21. VB Programming Fundamentals – Input and Output

21.1 Input and Output Files

We already learnt how to input data from and output result to a VB form. In this lecture, I am going to show you how to read the data from a text file or a database and write your data to a file or database.

Direct File Access

Ever since the first version of VB, files have been processed using Open statements and other related functions. These mechanisms are fully supported in VB 6.0. VB 6.0 also supports a new object called File System Object (FSO) that will eventually replace the direct file functions. However, the direct file access is convenient and easy to use.

File Access Types

A file consists of nothing more than a series of related bytes located on a disk. When your application accesses a file, it must assume what the bytes are supposed to represent (characters, integers, strings, and so on). Depending on what kind of data the file contains, you use the appropriate file access type. There are three types of file access type in VB:

- Sequential – for reading and writing text files in continuous blocks.
- Random – for reading and writing text or binary files structured as fixed-length records.
- Binary – for reading and writing arbitrarily structured files.

We just focus on how to use sequential access in this lecture. Sequential access is designed for use with plain text files and works best for the files that only consist of texts, such as the files created with a typical text editor. Each character in the file is assumed to represent either a text character or a text formatting sequence.

Opening Files for Sequential Access

When you open a file for sequential access, you open it to perform one of the following operations:

- Input – input characters from a file.
- Output – output characters to a file.
- Append – append characters to a file (Append).

To open a file for sequential access, use the following syntax for the Open statement:

Open pathname For [Input | Output | Append] As filenumber [Len = buffersize]

You can use the following functions to read strings from files and write strings to files:

Input #filenimber, variables, …
Write #filenumber, variable, …

21.2 Example

Say still using the tree data in our previous course project. Let's create a text file for input and write your results to another text file.

(1) Use either Dos Editor or Notepad to create a file named TreeData.txt and save it in the directory of your VB project. The data must be delimited by comma.

```
1,27,1
2,13,2
3,12,1
4,15,2.5
5,17,2
6,25,2
7,28,0.5
8,10,1
9,29,2.5
10,13,0.5
```

(2) Start a new VB project and put the following controls on form1 (Figure 1):

a. Two labels
b. A text box
c. A list box
d. Four command buttons

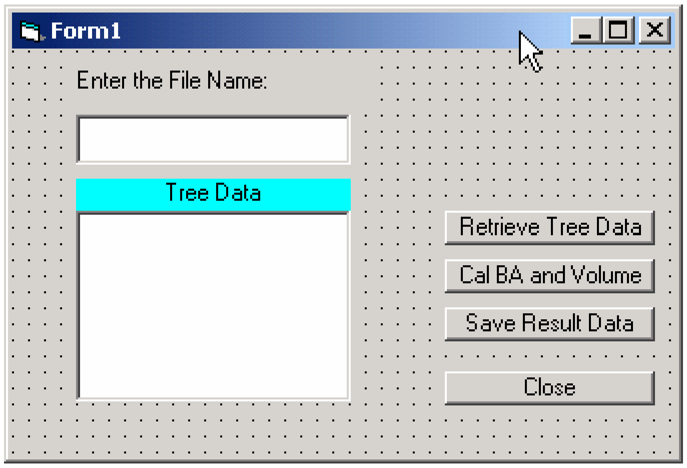

Figure 1. Interface for calculating BA and volume of trees.

