What is an Array?

Array is a homogeneous aggregate of data elements.

Array is a collection of objects of the same type.

Array is a collection of similar variables which are identified under the same name.

Why use an Array?

It is often the case when writing programs that one needs a large number of storage locations. Ordinary variables have to be declared individually, and are not well suited for this purpose.

Example: If we wish to have 5 variables to hold test scores, one might make the following declaration:

```plaintext
DIM Score1, Score2, Score3, Score4, Score5 AS Double
```

Imagine what would happen if we had to deal with 100 test scores! What if you had to create 1000?

Arrays make life a lot easier.

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

! Syntax:

\[
\text{Dim ArrayName(UpperSubscript) as DataType}
\]
or

\[
\text{Dim ArrayName() as DataType = \{InitialValueList\}}
\]

! Creating an Array:

Dim TestScore(10) As Double  'This declaration creates 10 storage locations in which one may put floating point values.

# Creates 10 “double” variables under the name TestScore.

# In order to access each variable (array element) we must use an array index.

<p>| | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>Array</td>
<td>Value to be placed in the array element</td>
</tr>
<tr>
<td>Name</td>
<td>Index</td>
</tr>
</tbody>
</table>

| 2 | 5 |   | 9 |   |   |   |   |   |   |

0 1 2 3 4 5 6 7 8 9 10
# The individual array members or cells of the array defined above are denoted by

`TestScore(0), TestScore(1), TestScore(2), .. .TestScore(10)`.

# The values 0, 1, 2, .. are known as array indexes and are used to access the value of the elements (cells) within the array.

For example to retrieve the element in location 2 in the above `TestScore` array, we perform the following:

```vbnet
DIM Test2 AS Double
Test2 = TestScore(2)
```

# All members of a single array must be the same type.
Array Properties:

# Array Length

MyArray.Length

Returns the length of the array.

'looking at array properties

Dim Score(10) As Double
Console.WriteLine("Array length = {0}", Score.Length)

Array length = 11

# Array Upper Bound

MyArray.GetUpperBound(0)

Returns the index of the last element in the array.

The argument indicates the dimension of the array. (0 indicates a one dimensional array)

'looking at array properties

Dim Score(10) As Double
Console.WriteLine("Array upper bound = {0}", Score.GetUpperBound(0))

Array upper bound = 10
**Operations on an Array:**

### Initializing the scores:

```vbnet
Dim Index As Integer
Dim Score(10) As Double 'This declaration makes available 10 storage locations (Actually 11) in which to put floats.

For Index = 0 To 10
    Score(Index) = 0.0
Next Index
```

### Inserting data in the array:

```vbnet
' Read some new values into the array locations
For Index = 0 To 10
    Console.Write("Enter array element ")
    Console.Write(Index)
    Console.Write(":")
    Score(Index) = CDbl(Console.ReadLine())
Next Index
```

### Displaying the cell contents of the array:

```vbnet
' Now print the values in the array.
For Index = 0 To 10
    Console.WriteLine(Score(Index))
Next Index
```

### Searching an Array: (Linear Search)

```vbnet
' Search for a value in an array
Dim value As Double
Console.Write("Enter the value to search for:")
value = Console.ReadLine() 'value to search for

For Index = 0 To 10
    If Score(Index) = value Then
        Console.WriteLine("location {0}, Found it!", Index)
    Else
        Console.WriteLine("Location {0}, Not Found...", Index)
    End If
Next Index
```
Sorting an Array

Private Sub BubbleSort(ByRef TheArray() As Integer)
    Dim Pass, Index, Hold As Integer
    For Pass = 1 To TheArray.GetUpperBound(0)
        For Index = 0 To TheArray.GetUpperBound(0) - 1
            If TheArray(Index) > TheArray(Index + 1) Then
                'swap(TheArray(Index), TheArray(Index+1))
                Hold = TheArray(Index)
                TheArray(Index) = TheArray(Index + 1)
                TheArray(Index + 1) = Hold
            End If
        Next Index
    Next Pass
End Sub
# Binary Search

' BinarySearch()
' --------------
' Perform a Binary Search for a value in an array and return
' the index, or return -1 to indicate that value was not found.
' NOTE: A binary Search requires the array to be sorted.

