VB.NET Programming Cheat Sheet

Davenant Foundation School - ICT & Computing Department

Contents

VARIABLES & DATA TYPES

```
Dim messagecontent As String = "Program!"     'String variable
Dim mynumber As Integer = 10                  'Integer variable
Dim myprice As Single = 3.99                  'Real number variable
Dim mychoice As Boolean = True                'Boolean variable
Dim names(4) As String                        '1D 5 element String array
Dim results(3,6) as Integer                   '2D 4 row, 7 column Integer array


Structure MemberDetails                       'Record of different data types
   Dim IDnumber As Integer
   Dim Custname As String
   Dim Paid As Boolean
End Structure
Dim Members(99) As MemberDetails              'Record variable with 100 rows
```

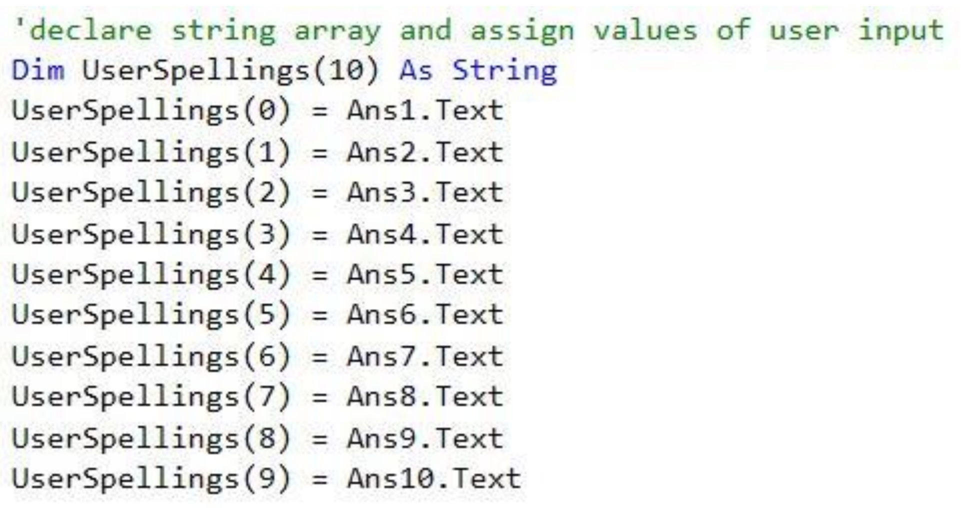
The data types are very similar to those used within Python. The main differences are:

- You must declare variables using **Dim** and assign a data type
- All variables will be local and passed between functions as parameters

Arrays

An array is a single data structure containing a list of elements of the same data type. For example Names could contain ‘Bob’, ‘Mary’ and ‘Mo’. The elements can be referred to as Names(0), Names(1) etc. The index number must start at 0 for the forst element.

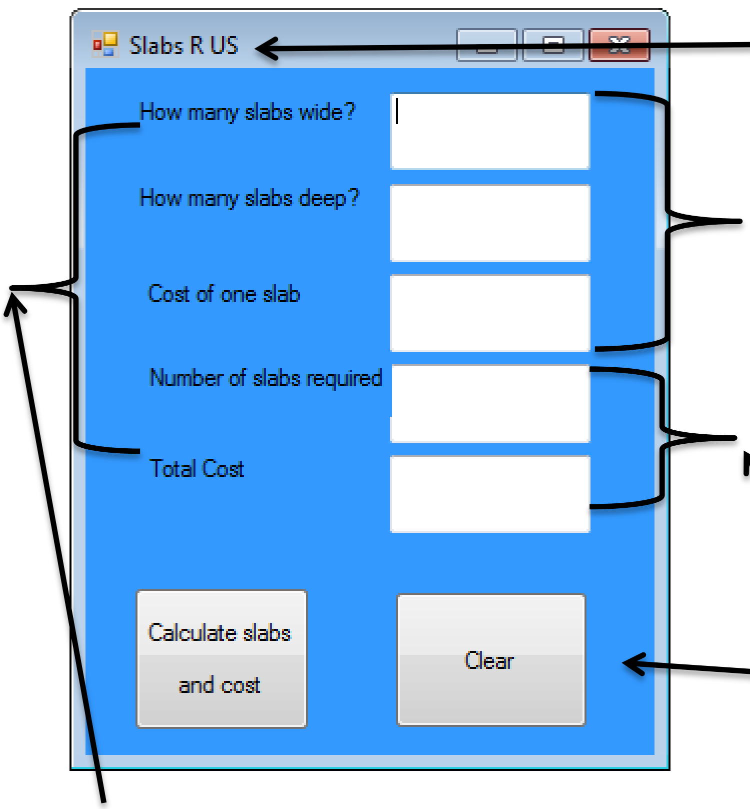
When declaring arrays and records you must assign the number of elements. See this example of declaring an array with 10 elements, then assigning values.



```
'declare string array and assign values of user input
Dim UserSpellings(10) As String
UserSpellings(0) = Ans1.Text
UserSpellings(1) = Ans2.Text
UserSpellings(2) = Ans3.Text
UserSpellings(3) = Ans4.Text
UserSpellings(4) = Ans5.Text
UserSpellings(5) = Ans6.Text
UserSpellings(6) = Ans7.Text
UserSpellings(7) = Ans8.Text
UserSpellings(8) = Ans9.Text
UserSpellings(9) = Ans10.Text
```

THE VISUAL BASIC ENVIRONMENT

Here is what a simple Visual Basic program might look like. When excecuted the program appears on the screen as a **form.**



At the top of the form there is a **caption** (Slabs R US)

There are 3 **text boxes** for entering data

Two **text boxes** which will display the results of a calculation.

