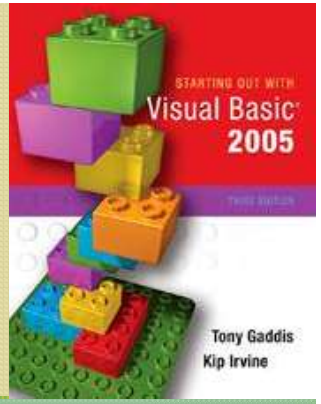
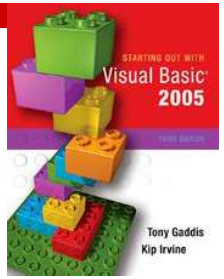


8.1 Arrays

An Array Is Like a Group of Variables With One Name

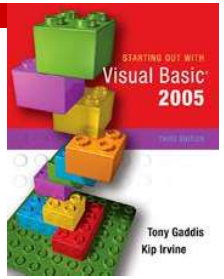
You Store and Work With Values in an Array by Using a Subscript

Array Characteristics



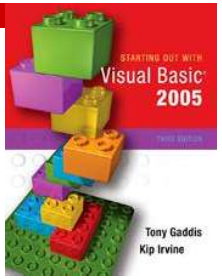
- An *array* stores multiple values of same type
- Like a group of variables with a single name
- For example, the days of the week would be:
  - a set of 7 string variables
  - maximum length of 9 characters
- All variables within an array are called *elements* and must be of the same data type
- You access the elements in an array through a subscript

Subscript Characteristics



- A *subscript*, also known as an *index*, is a number that identifies a specific element within an array
- Subscript numbering works like the index for a list box:
  - Subscript numbering begins at 0
  - 1st element in an array is always subscript 0
  - Last element is total number of elements - 1

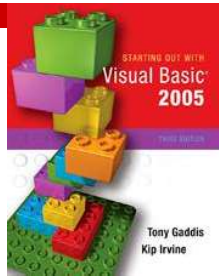
Declaring an Array



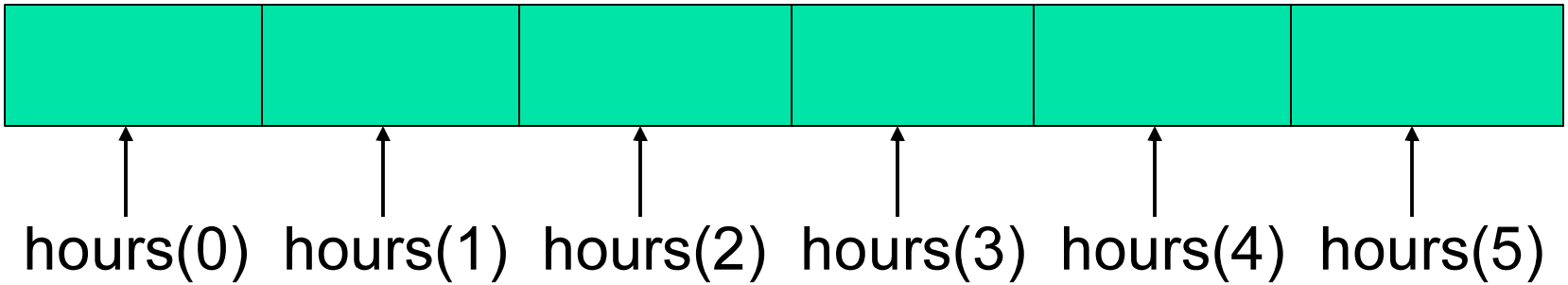

```
Dim ArrayName(UpperSubscript) As DataType
```

- ***ArrayName*** is the variable name of the array
- ***UpperSubscript*** is the value of the array's highest subscript
  - Must be a positive whole number
- ***DataType*** may be any valid Visual Basic data type
- VB knows an array is declared when *name* is followed by number of elements in parens

