Visual Basic – Messages and data input/output

Introduction
One way for a user to communicate with a procedure is via a dialogue box. The easiest way to do this in VB is to use one of the pre-defined ones. VB has two kinds, a Message box and an Input box.

Message box
The **MsgBox** function displays a message, waits for the user to click a button and returns a value indicating which button has been chosen. The simplest **MsgBox** contains only a message string and an OK button. The general syntax is:

```
MsgBox(prompt [,buttons] [,title]),
```

where the quantities within [] are optional arguments, with:
- **prompt**: string expression displayed in the message (max length 1024 characters)
- **buttons**: numerical expression that is sum of values specifying the type of buttons to display,
- **title**: string expression displayed in the title bar.

Some of the button values are given below. (For a full list see the Help file).

<table>
<thead>
<tr>
<th>Value</th>
<th>Constant</th>
<th>Display</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>vbOKOnly</td>
<td>OK button only</td>
</tr>
<tr>
<td>1</td>
<td>vbOKCancel</td>
<td>OK and Cancel buttons</td>
</tr>
<tr>
<td>3</td>
<td>vbYesNoCancel</td>
<td>Yes, No and Cancel buttons</td>
</tr>
<tr>
<td>4</td>
<td>vbYesNo</td>
<td>Yes and No buttons</td>
</tr>
<tr>
<td>32</td>
<td>vbQuestion</td>
<td>Query icon</td>
</tr>
<tr>
<td>48</td>
<td>vbExclamation</td>
<td>Warning message icon</td>
</tr>
<tr>
<td>0</td>
<td>vbDefaultButton1</td>
<td>First button is default</td>
</tr>
<tr>
<td>256</td>
<td>vbDefaultButton2</td>
<td>Second button is default</td>
</tr>
<tr>
<td>512</td>
<td>vbDefaultButton3</td>
<td>Third button is default</td>
</tr>
</tbody>
</table>

The value returned by the **MsgBox** function depends on the button pressed. Some values are listed below.

<table>
<thead>
<tr>
<th>Button selected</th>
<th>Value</th>
<th>Constant</th>
</tr>
</thead>
<tbody>
<tr>
<td>OK</td>
<td>1</td>
<td>vbOK</td>
</tr>
<tr>
<td>Cancel</td>
<td>2</td>
<td>vbCancel</td>
</tr>
<tr>
<td>Yes</td>
<td>6</td>
<td>vbYes</td>
</tr>
<tr>
<td>No</td>
<td>7</td>
<td>vbNo</td>
</tr>
</tbody>
</table>

The **MsgBox** function can be used as a simple debug tool. To display the contents of variable variDisplay then use:

```
MsgBox “Contents of variDisplay “ & variDisplay
```

which creates a message made from concatenating the text `Contents of variDisplay` with the data in the variable variDisplay.
The example below displays a message box with two buttons, Yes and No, with No as the default response. The value returned by the **MsgBox** function depends on the button pressed by the user.

```vbnet
msg = “Do you want to continue?”
boxButtons = vbYesNo + vbDefaultButton2
msgTitle = “Processing mode”
response = MsgBox(msg, boxButtons, msgTitle)
If response = vbYes Then
    msg = “clicked YES”
Else
    msg = “Clicked No or pressed ENTER”
End If
MsgBox msg
```

**Data input and output**

A procedure often needs some data on which it applies its actions. A **Function** procedure will usually get its input data from its arguments and returns a value in its name. A **Sub** procedure could get its input from arguments but it doesn’t return a value, yet it needs to return data somehow. One method is for the data to be taken from, or put into cells of a spreadsheet. For example

```vbnet
xVal = ActiveSheet.Range(“B2”).Value
```

assigns to the variable `xVal` in the code the contents of cell B2.

The previous technique presupposes that the values of the data are known in advance and have been entered into the spreadsheet BEFORE the procedure is run. The **InputBox** function creates and displays a simple dialogue box that contains a prompt, an edit box, and OK and Cancel buttons. You use this box to allow the user to input data at run-time. The format of the **InputBox** function is

```vbnet
InputBox(Prompt, [,title] [,default] [,xpos] [,ypos])
```

with

- **prompt**: a string expression displayed in the box,
- **title**: a string expression displayed in the dialogue box’s title bar. If omitted nothing is displayed.
- **default**: default response if no input provided.
- **xpos, ypos**: specify the horizontal and vertical position of the box. If omitted the box is centred horizontally and about one-third of the way down the screen.

A simple example is

```vbnet
radius = InputBox(“Enter radius of circle”, “Circle _
radius”)```
which will display a dialogue box with a title “Circle radius” and a message “Enter radius of circle” and wait for the user to enter a value.

The `InputBox` Method (of the Application object) works like the `InputBox` function but the method also allows you specify the data type. This enable you to enter a range, e.g. A1:A10. If data entered is of the wrong type Excel displays an error message. The format to use is

```
Application.InputBox(prompt, title, default, left, top, type)
```

The arguments, `prompt`, `title` and `default` are as for the `InputBox` function. The arguments, `left`, `top`, and `type` are optional, (`left` and `top` specify the horizontal and vertical position from the top left of the screen in point units (=1/72.27 inch)), `type` specifies the data type. If type is omitted the method returns text. Type has the following values,

<table>
<thead>
<tr>
<th>Value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>a formula</td>
</tr>
<tr>
<td>1</td>
<td>a number</td>
</tr>
<tr>
<td>2</td>
<td>text (string)</td>
</tr>
<tr>
<td>4</td>
<td>logical</td>
</tr>
<tr>
<td>8</td>
<td>cell reference (a range)</td>
</tr>
<tr>
<td>16</td>
<td>error value</td>
</tr>
<tr>
<td>64</td>
<td>an array of values</td>
</tr>
</tbody>
</table>

Note, type = 1 + 2 accepts text or number.

The code below asks the user to specify a range to search and a search value. The search range must be a valid range, e.g. sheet!A1:A10, and the search value is a number.

```vba
Sub CountEntries()
    Dim allCount As Integer, rangeToSearch As Object
    Dim searchValue, c
    cellCount = 0
    Set rangeToSearch = Application.InputBox( 
        Prompt: = “Enter range to search”, 
        Type: = 8)  
    ' type 8 means entry must be a range object
    searchValue = Application.InputBox( 
        Prompt: = “Search for value”, 
        Type: = 1)  
    ' type 1 means a number
    If searchValue = False Then Exit Sub  
    ' user clicked Cancel
    For Each c In rangeToSearch
        If c.Value = searchValue Then
            cellCount = cellCount + 1
        End If
    Next c
    MsgBox “Number of occurrences of “ & searchValue _
        & “ is “ & cellCount
End Sub
```