Two command **buttons** (Calculate Slabs and cost and Clear)

Five labels to describe the contents of the text boxes.

Each of these items is called an **object.**

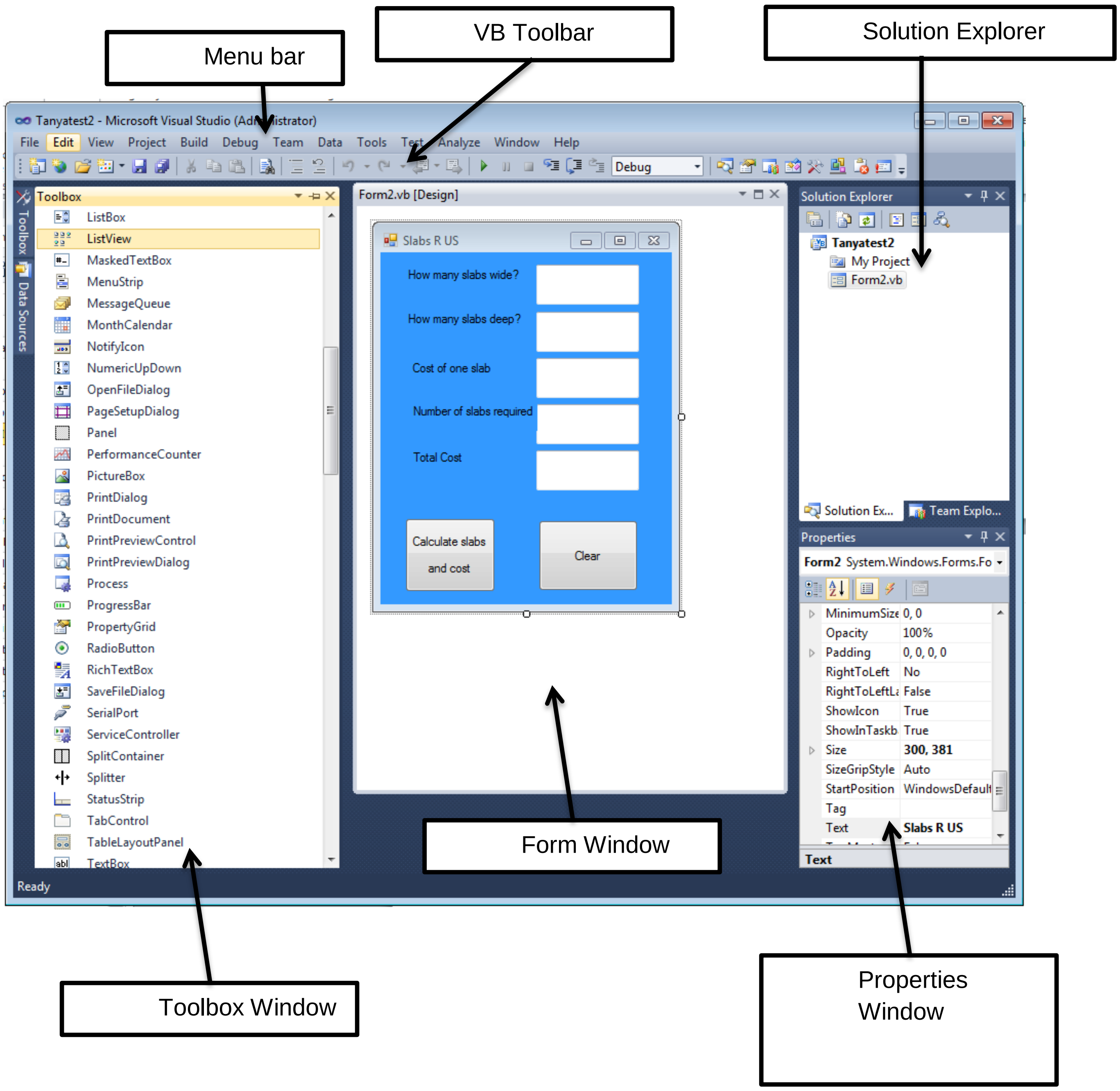
Each object has **properties.** When an object is selected you can see its properties in the properties window. The properties can be changed using your visual basic program code.

Some objects **(particularly command buttons)** have program code associated with them**, so something happens when a user clicks the button.**

There are lots of other objects but these are the fundamentals. Explore the others in the VB toolbox.

VISUAL BASIC ENVIRONMENT CONTINUED

Here is the same program when it was being developed, showing some of the main features of the Visual Basic development environment:



MinimumSize	0, 0
Opacity	100%
Padding	0, 0, 0, 0
RightToLeft	No
RightToLeftLa	False
ShowIcon	True
ShowInTaskb	True
Size	**300, 381**
SizeGripStyle	Auto
StartPosition	WindowsDefault
Tag	
Text	**Slabs R US**

INPUT AND OUTPUT

Common methods of input and output with VB are shown below. This is not by any means definitive.