Array Declaration Example

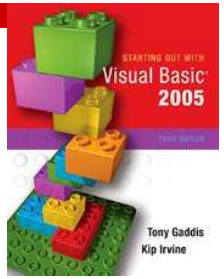



```
Dim hours(5) As Integer
```



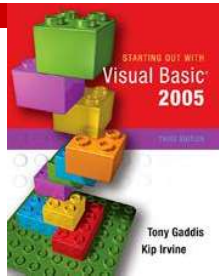
- Note that 6 elements are available when an array with an uppersubscript of 5 is declared
- Each element is an integer type

Default Initialization



- All elements of an Integer array are initialized to zero
  - Same initialization as an integer variable
- Each array element is initialized exactly the same as a simple variable of that data type
  - A Single is initializled to zero (0.0)
  - A String is initialized to nothing (empty string)

Implicit Array Sizing & Initialization

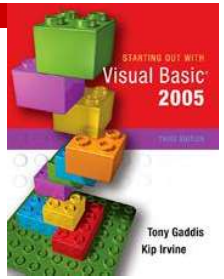


- An array can be initialized when declared
- Example:

```
Dim numbers() As Integer = {2, 4, 5, 7, 9, 12}
```

- This array is implicitly sized
  - Upper subscript value is left blank
  - Number of elements implied from initialization
  - Upper subscript of 5 implied by this example
  - A 6 element array is declared
- Elements are assigned the values shown

Named Constant as Upper Subscript

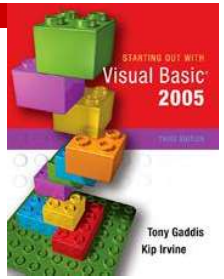


- A named constant may be used as an array's highest subscript instead of a number

```
Const UPPER_SUB As Integer = 100
Dim array(UPPER_SUB) As Integer
```

- A common use for named constants
  - Highest subscript is often used multiple times
  - If highest subscript changes, use of a named constant allows it to be changed in one place

Working With Array Elements



- Simply include the subscript to use an array element like an ordinary variable
- **`numbers(2)`** refers to 3rd element of array

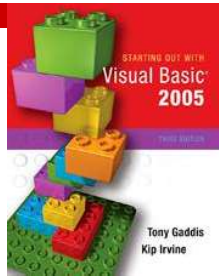
```
numbers(0) = 100
numbers(1) = 200
numbers(2) = 300
numbers(3) = 400
numbers(4) = 500
numbers(5) = 600
```

```
pay = hours(3) * rate

tallies(0) += 1

MessageBox.Show(grossPay(5).ToString)
```

Arrays and Loops



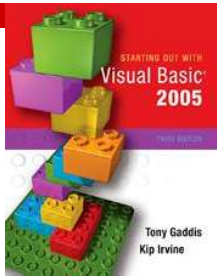
- Loops are frequently used to process arrays
- Use a loop to prompt for 10 values

```
Dim count As Integer
For count = 0 To 9
   series(count) = CInt(InputBox("Enter a number"))
Next count
```

- Use a loop to reset values in the names array to an empty string

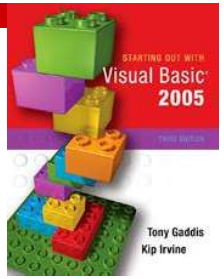
```
Dim count As Integer
For count = 0 To 999
   names(count) = ""
Next count
```

Array Bounds Checking



- Visual Basic checks each subscript value of each array reference used at run time
- A subscript is invalid if it is
  - Negative
  - Greater than the UpperSubscript declared
- An invalid subscript causes VB to throw an exception
- Bounds checking is *not* done at design time

For Each … Next Statement



- This statement is similar to a For…Next
- Iterates for each element in the array
- strName assigned value of next array element at each iteration

```
Dim employees As String() = {"Jim", "Sally", _
            "Henry", "Jean", "Renee"}
Dim strName As String
For Each strName In employees
      MessageBox.Show(strName)
Next name
```

- Tutorial 8-1 demonstrates array processing