(3) Add the following code to the project.

```
'In General Declaration
Dim aryTreeNo(), aryDBH(), aryNLogs()
Dim aryBA(), aryVol()
Dim NofTrees As Integer

'For command button 1
Private Sub Command1_Click()

   Dim Trees As String
   Dim i As Integer


   Trees = Text1.Text
   Trees = App.Path & "\" & Trees
   NofTrees = 0

   If Dir(Trees) <> "" Then
      Open Trees For Input As #1
      While Not EOF(1)
          NofTrees = NofTrees + 1
          Line Input #1, z$
      Wend
      Close #1

      ReDim aryTreeNo(NofTrees), aryDBH(NofTrees), aryNLogs(NofTrees)

      Open Trees For Input As #1
      i = 0
      While Not EOF(1)
          i = i + 1
          Input #1, treeno, dbh, noflogs
          aryTreeNo(i) = treeno
          aryDBH(i) = dbh
          aryNLogs(i) = noflogs
          List1.AddItem treeno & ", " & dbh & ", " & noflogs
      Wend
      Close #1
      Command2.Enabled = True
      Command3.Enabled = True
   Else
      MsgBox Trees & " can not be found!"
   End If

End Sub

'For command button 2
Private Sub Command2_Click()

   Dim j
   ReDim aryBA(NofTrees), aryVol(NofTrees)

   For j = 1 To NofTrees
      aryBA(j) = CalBA(aryDBH(j))
```

```
        aryVol(j) = CalVol(aryDBH(j), aryNLogs(j))
    Next j

    MsgBox "Calculation was done!"

End Sub

‘For command button 3
Private Sub Command3_Click()

    Dim i

    Trees = Text1.Text
    Trees = Mid(Trees, 1, Len(Trees) - 4)
    Trees = Trees + ".rlt"
    Trees = App.Path & "\" & Trees
    Open Trees For Output As #2
    For i = 1 To NofTrees
        Write #2, aryTreeNo(i), aryDBH(i), aryNLogs(i), aryBA(i), aryVol(i)
    Next i
    Close #2
    MsgBox "Results were saved!"

End Sub

‘For command button 4
Private Sub Command4_Click()

    Unload Me

End Sub

‘In General Declaration
Private Function CalBA(dbh As Variant) As Variant

    CalBA = 0.005454154 * dbh * dbh

End Function

Private Function CalVol(d As Variant, l As Variant) As Variant

         CalVol = ((0.55743 * l ^ 2 + 41.51275 * l - 29.37337) + _
                  (2.78043 - 0.04516 * l ^ 2 - 8.77272 * l) * d + _
                  (0.04177 - 0.01578 * l ^ 2 + 0.59042 * l) * d ^ 2)

End Function

‘In General Declaration
Private Sub Form_Load()

    Command2.Enabled = False
    Command3.Enabled = False

End Sub
```

(4) Run your application (Figure 2).

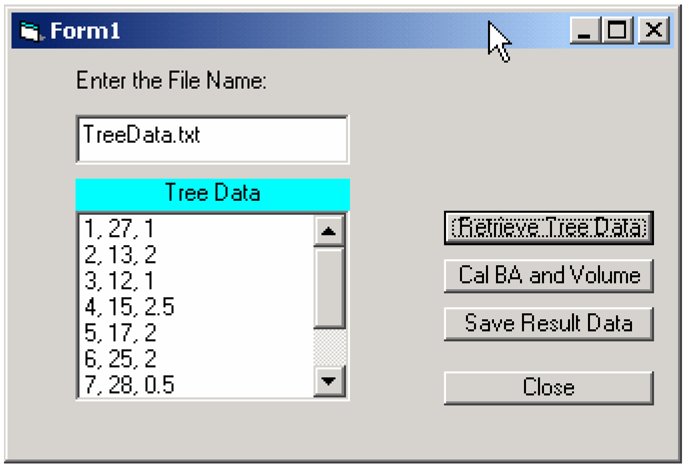

Figure 2. Display tree data and save results.

Your output file is TreeData.rlt that will look like that:

```
1,27,1,3.976078266,299.04855
2,13,2,.921752026,57.54221
3,12,1,.785398176,29.01045
4,15,2.5,1.22718465,105.7070175
5,17,2,1.576250506,136.89841
6,25,2,3.40884625,406.92185
7,28,.5,4.276056736,207.3396425
8,10,1,.5454154,13.96331
9,29,2.5,4.586943514,707.8604575
10,13,.5,.921752026,26.7814175
```