Forms of Input

With Python we use raw_input() or input() to take input from the user, with vb we use text boxes on a form or pop up input boxes.

1) **THROUGH A TEXT BOX**

‘Add a text box to the form called txtMessage

```
Dim name As String        ‘Declare a variable called name which will contain a string
```

```
name=txtMessage.Text   ‘Assigns the text in the text box to the name variable
```

Then output the contents of name using your chosen method (see below)

2) **THROUGH A POP-UP INPUT BOX**

```
Dim name as String        ‘Declare a string variable called name
```

```
name=InputBox(“Enter your name”)
```

Then output the contents of name using your chosen method (see below)

Forms of Output

With python we use print() to output on the screen. With vb output can be displayed in text boxes, labels or pop up message boxes.

THE BELOW EXAMPLES SHOW THREE WAYS TO OUTPUT A STRING OF TEXT “welcome to VB” FOLLOWED BY A VARIABLE CALLED name.   The code below could be added below the code shown the the input methods in 1 and 2 above.

1) **OUTPUT TO MESSAGE BOX**

```
MsgBox(“welcome to VB, “ & name)
```

2) **OUTPUT TO A LABEL**

! First add a label to the form called lblMessage

```
lblMessage.Text=”welcome to VB, “ & name
```

3) **OUTPUT TO A TEXT BOX**

‘Add a text box to the form called txtMessage

```
txtMessage.Text=”welcome to VB, “ &  name
```

CONDITIONAL STATEMENTS

If....End If

```
If TextBox1.Text = "hello" Then
    Label1.Text = "world"
End If
```

If....ElseIf....End If

```
If RadioButton1.Checked = True Then
    MessageBox.Show("You are under 16!")
ElseIf RadioButton2.Checked = True Then
    MessageBox.Show("You are over 16!")
End If
```

If....Else....End If

```
If TextBox1.Text = "password123" Then
    data.Show()
Else
    MessageBox.Show("Wrong Password!")
End If
```

Comparison	Meaning
=	Equal to
<>	Not equal to
>	Greater than
<	Less than
>=	Greater than or equal to
<=	Less than or equal to

LOOPS

You know how many times to repeat the loop, use a **For** with an integer counter:

```
For i = 0 To 10
    MessageBox.Show("The value of i is: " & i)
Next
```

The **Do While** loop is repeated until a certain condition is met:

```
Do While num1 < 10
    MessageBox.Show("The value of num1 is :" & num1)
    num1 = num1 + 1
Loop
```

The **Do Until** loop is a useful variant:

```
Do Until num = 5
    MessageBox.Show("The value of num is: " & num)
    num = num + 1
Loop
```

SUB PROGRAMS & FUNCTIONS

Event and Load Sub Programs

VB.NET programs should be modular. Just like python there are lots of built in functions for you to use. You can use the following subprogram types:

1. **An Event Procedure** – e.g the code is executed when a user clicks a button (the bulk of your code will be here)

```
Private Sub button1_Click (……….)

        Form2.show()

End Sub
```

Programmer Defines Sub Programs

Just like python you can write your own functions. This is appropriate if you want to reuse code. Functions in vb always return a value. If you don't want to return a value, use a sub (see next section).

There are two ways that a function can return a value (you must choose one of them).

Functions receiving values

In the first a variable is declared as part of the function statement (in this case E_to_P As Single).

```
Function E_to_P(ByVal Euros As Integer, ByVal Ex_Rate As Single) As Single
    E_to_P = Euros * Ex_Rate
End Function
```

A value must be assigned to E_to_P as part of the function body. This is the value that will be returned.

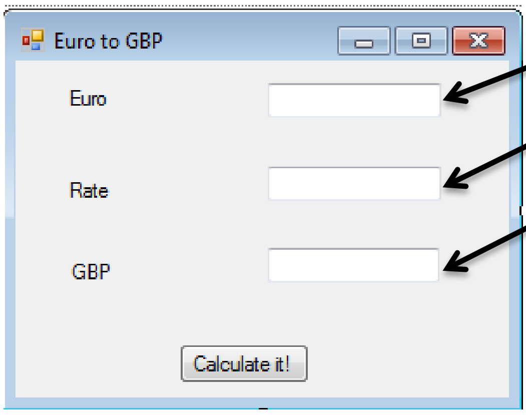
Functions returning values

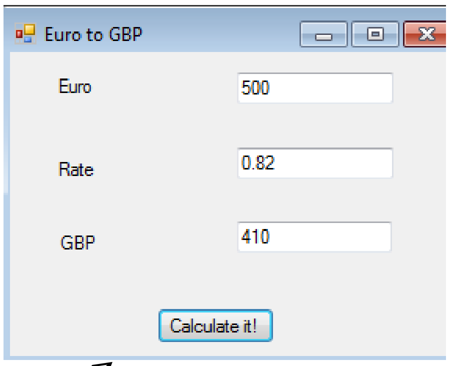
The second method, just like Python, uses a **return** statement. In this example the value of the Pounds variable will be returned.

```
Function E_to_P(ByVal Euros As Integer, ByVal Ex_Rate As Single)
    Dim Pounds As Single
    Pounds = Euros * Ex_Rate
    Return Pounds
End Function
```

SIMPLE EXAMPLE USING AN EVENT PROCEDURE AND THE FUNCTION E_TO_P ABOVE

This form has three text box objects. The name properties are txtEuro, txtRate and txtGBP.





