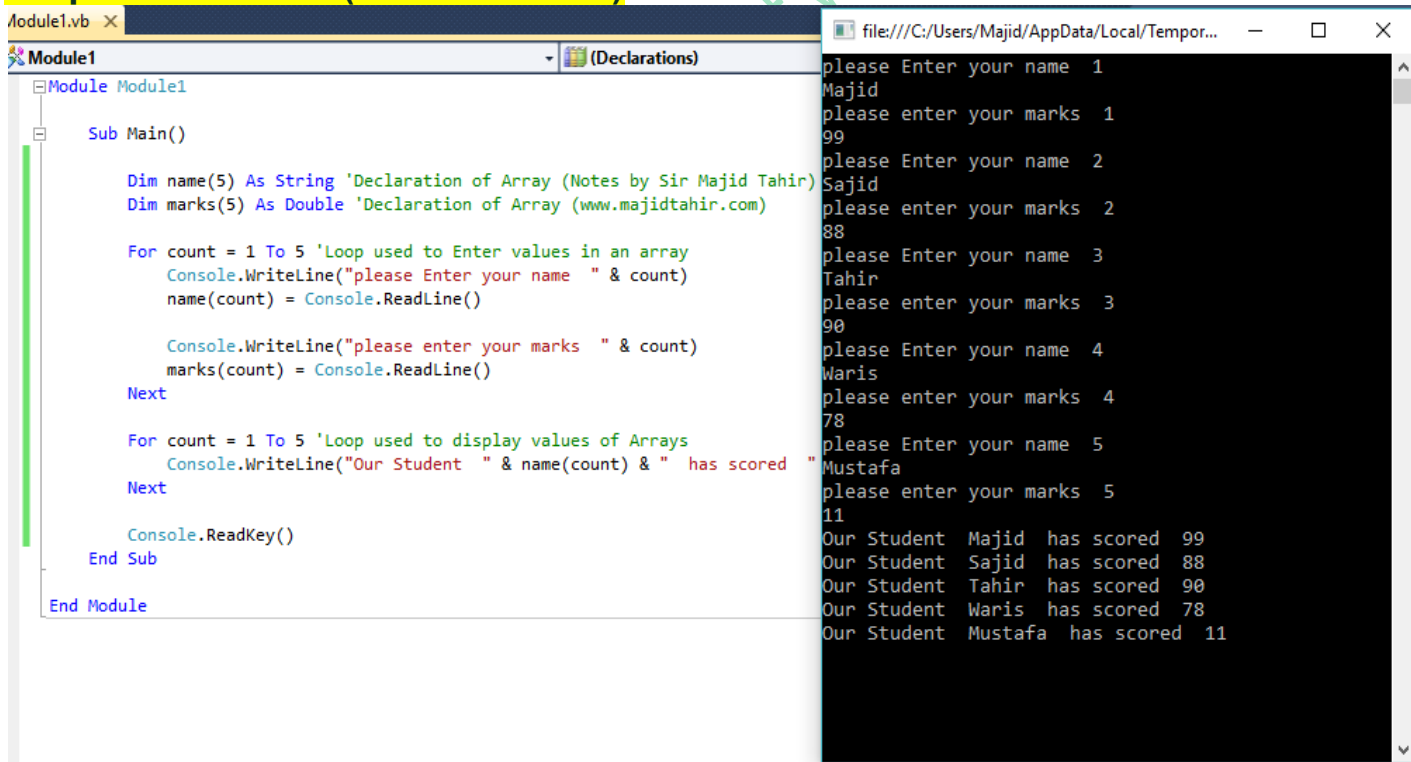




VB Code in Console Mode

```
Dim name(5) As String
Dim marks(5) As Integer
  For count = 1 To 5
    Console.WriteLine("input name " & count)
    name(count) = Console.ReadLine()
    Console.WriteLine("input marks " & count)
    marks(count) = Console.ReadLine()
    While marks(count) > 100 Or marks(count) < 0
      Console.WriteLine("Re-Enter marks" & count)
      marks(count) = Console.ReadLine()
    End While
  Next
  For count = 1 To 5
    Console.WriteLine(name(count) & " scored: " & marks(count))
  Next
```

Output of VB code (Console mode)



