Suburbs and Postcodes

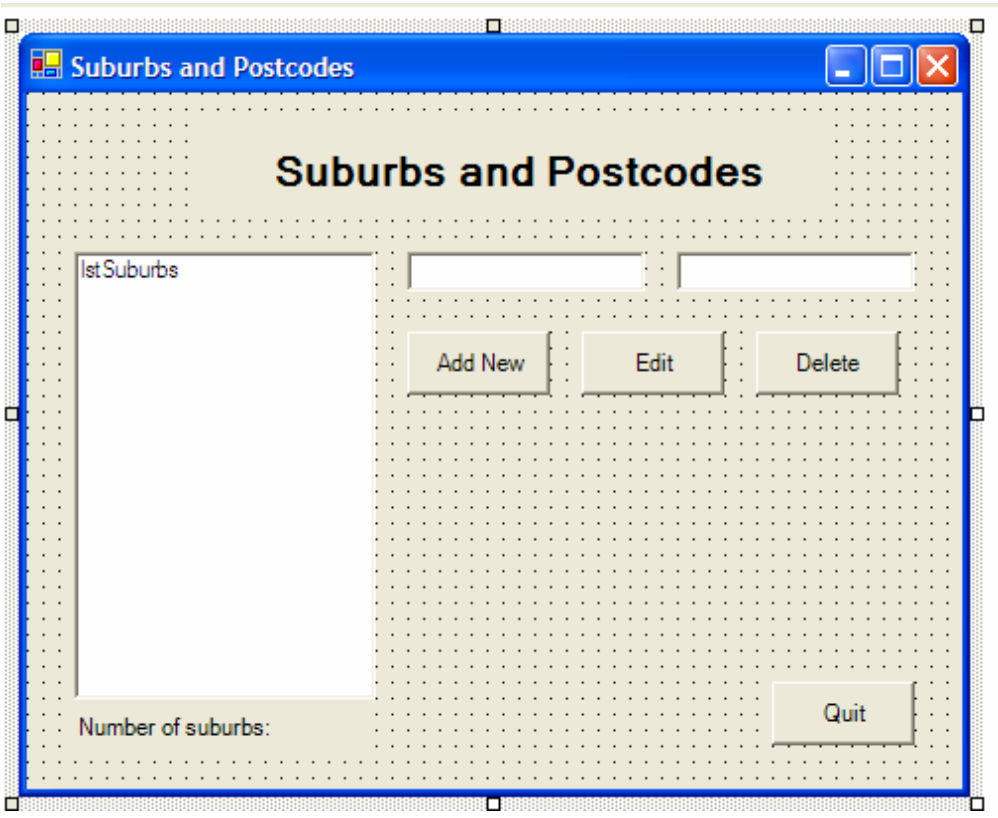
Problem Description: You are to create a program that uses a list box to display up to 50 towns or suburbs and their postcodes in order of suburb. All of the suburbs and postcodes will be stored together as single entries in the list, which means that the program will need to be able to extract the separate suburb and postcode from each entry. The user can interact with the program in the following ways:

1. Add town information to the list. (Ensure that no duplication of places occurs).
2. Find a postcode for a given town. Allow for the situation where a town you wish to find is not on the list.
3. Delete or edit an entry.

When the program starts, it should read the contents of the list from a text file. When the user is finished, the program should write the completed list back to the same file.

Programming Level: Intermediate

Skills Covered: reading and writing text files, list boxes, string processing commands, validation.



Setting up the objects for our program

Follow these steps to create the program:

1. Start a new Visual Basic.NET program called ‘Suburbs’.

2. Select the ListBox tool from the toolbox and draw a large list box on the left hand side of the form as shown. Name this list box ‘lstSuburbs’.

3. Select the Label tool from the toolbox. Use it to draw a large label at the top of the form and name it ‘lblTitle’. Draw a small label underneath the list box and name this ‘lblCount’.

4. Use the TextBox tool to draw two text boxes next to the list box. Name these text boxes: ‘txtSuburb’ and ‘txtPostcode’.

5. Use the Button tool to create four buttons on the form. Three of the buttons should be placed underneath the text boxes. Name these buttons ‘btnAdd’, ‘btnEdit’ and ‘btnDelete’. Place the last button in the bottom right hand corner and name it ‘btnQuit’.

6. Create a text file called ‘suburbs.txt’ and place it in the ‘bin’ folder of the program. Place the following data into it:

   Melbourne, 3000
   Warrnambool, 3280
   Castlemaine, 3450
   Trafalgar, 3824
   Ballarat, 3350
   Geelong, 3220
   Hopetoun, 3396
   Rainbow, 3424
   Mallacoota, 3889
   Seaspray, 3851
   Yuulong, 3237
   Dimboola, 3414
   Bairnsdale, 3875
   Korumburra, 3950
   Alexandra, 3714

   If you would like to add more suburbs and postcodes than this, a full list can be downloaded from the Australia Post web site.