The event procedure attached to the ‘Calculate it’ button is shown below:

```
Private Sub Button1_Click(ByVal sender As System.Object, ByVal e As System.EventArgs) Handles Button1.Click
    txtGBP.Text = E_to_P(txtEuro.Text, txtRate.Text)
End Sub
```

If the user enters 500 in the txtEuro text box and 0.82 in the txtRate text box and clicks ‘calculate it’ – the txtGBP text box will display 410 (as shown to the right):

Sub Routines

A (Programmer-Defined) **Sub** - like a function but cannot return a value. Use if you do not wish to return a value. Example of use: if you want to display a particular message.

```
Private Sub Say_Goodbye()

MessageBox(“Hope you enjoyed using our calculator – have a nice day!”)

End Sub
```

Parameters

Just like in python you can pass parameters to a user created function or sub.

In vb you need to declare the local variable in the first line of the function.

For example

If you are creating a function, ‘**BMI**’ which takes 2 parameters, **height** and **weight** of type **single.**

Function BMI(myVal height As single, myVal weight As single)

myVal and myRef

myVal and myRef precede the names of the parameters passed to a function. myVal is the default.You can pass variables by reference (ByRef) or by value (ByVal). Without getting technical, the basic difference is that if you pass a variable by reference, that means the function or subroutine can change the value of the global variable; if you pass it by value, it cannot. For example, consider this code:

```
Dim A As Integer = 5
Dim B As Integer = 7
PassByValue(A)
PassByRef(B)
Messagebox.show(A)
Messagebox.show(B)

Sub PassByValue(ByVal intIn As Integer)
intIn = 10
End Sub

Sub PassByRef(ByRef intIn As Integer)
intIn = 10
End Sub
```

A would remain as 5 as it is passed by value

B would be changed to 7 as it is passed by reference

USEFUL CODE SNIPPETS

Load a Form

To launch another form:

```
        Form2.Show()
```

Using Radio Buttons

To test if a radio button is selected or not:

```
        If RadioButton1.Checked = True Then
            MessageBox.Show("You are under 16!")
        End If
```

Using Check Boxes

To test if a checkbox is selected or not:

```
        If CheckBox1.Checked = True Then
            MessageBox.Show("You like Harry Potter, nerd")
        End If
```

Check a user wants to exit a form:

```
Private Sub Button1_Click(sender As Object, e As EventArgs) Handles Button1.Click
        Dim result = MsgBox("Are you sure you want to Exit ?", vbYesNo)
        If result = DialogResult.Yes Then
            Me.Close()
        End If
End Sub
```

Validation Examples

To use a **length check**:

```
Dim Password As String
Dim PasswordOK As Boolean = False
Do Until PasswordOK = True
        Password = InputBox("Enter password of at least 6 characters")
        If Len(Password) >= 6 Then
                PasswordOK = True
        End If
Loop
```

To use a **range check**:

```
Dim PIN As Single
Dim PINOK As Boolean = False
Do Until PINOK = True
        PIN = InputBox("Enter a 4 digit PIN")
        If PIN >= 1000 And PIN <= 9999 Then
                PINOK = True
        Else
                MessageBox("Please enter valid PIN")
        End If
Loop
```

To use a **list check**:

```
Dim response As String
Dim responseOK As Boolean = False
Do Until responseOK = True
        response = InputBox("Enter YES or NO")
        If UCase(response) = "YES" Or UCase(reponse) = "NO" Then
                responseOK = True
        Else
                MessageBox("Please enter valid PIN")
        End If
Loop
```