**Python** One-dimensional array with values in it.

```
num = [1, 2, 3]
num.append(4)
num.extend([5, 6])
print(num) # will print this in output [1, 2, 3, 4, 5, 6]
```

Another example of One-Dimensional Array

Module Module1

Sub Main()

Dim count As Integer

Dim name(4) As String

Dim marks(4) As Integer

Dim gender(4) As String

For count = 0 To 4

Console.WriteLine("please enter your name" & count)

name(count) = Console.ReadLine()

Console.WriteLine("please enter your gender" & count)

gender(count) = Console.ReadLine()

Console.WriteLine("please enter your marks" & count)

marks(count) = Console.ReadLine()

Next count

For count = 0 To 4

Console.WriteLine("your name is : " & name(count))

Console.WriteLine("your gender is : " & gender(count))

Console.WriteLine("your marks are : " & marks(count))

Next count

Console.ReadKey()

End Sub

End Module

Multi-Dimensional Arrays or Two dimensional Arrays (2D Array):

A multi-dimensional array can be thought of as a table, each element has a row and column index. Following example declares a two-dimensional array called `table` with 3 rows and 4 columns and would be declared in **PseudoCode** as follows:

```
DECLARE table(3, 4) : INTEGER
```

Visual Basic(Console mode)

```
Dim table(3, 4) : As Integer
```

Python Code

```
row, col = 3, 4
```

```
table = [[0 for x in range(row)] for y in range(col)]
```

```
# Creates a list containing 5 lists, each of 8 items, all set to 0
```





```

IDLE Shell 3.7.5 Two Dimensional Array.py - C:/Users/majid/AppData/Local/Programs/Python/Python312/Tw...
File Edit Shell File Edit Format Run Options Window Help
Python # Creates a list containing 5 lists, each of 8 items, all set to 0
AMD64] row, col = 3, 4
Type "h table = [[0 for x in range(row)] for y in range(col)]
>>> print(table)
= RESTA
l Array.py
[[0, 0, 0], [0, 0, 0], [0, 0, 0], [0, 0, 0]]
>>>
Ln: 4 Col: 12

```

PSEUDOCODE Example of Two-Dimension Array

BEGIN

```

DECLARE table(3, 4) : Integer
FOR row = 1 To 3
  FOR column = 1 To 4
    PRINT("Please Input Value in Row: ",row, "column : ", column)
    INPUT table(row, column)
  NEXT
NEXT
FOR row = 1 To 3
  FOR column = 1 To 4
    PRINT ("Row = " & row & "column = " & column & "has Value")
    PRINT (table(row, column))
  NEXT
NEXT
END

```

VB Code Example of Two-Dimension Array

```

Sub Main()
  Dim table(2, 3) As Integer
  For row = 0 To 2
    For column = 0 To 3
      Console.WriteLine("Please Input Value in Row: " & row & "column : " & column)
      table(row, column) = Console.ReadLine()
    Next
  Next
  Console.Clear()
  For row = 0 To 2
    For column = 0 To 3
      Console.WriteLine("Row = " & row & "column = " & column & "has Value")
      Console.WriteLine(matrix(row, column))
    Next
  Next
  Console.ReadKey()
End Sub

```





Multi-Dimensional Arrays:

A multi-dimensional array can be thought of as a table, each element has a row and column index.

Following example declares a two-dimensional array called matrix and would be declared by

```
Dim matrix(2,3) As Integer
```

Usually we refer to the first dimension as being the rows, and the second dimension as being the columns.

| index | 0 | 1 | 2 | 3 |
|-------|---|---|---|---|
| 0 | A | B | C | D |
| 1 | E | F | G | H |
| 2 | I | J | K | L |

The following statements would generate the following

```
Console.WriteLine(matrix(0, 0))
```

Would display A

```
Console.WriteLine(matrix(2, 1))
```

Would display J

```
Console.WriteLine("first row, first column : " & matrix(2, 3))
```