7. Rename the form to ‘frmSuburbs’. Make sure that you not only change the ‘Name’ property, but that you also change the name in the Solution Explorer and the startup form.

Changing the properties of the objects

Click on each object in turn and set the following properties. Note that properties such as font and colour have not been listed here. Customise these properties as you wish to make the form look well presented.

frmSuburbs

StartPosition: CenterScreen
Text: Suburbs and Postcodes

lblTitle

Text: Suburbs and Postcodes

lblCount

Text: Number of suburbs:

txtSuburb

Text: *Nothing*
Locked: True

txtPostcode

Text: *Nothing*
Locked: True

btnAdd

Text: Add New

btnEdit

Text: Edit

btnDelete

Text: Delete

btnQuit

Text: Quit

Adding code to our program

1. Double click on the form and click in the declarations section (above the 'Private Class frmSuburbs' line). Type in the following 'Imports' statement:

```
Imports System.IO
```

   This statement allows us to use the various file reading and writing commands that we will need to make use of.

2. Double click on the form and type the following code into the frmSuburbs load event:

```
Dim SuburbFile As StreamReader = File.OpenText("suburbs.txt")
Do While SuburbFile.Peek <> -1
      lstSuburbs.Items.Add(SuburbFile.ReadLine())
Loop
SuburbFile.Close()
```

   This code firstly creates a variable to point to the file the wish to open using the 'StreamReader' command. The 'Peek' command can be used to look ahead at what is about to be read in, to see if the end of the file has been reached. If it has, the 'Peek' command returns a value of '-1'. The 'ReadLine' command reads a single line in from the text file and this is added to the list box by using the 'Items.Add' method. Once the contents of the file has been read in, it is closed.

3. Select 'btnQuit' and the 'click' event and type in the following code:

```
Dim SuburbFile As StreamWriter = File.CreateText("suburbs.txt")
For Each Suburb As Object In lstSuburbs.Items
      SuburbFile.WriteLine(Suburb)
Next
SuburbFile.Close()
End
```

   The 'Quit' button will be used to write all of the data that is in the list box, back to the text file, before quitting the program. The code is very similar to that shown above, however this time a 'StreamWriter' variable is declared rather than a 'StreamReader'. The 'For Each' loop cycles through each line in the list box and the 'WriteLine' command writes the line to the text file. Once all of the data has been written, the file is closed and the program ends.

4. Select 'btnAdd' and the 'click' event and type in the following code:

```
Dim SuburbLine As String
SuburbLine = txtSuburb.Text + ", " + txtPostcode.Text
lstSuburbs.Items.Add(SuburbLine)
```

   This code will add what the user has typed into the 'txtSuburb' and 'txtPostcode' boxes, into the list box. Note that the two strings are added together with a comma in the middle.

5. Now select ‘btnDelete’ and the ‘click’ event and enter the following code:

```
lstSuburbs.Items.Remove(lstSuburbs.Text)
```

Removing items from a list box is fairly easy. The ‘Items.Remove’ command will remove the currently selected item.

6. Lastly select the ‘btnEdit’ button and the ‘click’ event and type in the following code:

```
Dim SubLength As Integer
SubLength = Microsoft.VisualBasic.Len(lstSuburbs.Text)
If SubLength > 0 Then
      txtPostcode.Text = _
      Microsoft.VisualBasic.Right(lstSuburbs.Text, 4)
      txtSuburb.Text = _
      Microsoft.VisualBasic.Left(lstSuburbs.Text, SubLength - 6)
      lblCount.Text = "Number of suburbs: " + _
      CStr(lstSuburbs.Items.Count)
End If
lstSuburbs.Items.Remove(lstSuburbs.Text)
```

This is just one in which you could write code to edit the values that have been entered into the list box. This code firstly determines the total length of the selected value. This is required so that the suburb and the postcode can be separated from each other. You will have noticed in the adding function, that the suburb string was added to the postcode string with a comma (and a space) placed between them. This section of code uses the ‘Left’ and ‘Right’ string functions to separate these parts and place them in their respective text boxes. The item being edited is then removed from the list box.

The way this code has been designed, the user would choose to edit an item and it would be removed from the list box and placed in the text boxes. After the user had finished editing the item, they would then press the add button to place it back into the list box once again.

7. Save your program.

Testing the program

Run and test the program and ensure that it behaves as expected. Test that the program loads the suburbs and postcodes from the text file when it is started and writes the information back to the file at the end. Test to see that the ‘add’, ‘edit’ and ‘delete’ functions work correctly and that the program picks up data entered in the incorrect form.

Further ideas to develop

- Add an extension to the program that will sort the suburbs by postcode.