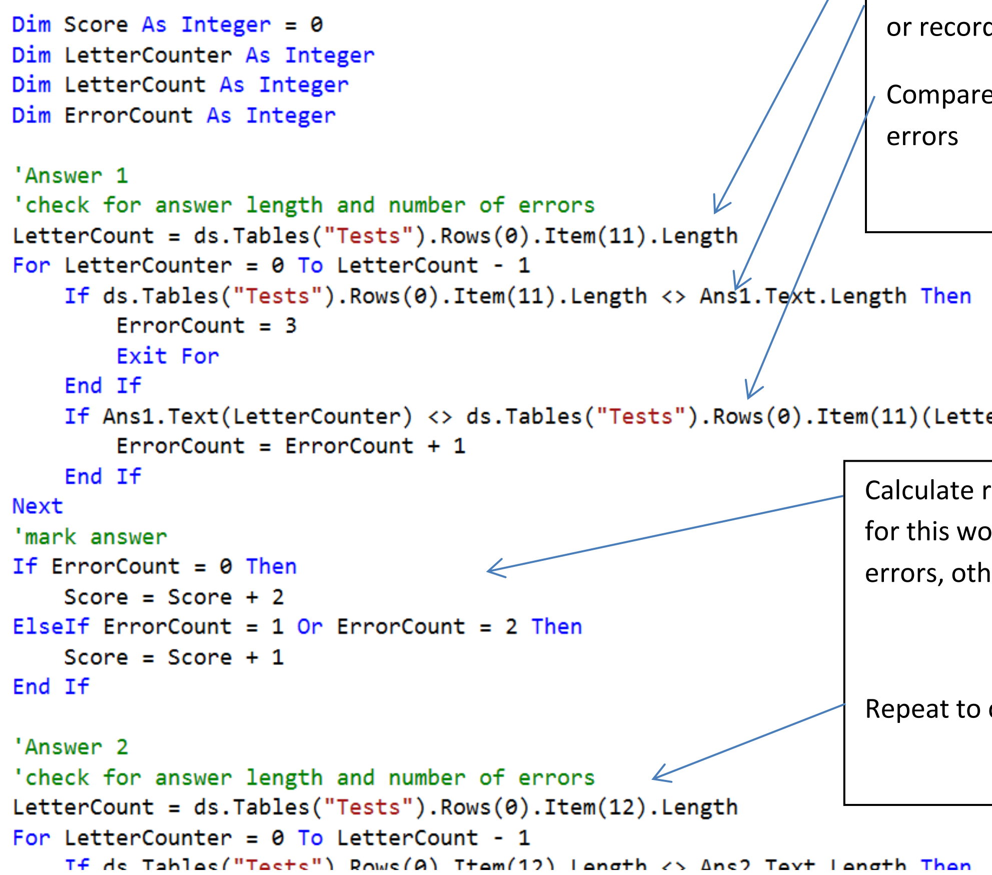
Comparing Two Strings character by character

Individual characters in a string can be used. These start from the first character, that being position 0.

For example, a string variable named Animal contains the word ‘cat’. Animal(0) contains c, Animal(1) contains a and Animal(2) contains t,

This example compares a String entered in a textbox with a record held in a database.

.....SELECT statement returns users data and fills a dataset table…..



```
Dim Score As Integer = 0
Dim LetterCounter As Integer
Dim LetterCount As Integer
Dim ErrorCount As Integer

'Answer 1
'check for answer length and number of errors
LetterCount = ds.Tables("Tests").Rows(0).Item(11).Length
For LetterCounter = 0 To LetterCount - 1
    If ds.Tables("Tests").Rows(0).Item(11).Length <> Ans1.Text.Length Then
        ErrorCount = 3
        Exit For
    End If
    If Ans1.Text(LetterCounter) <> ds.Tables("Tests").Rows(0).Item(11)(LetterCounter) Then
        ErrorCount = ErrorCount + 1
    End If
Next
'mark answer
If ErrorCount = 0 Then
    Score = Score + 2
ElseIf ErrorCount = 1 Or ErrorCount = 2 Then
    Score = Score + 1
End If

'Answer 2
'check for answer length and number of errors
LetterCount = ds.Tables("Tests").Rows(0).Item(12).Length
For LetterCounter = 0 To LetterCount - 1
    If ds.Tables("Tests").Rows(0).Item(12).Length <> Ans2.Text.Length Then
```

Determine the length of the word.

Ensure both words are the same length or record error

Compare character by character to count errors

Calculate running total based on mark for this word (2 if correct, 1 if one or two errors, otherwise 0)