Private Function BinarySearch(ByVal value As Integer,
    ByRef TheArray() As Integer) As Integer

    Dim Low, High, Middle As Integer
    Low = 0
    High = TheArray.GetUpperBound(0)
    Do While Low <= High
        Middle = (Low + High) \ 2
        If value = TheArray(Middle) Then 'It’s a match!!
            Return (Middle)
        ElseIf value < TheArray(Middle) Then 'Search the low end of array
            High = Middle - 1
        Else
            Low = Middle + 1
        End If
    Loop
    Return (-1) 'return -1 to indicate the value was not found

End Function
Module Module1
Sub Main()
    Dim Index As Integer
    Dim Score(10) As Double
    'This declaration makes available 10 storage
    'locations (Actually 11) in which to put floats.

    'looking at array properties
    Console.WriteLine("Array length = {0}", Score.Length)
    Console.WriteLine("Array upper bound = {0}", Score.GetUpperBound(0))

    'To initialize the locations of the array
    For Index = 0 To 10
        Score(Index) = 0.0
    Next Index

    'Now print the values in the array.
    For Index = 0 To 10
        Console.WriteLine(Score(Index))
    Next Index

    'Read some new values into the array locations
    For Index = 0 To 10
        Console.Write("Enter array element ")
        Console.Write(Index)
        Console.Write(": ")
        Score(Index) = Console.ReadLine()
    Next Index

    'Now print the values again
    For Index = 0 To 10
        Console.WriteLine(Score(Index))
    Next Index

    'Search for a value in an array
    Dim value As Double
    Console.Write("Enter the value to search for:")
    value = Console.ReadLine() 'value to search for
    For Index = 0 To 10
        If Score(Index) = value Then
            Console.WriteLine("location {0}, Found it!", Index)
        Else
            Console.WriteLine("Location {0}, Not Found..", Index)
        End If
    Next Index
End Sub
End Module
Complete Example 2:

' One-Dimensional Array and some of its operations

Module Module1

Sub Main()
    Dim Location As Integer
    Dim Score(9) As Integer  'make 10 Integer storage locations

    'looking at array properties
    Console.WriteLine("Array length = {0}", Score.Length)
    Console.WriteLine("Array upper bound = {0}", Score.GetUpperBound(0))

    Initialize(Score)
    Print(Score)
    InitializeRandom(Score)
    Print(Score)

    'ReadData(Score)
    'Print(Score)

    Location = Search(Score)  'call the first search() function (One parameter)
    Console.WriteLine("Found value at Location: {0}", Location)

    Console.WriteLine("Sort the Array")
    BubbleSort(Score)
    Print(Score)

    Location = Search(5, Score)  'call the second search() function (Two Parameters)
    Console.WriteLine("Found 5 at Location: {0}", Location)

    Location = BinarySearch(0, Score)
    Console.WriteLine("Found 0 at Location: {0}", Location)

    Console.ReadLine()
End Sub
' Initialize()
' ------------
'to initialize the locations of the array to 0
Private Sub Initialize(ByRef TheArray() As Integer)
    Dim Index As Integer
    For Index = 0 To TheArray.GetUpperBound(0)
        TheArray(Index) = 0
    Next Index
End Sub

' InitializeRandom()
' ------------------
'to initialize the locations of the array to 0
Private Sub InitializeRandom(ByRef TheArray() As Integer)
    Dim Index As Integer
    Randomize() ' Seed the random number generator
    For Index = 0 To TheArray.GetUpperBound(0)
        TheArray(Index) = CInt(Rnd() * 10) ' return a number in the range 0 to 10
    Next Index
End Sub

' Print()
' -------
'print the values in the array.
Private Sub Print(ByVal TheArray() As Integer)
    Dim Index As Integer
    Console.WriteLine("-----------------------------")
    For Index = 0 To TheArray.GetUpperBound(0)
        Console.WriteLine(TheArray(Index))
    Next Index
    Console.WriteLine("-----------------------------")
End Sub
Private Sub ReadData(ByRef TheArray() As Integer)

