


Files

There are two types of files that can be created: text files and binary files.


Text files have variable length records. (One line of text may have 20 characters, the next line 5, etc.) A text file must be read sequentially: line 1, then line 2, etc. Text files are also called sequential files. We will learn about text files in this lesson.

Each record in a **binary file** must be exactly the same size. Because all of the records are the same length, a binary file can be read either sequentially or randomly. Suppose each record has exactly 20 bytes. The location of the first record in the file is always known. The second record begins 20 bytes past that. The fifth record begins 80 bytes past the beginning of the file. If the position where each record starts can be calculated, then we can read the Nth record without reading all of the records before it. Binary files are also called **Random Access** files.


The “To-Do List” Program

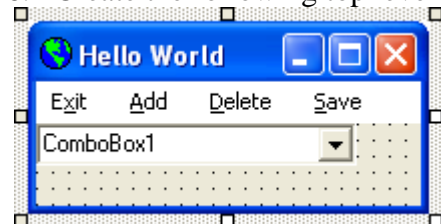
This program reads and writes to a sequential file. Each line in the file is added to the ComboBox  and displayed.



1. Using Notepad, create a short list of about 5 items and save it as C:\todo.txt.
2. Start a new Windows application, name it ToDo.
3. Add a ComboBox  to the form. Name it cboToDo.
4. Add the following code to form load:

```
Private Sub Form1_Load(ByVal sender As System.Object, _
    ByVal e As System.EventArgs) Handles MyBase.Load
    Dim S As String
    FileOpen(1, "C:\todo.txt", OpenMode.Input)
    While Not EOF(1)
        Input(1, S)
        Me.ComboBox1.Items.Add(S)
    End While
    FileClose(1)
    If Me.ComboBox1.Items.Count > 0 Then
        Me.ComboBox1.SelectedIndex = 0
    End If
End Sub 'Form1_Load
```

5. Add a MainMenu  to the project.
6. Create the following top level menu items: mnuExit, mnuAdd, mnuDelete, mnuSave.



We will use general procedures to add, save, and delete so that we can call them from a toolbar later.

7. Add gets an item from the user with an input box. The item is added only if it is not already in the combo box. After an item is added it is displayed in the text of the combo box.

```
Public Sub AddItem()
    Dim Item As String
    Dim P As Integer
    Item = InputBox("Task to add:", "Add")
    If Item <> "" Then
        P = Me.ComboBox1.FindString(Item) 'is item already in box?
        If P = -1 Then 'add if not there already
            Me.ComboBox1.Items.Add(Item)
            P = Me.ComboBox1.FindString(Item)
        End If
        Me.ComboBox1.SelectedIndex = P 'display added item
    End If
End Sub 'AddItem

Private Sub mnuAdd_Click(ByVal sender As Object, ByVal e As System.EventArgs) _
    Handles mnuAdd.Click
    AddItem()
End Sub 'mnuAdd_Click
```

8. Compare the method to save the file with the one to with the one to read it:

```
Private Sub SaveList()
    Dim S As String
    Dim Num As Integer
    FileOpen(1, "C:\todo.txt", OpenMode.Output)
    For Num = 0 To Me.ComboBox1.Items.Count - 1
        S = Me.ComboBox1.Items.Item(Num)
        PrintLine(1, S)
    Next Num
    FileClose(1)
    MsgBox("Items saved: " & Me.ComboBox1.Items.Count)
End Sub 'SaveList

Private Sub mnuSave_Click(ByVal sender As Object, ByVal e As System.EventArgs) _
    Handles mnuSave.Click
    SaveList()
End Sub 'mnuSave_Click
```

9. Deleting takes a little extra work because we have to make sure that we don't try to delete after the list is already empty. After an item is deleted, we would like to display the first item. Again, we can't display the first item if we just deleted the last item.

```
Private Sub DeleteItem()
    If Me.ComboBox1.Items.Count > 0 Then
        Me.ComboBox1.Items.RemoveAt(Me.ComboBox1.SelectedIndex)
    Else
        Me.ComboBox1.Text = ""
    End If
    If Me.ComboBox1.Items.Count > 0 Then
        Me.ComboBox1.SelectedIndex = 0
    End If
End Sub 'DeleteItem
```

```
Private Sub mnuDelete_Click(ByVal sender As Object, ByVal e As System.EventArgs) _
    Handles mnuDelete.Click
    DeleteItem()
End Sub 'mnuDelete_Click
```

Run, test, and save. You can look at the file after you create it, using notepad.