Repeat to compare next words

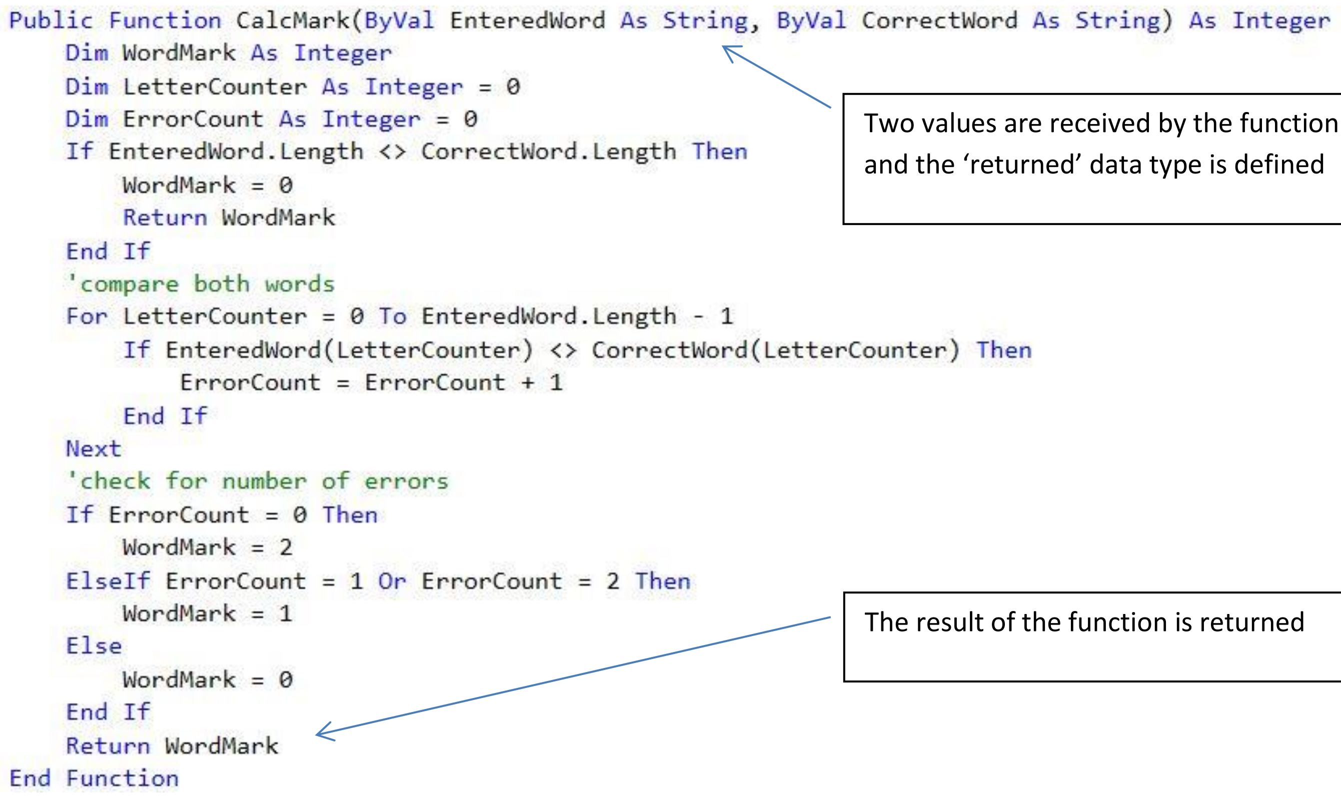
.....subsequent words are checked and score eventually saved to the database…..

Note – an issue here is that the program will be long owing to the very similar code repeated for each word comparison

Comparing Two Strings in a Function

The example code snippet on the previous page is functional but the length of the code can be reduced. This will use the information from the previous section on ‘programmer defined functions’. Remember a function received values and returns a result.

Here is a function to calculate the score based on the methodology used in the previous example

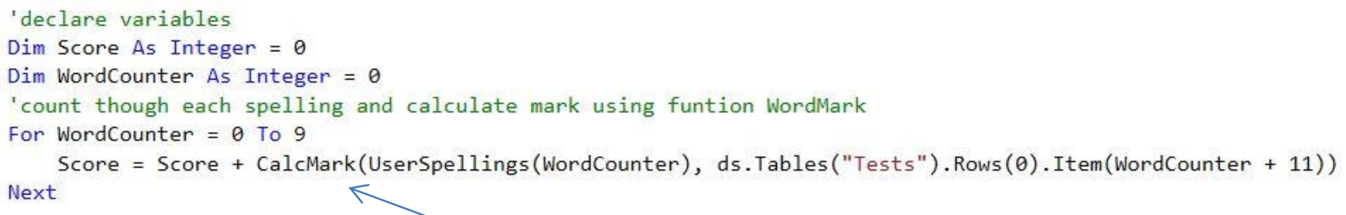


```
Public Function CalcMark(ByVal EnteredWord As String, ByVal CorrectWord As String) As Integer
    Dim WordMark As Integer
    Dim LetterCounter As Integer = 0
    Dim ErrorCount As Integer = 0
    If EnteredWord.Length <> CorrectWord.Length Then
        WordMark = 0
        Return WordMark
    End If
    'compare both words
    For LetterCounter = 0 To EnteredWord.Length - 1
        If EnteredWord(LetterCounter) <> CorrectWord(LetterCounter) Then
            ErrorCount = ErrorCount + 1
        End If
    Next
    'check for number of errors
    If ErrorCount = 0 Then
        WordMark = 2
    ElseIf ErrorCount = 1 Or ErrorCount = 2 Then
        WordMark = 1
    Else
        WordMark = 0
    End If
    Return WordMark
End Function
```

Two values are received by the function and the ‘returned’ data type is defined

The result of the function is returned

The function shown above will need to be called each time two strings are to be compared and scored. This will be in the event driven sub routine initiated by the user hitting the ‘mark test’ button.



