Indexing -- Just More Of The Same

A common way to refer to many instances of the same thing is to give them a single name and index them. So we have Super Bowl XX, Pope John 23, Taco Bell Franchise 229, etc. Indexing is handy in programming
Motivation: When there is a large number of similar things that must be referenced and manipulated, it can be inconvenient to think up a unique name for each, and to refer to them by the name.

For example: Each of the Seven Dwarfs has a name, but who can remember them? Also, it is difficult to refer to them in a loop since there is no way to enumerate them.

Indexing names the items by associating a base name and a number -- the index -- with each one.

Computer notation: Dwarf(5) ↔ Happy
Indexing Particulars

- In everyday indexing, it is common to begin the indexing with 1, e.g. May 1, SuperBowl I, Elizabeth I
- The number at which indexing begins is its origin
- Many computer languages use 1 as the origin, but many others, including Visual Basic 6.0, use 0 as the index origin
Arrays

- When a variable is indexed it is called an array.
- Arrays are used for representing collections of data values, e.g. integers, strings, etc.
  
  For example:
  
  \[
  \text{dwarf}(0) = "\text{Sneezy}\"
  \]
  
  \[
  \text{dwarf}(1) = "\text{Dopey}\"
  \]
  
  \[
  \text{dwarf}(2) = "\text{Grumpy}\"
  \]
  
  ...

- Elements of an array must all be of the same type.
- The index of an array element is also known as a subscript.

Notice \(x_0\) and \(x_1\) are variable names, while \(x(0)\) and \(x(1)\) are different elements of array \(x\).
Arrays are declared like any other variable using a Dim statement:

- Keyword
- Array name
- Largest Index
- Type

Dim dwarf(6) As String

**Notice**

- The syntax is just like a normal declaration except for the parenthesis pair
- In the parentheses is the largest desired index
- The total number of elements of the array will be one more than the largest index, since the origin is 0
- The type applies to all of the elements
Indexing Arrays

- To refer to different elements of the array, it is necessary only to change the index …
- The index value must be an integer constant (1), a variable (myNdex) or expression (myNdex+1)
- A loop can sweep through all elements

The dwarf array values were initialized in event handler Form_Load()
Combining Indexing, Arrays, Loops

- A common error is to index beyond the end of the array ...

```vbnet
Option Explicit
dim dwarf(6) as string

Private Sub Form_Click()
    Dim index as integer
    index = 1
    Do While index <= 7
        Print dwarf(index)
        index = index + 1
    Loop
End Sub
```

Microsoft Visual Basic

Run-time error '9':
Subscript out of range
Practice Using Arrays

- Draw a 10-segment “inch worm” on the screen and move it forward
- Use arrays to keep the positions of the segments
- Write procedures to initialize worm and draw it

Goals of exercise:
- Practice with arrays
- Practice with indexing
- Practice writing procedures
- Notice how arrays are passed as parameters

Center
inWx(0)
inWy(0)
Programming Drawing -- Step 1

- The first step is to declare the arrays and variables
- Will use the form click event handler to control the operation

The array inWx will store the x-coordinates for the segments, and inWy will store the y values. There are 10 segments.

The wormInit procedure will initialize the segment positions in the arrays.
Step 2, Initialize The Coordinates

- When arrays are passed to procedures, the formal parameter must show this with an empty parenthesis pair.
Step 3 -- Draw The Figure

- It will be necessary to draw the inch worm in different colors, so the color becomes a parameter -- Double

```vbnet
Private Sub segment(x1 As Integer, y1 As Integer, hue As Double)
    Circle (x1, y1), 100, hue
End Sub

Private Sub inWorm(x1() As Integer, y1() As Integer, hue As Double)
    Dim iterate As Integer
    iterate = 0
    Do While iterate < 10
        Call segment(x1(iterate), y1(iterate), hue)
        iterate = iterate + 1
    Loop
    Line (x1(9), y1(9)) - (x1(9) + 100, y1(9) - 200), hue
End Sub
```

Notice that inWorm is passed an array of x,y segments but segment is passed just a single position.
To make the worm move, it must be erased/redrawn
Click To Draw, Click To Erase

Private Sub Form_Click()
    clickCount = clickCount + 1
    If clickCount = 1 Then
        FillColor = RGB(0, 255, 0)
        FillStyle = 0
        Call inWorm(inWx(), inWy(), RGB(0, 255, 0))
    Else
        FillColor = RGB(255, 255, 255)
        FillStyle = 0
        Call inWorm(inWx(), inWy(), RGB(255, 255, 255))
    End If
End Sub
Move Worm With Repeated Drawing

Else
    stepNum = 0
Do While stepNum < 10
    FillColor = RGB(255, 255, 255)  * Erase existing worm
    FillStyle = 0
    Call inWorm(inWx(), inWy(), RGB(255, 255, 255))
    index = 0
Do While index < 10
    inWx(index) = inWx(index) + 100
    index = index + 1
Loop
    stepNum = stepNum + 1
    FillColor = RGB(0, 255, 0)  * Draw new worm 100 units right
    FillStyle = 0
    Call inWorm(inWx(), inWy(), RGB(0, 255, 0))
Loop