    Dim Index As Integer
    For Index = 0 To TheArray.GetUpperBound(0)
        Console.Write("Enter array element ")
        Console.Write(Index)
        Console.Write(":")
        TheArray(Index) = Console.ReadLine()
    Next Index
End Sub

Private Function Search(ByRef TheArray() As Integer) As Integer

    Dim Index As Integer
    Dim value As Double
    Console.Write("Enter the value to search for:")
    value = Console.ReadLine() 'value to search for
    For Index = 0 To TheArray.GetUpperBound(0)
        If TheArray(Index) = value Then
            Return (Index)
        End If
    Next Index
    Return (-1) 'return -1 to indicate the value was not found
End Function
Overloaded Search Function (Two Parameters)

Search for a value in an array and return the index, or return -1 to indicate that value was not found.

Private Function Search(ByVal value As Integer, ByRef TheArray() As Integer) As Integer
    Dim Index As Integer

    For Index = 0 To TheArray.GetUpperBound(0)
        If TheArray(Index) = value Then
            Return (Index)
        End If
    Next Index

    Return (-1) 'return -1 to indicate the value was not found
End Function
' Bubble Sort()
'  -----------------
Private Sub BubbleSort(ByRef TheArray() As Integer)
    Dim Pass, Index, Hold As Integer
    For Pass = 1 To TheArray.GetUpperBound(0)
        For Index = 0 To TheArray.GetUpperBound(0) - 1
            If TheArray(Index) > TheArray(Index + 1) Then
                'swap(TheArray(Index), TheArray(Index+1))
                Hold = TheArray(Index)
                TheArray(Index) = TheArray(Index + 1)
                TheArray(Index + 1) = Hold
            End If
        Next Index
    Next Pass
End Sub
' BinarySearch()
' --------------
' Perform a Binary Search for a value in an array and return
' the index, or return -1 to indicate that value was not found.
' NOTE: A binary Search requires the array to be sorted.
Private Function BinarySearch(ByVal value As Integer, ByRef TheArray() As Integer) As Integer
    Dim Low, High, Middle As Integer
    Low = 0
    High = TheArray.GetUpperBound(0)

    Do While Low <= High
        Middle = (Low + High) \ 2
        If value = TheArray(Middle) Then 'It’s a match!!
            Return (Middle)
        ElseIf value < TheArray(Middle) Then 'Search the low end of array
            High = Middle - 1
        Else
            Low = Middle + 1
        End If
    Loop

    Return (-1) 'return -1 to indicate the value was not found
End Function

End Module

Try writing sub-programs to accomplish the following:

private function Sum(ByVal TheArray()) as integer
private function Average(ByVal TheArray()) as double
private function Min(ByVal TheArray()) as integer
private function Max(ByVal TheArray()) as integer
private function Count(ByVal TheArray()) as integer
private function stddev(ByVal TheArray()) as double
## Alternative Methods for Declaring an Array:

<table>
<thead>
<tr>
<th>Static allocation at compile time:</th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>Dim MyArray(10) as Integer</td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Dynamic allocation at runtime:</th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>Dim MyArray as Integer()</td>
<td></td>
</tr>
<tr>
<td>MyArray = New Integer(ArraySize)</td>
<td>{initializer list}</td>
</tr>
<tr>
<td>MyArray = New Integer(5) {10,20,30,40, 50}</td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Resizing or ReDimensioning the Array:</th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>Dim MyArray() as Integer</td>
<td></td>
</tr>
<tr>
<td>ReDim MyArray(10) as Integer</td>
<td></td>
</tr>
<tr>
<td>' Useful in some occasions. (See declaring arrays within structures.)</td>
<td></td>
</tr>
</tbody>
</table>
Multi-Dimensional Arrays:

# Arrays can have more than one dimension

# Like a table of values

# You must specify the boundaries for each dimension using subscripts

# Syntax:

```
Dim ArrayName(HighestRowSubscript, HighestColumnSubscript) as Datatype
```

or

```
Dim ArrayName( , ) as Datatype = {ListOfValues}
```

# Example-1: Two Dimensional Array

```
Dim ScoreBoard(1,8) As Integer  'This declaration makes available 18 storage locations in which to put Integers.
```

<p>| | | | | | | | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>2</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>1</td>
<td>5</td>
<td>1</td>
<td>2</td>
<td>3</td>
<td>4</td>
<td>5</td>
<td>6</td>
</tr>
</tbody>
</table>

# Creates \textbf{18 (2 \times 9) Integer variables} under the name \textbf{ScoreBoard}.

# In order to access each variable (array element) we must use two array indexes.

Score (0,0) = 2  
Score (1,0) = 5  
Score (1,4) = 9  

\begin{tabular}{l l}
Row & Column \\
\hline
0 &   \\
1 & 0 \\
   & 1 \\
   & 2 \\
   & 3 \\
   & 4 \\
   & 5 \\
   & 6 \\
   & 7 \\
   & 8 \\
\end{tabular}
**Example -2: Two Dimensional Array**

<table>
<thead>
<tr>
<th>Dim strName(2, 3) as String</th>
</tr>
</thead>
<tbody>
<tr>
<td>Results in an array of 12 elements:</td>
</tr>
<tr>
<td>3 rows: 0, 1, 2, 3</td>
</tr>
<tr>
<td>4 columns: 0, 1, 2, 3, 4</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Dim strName(, ) as String =</th>
</tr>
</thead>
<tbody>
<tr>
<td>{{&quot;Jim&quot;, &quot;Mary&quot;, &quot;Sam&quot;, &quot;Sean&quot;},</td>
</tr>
<tr>
<td>{{&quot;Tom&quot;, &quot;Sue&quot;, &quot;Fred&quot;, &quot;Paul&quot;},</td>
</tr>
<tr>
<td>{{&quot;Tim&quot;, &quot;Al&quot;, &quot;Bob&quot;, &quot;Pete&quot;},</td>
</tr>
<tr>
<td>{{&quot;Joy&quot;, &quot;Wes&quot;, &quot;Kim&quot;, &quot;Beth&quot;}}</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Dim ScoreBoard(9, 4) As Integer</th>
</tr>
</thead>
<tbody>
<tr>
<td>'Make a 10x5 storage locations</td>
</tr>
</tbody>
</table>
Operations on a Two-Dimensional Array:

# Initializing the ScoreBoard:

```vbnet
Dim ScoreBoard(9, 4) As Integer  'Make a 10x5 storage locations

' Initialize()
' ------------
' to initialize the locations of the array to 0
Private Sub Initialize(ByRef TheArray(,) As Integer)
    Dim Row, Col As Integer
    For Row = 0 To TheArray.GetUpperBound(0)
        For Col = 0 To TheArray.GetUpperBound(1)
            TheArray(Row, Col) = 0
        Next Col
    Next Row
End Sub
```

# Displaying the cell contents of the array:

```vbnet
' Print()
' -------
' print the values in the array.
Private Sub Print(ByVal TheArray(,) As Integer)
    Console.WriteLine("-------------------------------")
    Dim Row, Col As Integer
    For Row = 0 To TheArray.GetUpperBound(0)
        For Col = 0 To TheArray.GetUpperBound(1)
            Console.Write("{0,4}", TheArray(Row, Col))
        Next Col
    Next Row
    Console.WriteLine()
    Console.WriteLine("-------------------------------")
End Sub
```
# Searching the 2D Array:

' Search Function
' -------------------
' Search for a value in an array and print the row and
column in which it was found.
' return -1 to indicate that value was not found.