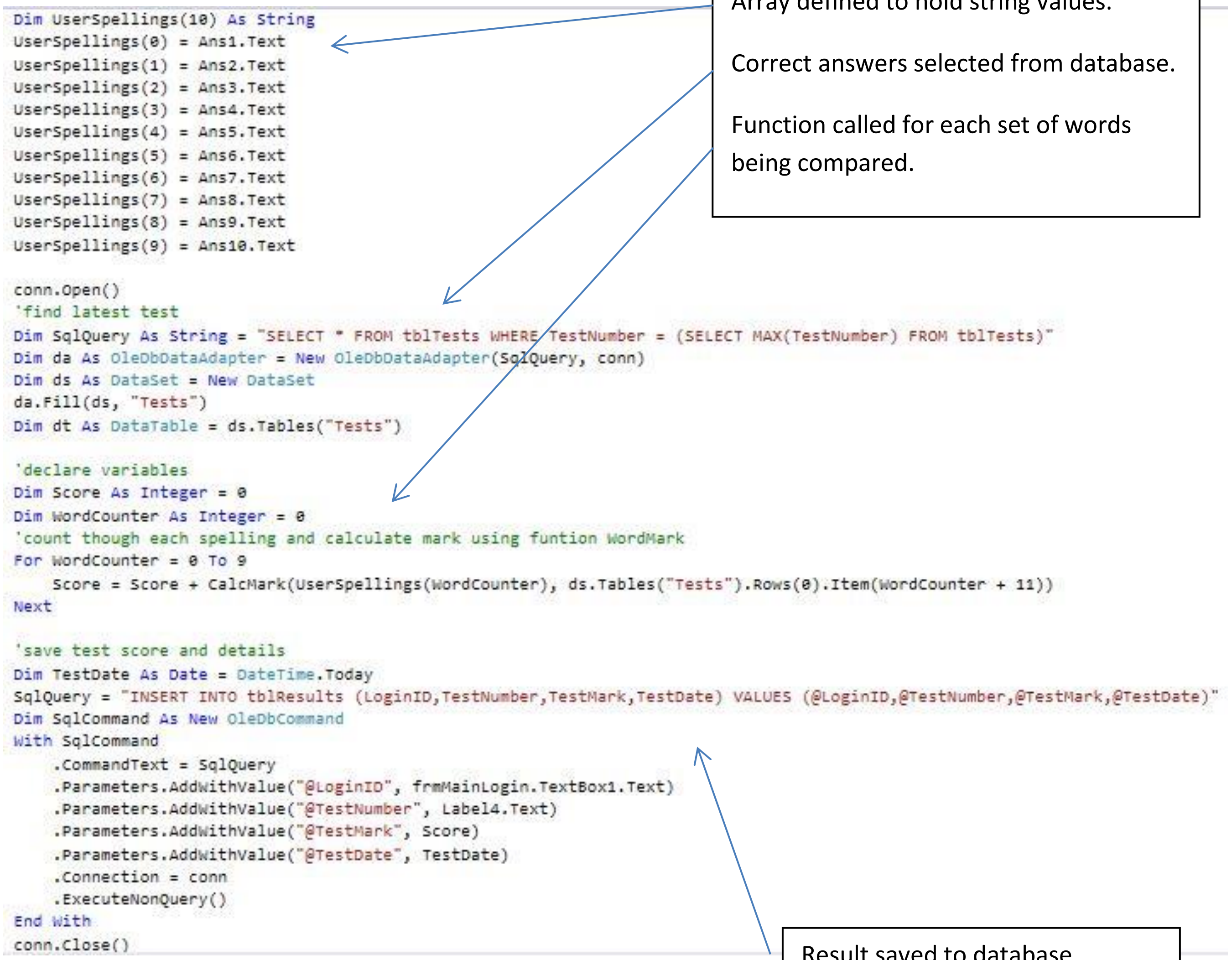
```
'declare variables
Dim Score As Integer = 0
Dim WordCounter As Integer = 0
'count though each spelling and calculate mark using funtion WordMark
For WordCounter = 0 To 9
    Score = Score + CalcMark(UserSpellings(WordCounter), ds.Tables("Tests").Rows(0).Item(WordCounter + 11))
Next
```

The function is called as part of a calculation.

Note the two values are passed to the function.

If you use a programmer defined function, you must plan how it is used in the overall sub program. Remember, you are passing values to the function and returning a value from the function.

Here is the example in the context of the overall sub program.



```
Dim UserSpellings(10) As String
UserSpellings(0) = Ans1.Text
UserSpellings(1) = Ans2.Text
UserSpellings(2) = Ans3.Text
UserSpellings(3) = Ans4.Text
UserSpellings(4) = Ans5.Text
UserSpellings(5) = Ans6.Text
UserSpellings(6) = Ans7.Text
UserSpellings(7) = Ans8.Text
UserSpellings(8) = Ans9.Text
UserSpellings(9) = Ans10.Text

conn.Open()
'find latest test
Dim SqlQuery As String = "SELECT * FROM tblTests WHERE TestNumber = (SELECT MAX(TestNumber) FROM tblTests)"
Dim da As OleDbDataAdapter = New OleDbDataAdapter(SqlQuery, conn)
Dim ds As DataSet = New DataSet
da.Fill(ds, "Tests")
Dim dt As DataTable = ds.Tables("Tests")

'declare variables
Dim Score As Integer = 0
Dim WordCounter As Integer = 0
'count though each spelling and calculate mark using funtion WordMark
For WordCounter = 0 To 9
    Score = Score + CalcMark(UserSpellings(WordCounter), ds.Tables("Tests").Rows(0).Item(WordCounter + 11))
Next

'save test score and details
Dim TestDate As Date = DateTime.Today
SqlQuery = "INSERT INTO tblResults (LoginID,TestNumber,TestMark,TestDate) VALUES (@LoginID,@TestNumber,@TestMark,@TestDate)"
Dim SqlCommand As New OleDbCommand
With SqlCommand
    .CommandText = SqlQuery
    .Parameters.AddWithValue("@LoginID", frmMainLogin.TextBox1.Text)
    .Parameters.AddWithValue("@TestNumber", Label4.Text)
    .Parameters.AddWithValue("@TestMark", Score)
    .Parameters.AddWithValue("@TestDate", TestDate)
    .Connection = conn
    .ExecuteNonQuery()
End With
conn.Close()
```

Array defined to hold string values.

