Programmering med Visual Basic, grundkurs

Comparing, Testing, and Working With Strings

Contents:

Strings

Comparing and manipulating strings.

Instructor: Farid Naisan, University Lecturer, farid.naisan@mah.se

Strings Can Be Compared

- Relational operators can be used to compare strings and string literals as well as numbers

```
strName1 = "Mary"
strName2 = "Mark"
If strName1 = strName2 Then
      lblMessage.Text = “Names are the same"
Else
      lblMessage.Text = “Names are NOT the same"
End If


If strMonth <> "October" Then
      ' statement
End If
```

How Are Strings Compared?

- Each character is encoded as a numerical value using the *Unicode* standard
- Letters are arranged in alphabetic order
  - The Unicode numeric code for A is less than the Unicode numeric code for B
- Characters of each string are compared one by one until a difference is found
  - `M a r y`
  - `M a r k`

Mary is greater than Mark because "**y**" has a Unicode value greater than "**k**"

How Are Strings Compared?

- Upper case letters do ***not*** have the same value as their lower case equivalents
  - Upper case letters are less than lower case
- The >, <, >=, and <= operators can be used with strings as well
- If one string is shorter than another, spaces are substituted for the missing characters
- Spaces have a lower value than letters
  - "Hi" has 2 spaces added if compared to "High"
  - "Hi " is less than "High"

The Empty String

- A space (or blank) is considered a character
- An empty string is a string with no characters
  - A string with just spaces ***has*** characters in it
- The empty string is written as "", as in the following code that tests for no input:

```
If txtInput.Text = "" Then
      lblMessage.Text = "Please enter a value"
End If
```


ToUpper Method

- *ToUpper* method can be applied to a string
- Results in a string with lowercase letters converted to uppercase
- The original string is not changed

```
littleWord = "Hello"
bigWord = littleWord.ToUpper()
      ' littleWord retains the value "Hello"
      ' bigWord is assigned the value "HELLO"
```

ToLower Method

- The *ToLower* method performs a similar but opposite purpose
- Can be applied to a string
- Results in a string with the lowercase letters converted to uppercase
- The original string is not changed

```
bigTown = “New York"
littleTown = bigTown.ToLower()
      ' bigTown retains the value “New York"
      ' littleTown is assigned the value “new york"
```


A Handy Use for ToUpper or ToLower

- *ToUpper* or *ToLower* can be used to perform case insensitive comparisons of strings
- 1st comparison below is false “Hello”<>“hello”
- 2nd comparison is true
  - ToLower converts both strings to lower case
  - Causes “hello” to be compared to “hello”

```
word1 = "Hello“
Word2 = “hello”
If word1 = word2                      ‘false, not equal
If word1.ToLower() = word2.ToLower()  ‘true, equal
```

- Tutorial 4-6 demonstrates how this is used

Determining the Length of a String

- The *Length* method determines the length of a string, e.g.:

```
If txtInput.Text.Length > 20 Then
      lblMessage.Text = “Enter fewer than 20 characters."
End If
```

Note: **`txtInput.Text.Length`** means to apply the Length Method to the value of the Text property of the Object txtInput

Trimming Spaces from Strings

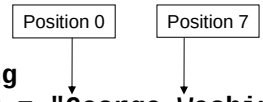
- There are three Methods that remove spaces from strings:
  - *TrimStart* – removes leading spaces
  - *TrimEnd* – removes trailing spaces
  - *Trim* – removes leading ***and*** trailing spaces

```
greeting = " Hello "
lblMessage1.Text = greeting.TrimStart()
      ' Returns the value "Hello "

lblMessage1.Text = greeting.Trim()
      ‘ Returns the value "Hello"
```

The Substring Method

- The *Substring* method returns a portion of a string or a “string within a string” (a substring)
- Each character position is numbered sequentially with the 1st character referred to as position zero
- *StringExpression*.Substring(*Start*)
  - returns the characters from the *Start* position to the end
- *StringExpression*.Substring(*Start*, *Length*)
  - returns the number of characters specified by *Length* beginning with the *Start* position

Substring Method Examples



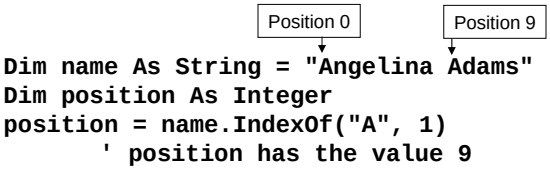

```
Dim firstName As String
Dim fullName As String = "George Washington"
firstName = fullName.Substring(0, 6)
      ' firstName assigned the value "George"
      ' fullName is unchanged

lastName = fullName.Substring(7)
      ‘ lastName assigned the value “Washington”
      ‘ fullName unchanged
```

Search for a String Within a String

- Use the *IndexOf* method
- *StringExpression*.IndexOf(*Searchstring*)
  - Searches the entire string for *Searchstring*
- *StringExpression*.IndexOf(*SearchString*, *Start*)
  - Starts at the character position *Start* and searches for *Searchstring* from that point
- *StringExpr*.IndexOf(*SearchString*, *Start*, *Count*)
  - Starts at the character position *Start* and searches *Count* characters for *SearchString*

IndexOf Method Examples

- *IndexOf* will return the starting position of the SearchString in the string being searched
- Positions are numbered from 0 (for the first)
- If SearchString is not found, a -1 is returned




```
Dim name As String = "Angelina Adams"
Dim position As Integer
position = name.IndexOf("A", 1)
       ' position has the value 9
```

- Tutorial 4-7 provides an opportunity to work with several of the string methods