Private Function Search(ByRef TheArray(,) As Integer) As Integer
    Dim Row, Col As Integer
    Dim value As Double

    Console.Write("Enter the value to search for:")
    value = Console.ReadLine() 'value to search for

    For Row = 0 To TheArray.GetUpperBound(0)
        For Col = 0 To TheArray.GetUpperBound(1)
            If TheArray(Row, Col) = value Then
                Console.WriteLine("Value found: Row = {0}, Col = {1}",
                                   Row, Col)
            End If
        Next Col
    Next Row

    Return (-1) 'return -1 to indicate the value was not found
End Function
Complete 2D Example:

' Two-Dimensional Array and some of its operations

Module Module1

Sub Main()
    Dim Location As Integer
    Dim ScoreBoard(9, 4) As Integer 'Make a 10x5 storage locations

    'looking at array properties
    Console.WriteLine("Array length = {0}", ScoreBoard.Length)
    Console.WriteLine("Array upper bound (1st Dimension) = {0}",
                      ScoreBoard.GetUpperBound(0))
    Console.WriteLine("Array upper bound (2st Dimension) = {0}",
                      ScoreBoard.GetUpperBound(1))

    Initialize(ScoreBoard)
    Print(ScoreBoard)
    InitializeRandom(ScoreBoard)
    Print(ScoreBoard)

    Search(ScoreBoard)

    Console.ReadLine()
End Sub

' Initialize()
' -----------
' to initialize the locations of the array to 0
Private Sub Initialize(ByRef TheArray(,) As Integer)
    Dim Row, Col As Integer

    For Row = 0 To TheArray.GetUpperBound(0)
        For Col = 0 To TheArray.GetUpperBound(1)
            TheArray(Row, Col) = 0
        Next Col
    Next Row
End Sub
' InitializeRandom()
' ------------------
'to initialize the locations of the array to 0
Private Sub InitializeRandom(ByRef TheArray(,) As Integer)

    Randomize() ' Seed the random number generator

    Dim Row, Col As Integer

    For Row = 0 To TheArray.GetUpperBound(0)
        For Col = 0 To TheArray.GetUpperBound(1)
            TheArray(Row, Col) = CInt(Rnd() * 10) ' return a number
                                                   ' in the range 0 to 10

            Next Col
        Next Row

End Sub

' Print()
' -------
'print the values in the array.
Private Sub Print(ByVal TheArray(,) As Integer)

    Console.WriteLine("-------------------------------")

    Dim Row, Col As Integer

    For Row = 0 To TheArray.GetUpperBound(0)
        For Col = 0 To TheArray.GetUpperBound(1)
            Console.Write("{0,4}", TheArray(Row, Col))

            Next Col
        Next Row

    Console.WriteLine("-------------------------------")

End Sub
### OUTPUT

Array length = 50
Array upper bound (1st Dimension) = 9
Array upper bound (2nd Dimension) = 4

<p>| | | | | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
</tbody>
</table>

<p>| | | | | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>4</td>
<td>6</td>
<td>8</td>
<td>4</td>
<td>5</td>
</tr>
<tr>
<td>6</td>
<td>1</td>
<td>6</td>
<td>10</td>
<td>0</td>
</tr>
<tr>
<td>1</td>
<td>7</td>
<td>10</td>
<td>5</td>
<td>5</td>
</tr>
<tr>
<td>2</td>
<td>5</td>
<td>4</td>
<td>4</td>
<td>8</td>
</tr>
<tr>
<td>8</td>
<td>3</td>
<td>2</td>
<td>3</td>
<td>2</td>
</tr>
<tr>
<td>9</td>
<td>8</td>
<td>5</td>
<td>6</td>
<td>1</td>
</tr>
<tr>
<td>4</td>
<td>6</td>
<td>4</td>
<td>6</td>
<td>4</td>
</tr>
<tr>
<td>5</td>
<td>4</td>
<td>2</td>
<td>3</td>
<td>6</td>
</tr>
<tr>
<td>7</td>
<td>9</td>
<td>8</td>
<td>0</td>
<td>5</td>
</tr>
<tr>
<td>2</td>
<td>10</td>
<td>8</td>
<td>7</td>
<td>0</td>
</tr>
</tbody>
</table>

Enter the value to search for: 3
Value found: Row = 4, Col = 1
Value found: Row = 4, Col = 3
Value found: Row = 7, Col = 3