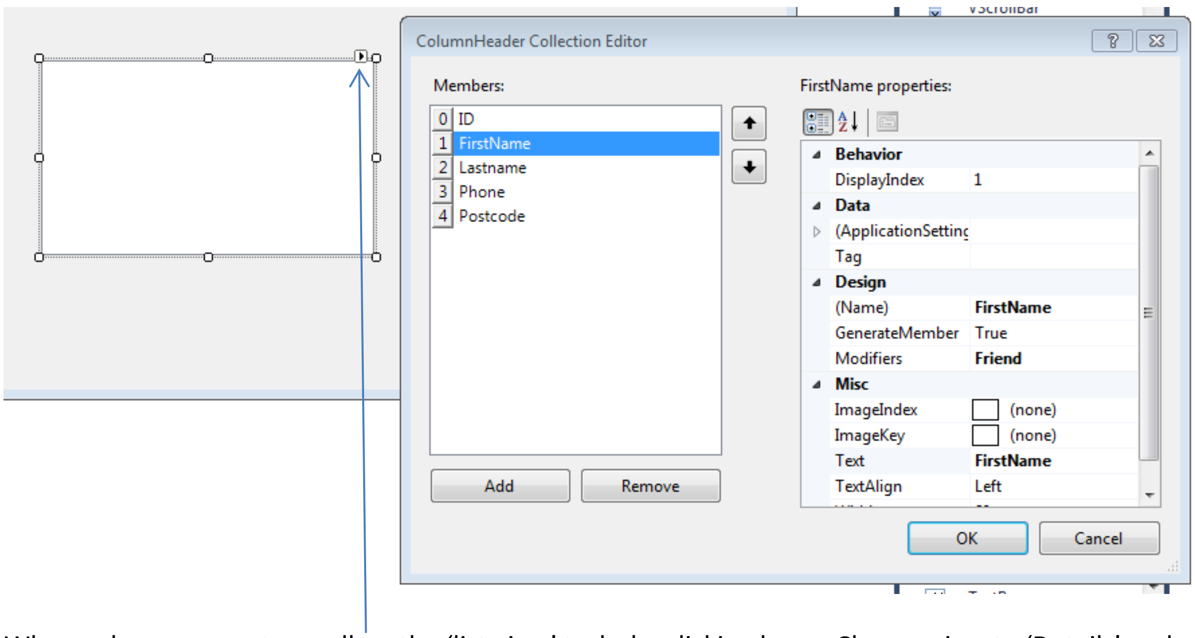
Correct answers selected from database.

Function called for each set of words being compared.

Result saved to database

Create a new List View Object

Add columns to the List View Object (right click, ‘edit columns’), changing the ‘Text Property’ in Misc to the name of the column.


ColumnHeader Collection Editor

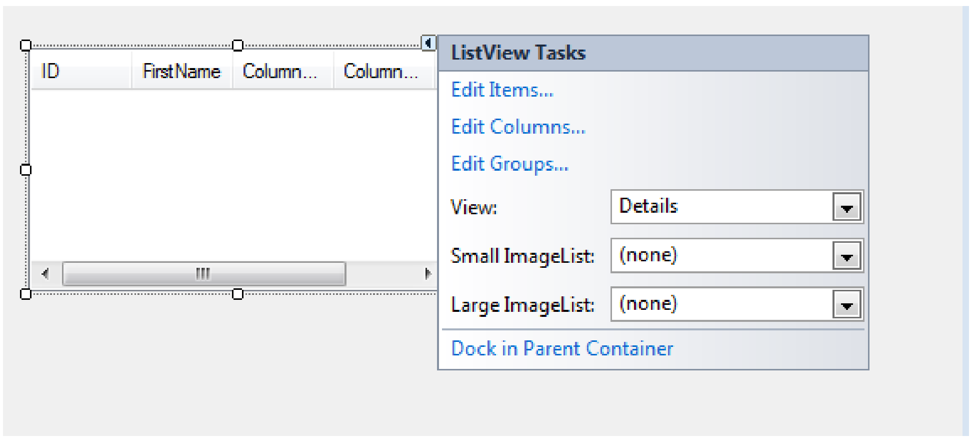
Members:

0	ID
1	FirstName
2	Lastname
3	Phone
4	Postcode

FirstName properties:

Behavior	
DisplayIndex	1
Data	
(ApplicationSetting	
Tag	
Design	
(Name)	**FirstName**
GenerateMember	True
Modifiers	**Friend**
Misc	
ImageIndex	(none)
ImageKey	(none)
Text	**FirstName**
TextAlign	Left

When columns are set up call up the ‘list view’ tasks by clicking here. Change view to ‘Details’ and column headings will appear:


ID	FirstName	Column...	Column...

ListView Tasks

Edit Items...

Edit Columns...

Edit Groups...

View: Details

Small ImageList: (none)

Large ImageList: (none)

Dock in Parent Container