Would display first row, first column : L

VB Code for 2-D Array is:

```

Module1 (Declarations)
Module Module1
Sub Main() ' Notes by Sir Majid Tahir ( Download free at www.majidtahir.com)
Dim table(3, 4) As Integer ' DECLARING TWO-DIMENSIONAL ARRAY
For row = 1 To 3 ' Variable Row is used to use in loop for rows
For column = 1 To 4 ' Variable column is used to use in Columns
Console.WriteLine("please Enter data in row= " & row & " column = " & column)
table(row, column) = Console.ReadLine()
Next
Next
For row = 1 To 3
For column = 1 To 4
Console.WriteLine("Data is Row= " & row & " column = " & column & " = " & table(row, column))
Next
Next
Console.ReadKey()
End Sub
End Module

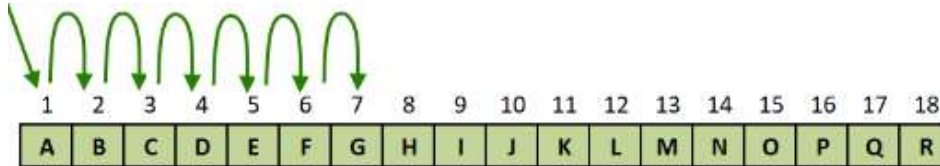
```



Searching & Sorting Algorithms in an ARRAY

Linear Search:

Linear search is a method of searching a list in which each element of an array is checked in order, from the lower bound to the upper bound, until the item is found, or the upper bound is reached.



Searching G in the sorted list via Linear search.

The pseudocode linear search algorithm and identifier table to find if an item is in the 1D array(myList) is given below.

```

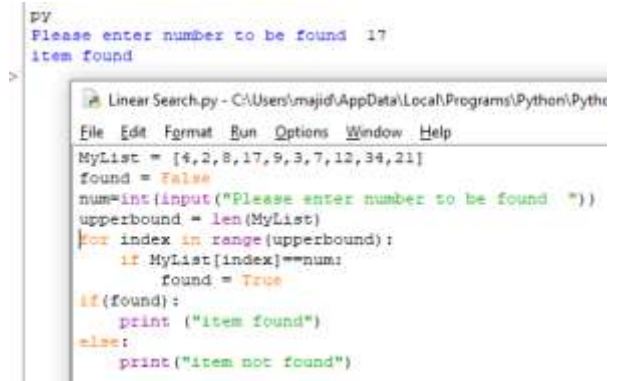
DECLARE count, num As Integer
DECLARE found As Boolean = False
//Creating array to search item (Free notes @ www.majidtaahir.com)
DECLARE Mylist() As Integer = {4, 2, 8, 17, 9, 3, 7, 12, 34, 21}
OUTPUT ("Please enter any integer to be checked in List")
INPUT num
For count = 1 To 10
    If num = Mylist(count) Then
        found = True
    End If
Next count

If found = True Then
    OUTPUT ("Item Found = ", num)
Else
    OUTPUT ("Item Found is unsuccessful")
End If
    
```

PYTHON Linear Search Algorithm

```

MyList = [4,2,8,17,9,3,7,12,34,21]
found = False
num = int(input("Please enter number to be found")) OUTPUT
upperbound = len(MyList)
for index in range(upperbound):
    if MyList[index]==num:
        found = True
if(found):
    print ("item found")
else:
    print("item not found")
    
```



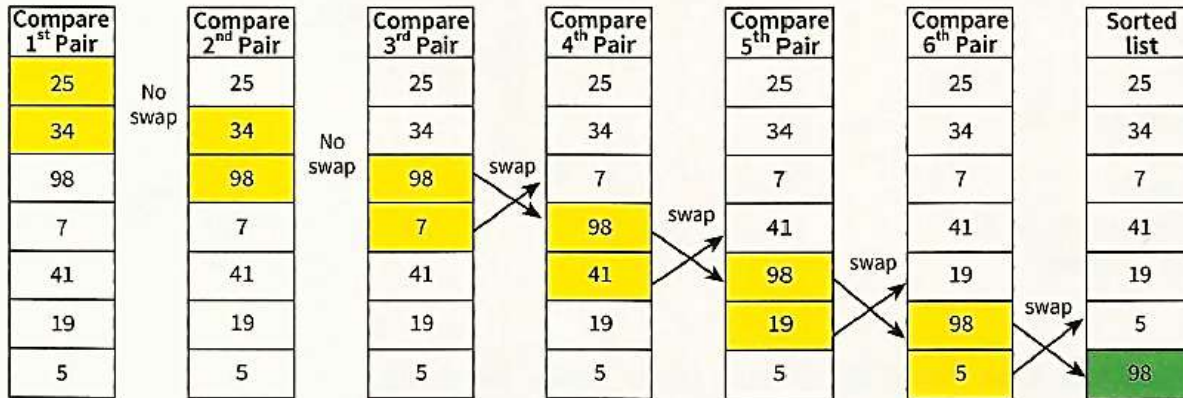
```

PY
Please enter number to be found 17
item found

Linear Search.py - C:\Users\majid\AppData\Local\Programs\Python\Pytho
File Edit Format Run Options Window Help
MyList = [4, 2, 8, 17, 9, 3, 7, 12, 34, 21]
found = False
num=int(input("Please enter number to be found: "))
upperbound = len(MyList)
for index in range(upperbound):
    if MyList[index]==num:
        found = True
if(found):
    print ("item found")
else:
    print("item not found")
    
```

Bubble Sort:

Bubble sort is a sorting algorithm that compares each value in the list with the next value, and swaps the value if not in order. Algorithm is repeated many times in loop to arrange all values in order. We call it **Bubble Sort** because small values rise to the top slowly like bubbles rise in water. After every sorting from start to end of list, biggest value goes at the bottom.



When we have completed the first pass through the entire array, the largest value is in the correct position. Swapping ends the working down the array. We need to work through the array again and again. After each pass through the array the next largest value will be in its correct position, as shown in Figure below.

| Original list | After pass 1 | After pass 2 | After pass 3 | After pass 4 | After pass 5 | After pass 6 |
|---------------|--------------|--------------|--------------|--------------|--------------|--------------|
| 25 | 25 | 25 | 7 | 7 | 7 | 5 |
| 34 | 34 | 7 | 25 | 19 | 5 | 7 |
| 98 | 7 | 34 | 19 | 5 | 19 | 19 |
| 7 | 41 | 19 | 5 | 25 | 25 | 25 |
| 41 | 19 | 5 | 34 | 34 | 34 | 34 |
| 19 | 5 | 41 | 41 | 41 | 41 | 41 |
| 5 | 98 | 98 | 98 | 98 | 98 | 98 |

Figure 11.13 States of the array after each pass

In effect we perform a loop within a loop, a nested loop. This method is known as a **bubble sort**. The name comes from the fact that smaller values slowly rise to the top, like bubbles in a liquid.

KEY TERMS

Bubble sort: a sort method where adjacent pairs of values are compared and swapped



Bubble sort PSEUDOCODE:

```

DECLARE MyList [10] : INTEGER = {4,2,8,17,9,3,7,12,34,21}
DECLARE count, temp, top : INTEGER
DECLARE swap : BOOLEAN = False
top = LENGTH (MyList())
WHILE top <> 0
  FOR count = 1 to (top-1)//(top-1)for loop to run till second last value
    IF Mylist(count)> Mylist(count + 1)//compare second last with last value
      THEN
        temp = Mylist(count)
        Mylist(count) = Mylist(count + 1)
        Mylist(count + 1) = temp
        swap = True
      End If
    top = top-1
  NEXT count
END WHILE

```

Alternate sample program of (Bubble sort)

```

DECLARE myList : ARRAYS[0:8] OF INTEGER = [4,2,8,17,9,3,7,12,34,21]
DECLARE upperBound, lowerbound, count, temp, top : INTEGER
DECLARE swap : BOOLEAN

```

```

upperBound ← LENGTH(MyList)
lowerBound ← 0
top ← upperBound

```






```

REPEAT
  FOR index = lowerBound TO (top - 1)
    Swap ← FALSE
    IF MyList [count] > MyList [count + 1]
      THEN Temp ← MyList[count]
        MyList[count] ← MyList[count + 1]
        MyList[count + 1] ← Temp
        swaps ← TRUE
      END IF
    NEXT
  top ← top - 1
UNTIL (NOT swap) OR (top = 0)

```



References:

-  Computer Science by David Watson & Helen Williams
-  Visual Basic Console Cook Book
-  Computer Science AS and A level by Sylvia Langfield and Dave Duddell
-  <https://www.sitesbay.com/javascript/javascript-looping-statement>
-  <http://wiki.jikexueyuan.com/project/lua/if-else-if-statement.html>

Computer Sc. (9618) with Majid Tahirat www.majidtahir.com