Do You Want to Save?

We have not written any code for Exit yet. Instead of just using End for this procedure, we will give the user a chance to save if they haven't saved since the last change. Almost every program will remind you if you didn't save. The additions below will implement the "Save Reminder". A variable Changed will be used to keep track of whether there have been changes since the last save:

- ◆ Set Changed to False when a new file is opened (no changes yet),
- ◆ Set Changed to False when the file is saved.
- ◆ Set Changed to True when you add or delete an item.
- ◆ When you end the program, display a message if Changed is true.

The entire program is shown at this point, with the new changes in bold:

```
Public Class Form1
    Inherits System.Windows.Forms.Form
    Dim Changed As Boolean
    Windows Form Designer generated code
    Private Sub Form1_Load(ByVal sender As System.Object, _
        ByVal e As System.EventArgs) Handles MyBase.Load
        Dim S As String
        FileOpen(1, "C:\todo.txt", OpenMode.Input)
        While Not EOF(1)
            Input(1, S)
            Me.ComboBox1.Items.Add(S)
        End While
        FileClose(1)
        If Me.ComboBox1.Items.Count > 0 Then
            Me.ComboBox1.SelectedIndex = 0
        End If
        Changed = False
    End Sub 'Form1_Load

    Public Sub AddItem()
        Dim Item As String
        Dim P As Integer
        Item = InputBox("Task to add:", "Add")
        If Item <> "" Then
            P = Me.ComboBox1.FindString(Item) 'is item already in box?
            If P = -1 Then 'add if not there already
                Me.ComboBox1.Items.Add(Item)
                P = Me.ComboBox1.FindString(Item)
                Changed = True
            End If
            Me.ComboBox1.SelectedIndex = P 'display added item
        End If
    End Sub 'AddItem

    Private Sub mnuAdd_Click(ByVal sender As Object, ByVal e As System.EventArgs)_
        Handles mnuAdd.Click
        AddItem()
    End Sub 'mnuAdd_Click
```

```
Private Sub SaveList()  
    Dim S As String  
    Dim Num As Integer  
    FileOpen(1, "C:\todo.txt", OpenMode.Output)  
    For Num = 0 To Me.ComboBox1.Items.Count - 1  
        S = Me.ComboBox1.Items.Item(Num)  
        PrintLine(1, S)  
    Next Num  
    FileClose(1)  
    Changed = False  
    MsgBox("Items saved: " & Me.ComboBox1.Items.Count)  
End Sub 'SaveList  
  
Private Sub mnuSave_Click(ByVal sender As Object, ByVal e As System.EventArgs) _  
    Handles mnuSave.Click  
    SaveList()  
End Sub 'mnuSave_Click  
  
Private Sub DeleteItem()  
    If Me.ComboBox1.Items.Count > 0 Then  
        Me.ComboBox1.Items.RemoveAt(Me.ComboBox1.SelectedIndex)  
        Changed = True  
    Else  
        Me.ComboBox1.Text = ""  
    End If  
    If Me.ComboBox1.Items.Count > 0 Then  
        Me.ComboBox1.SelectedIndex = 0  
    End If  
End Sub 'DeleteItem  
  
Private Sub mnuDelete_Click(ByVal sender As Object, ByVal e As System.EventArgs) _  
    Handles mnuDelete.Click  
    DeleteItem()  
End Sub 'mnuDelete_Click  
  
Private Sub AskToSave()  
    Dim Answer As MsgBoxResult  
    If Changed Then  
        Answer = MsgBox("Do you want to save?", _  
            MsgBoxStyle.YesNoCancel + MsgBoxStyle.Question, "Save")  
        If Answer = MsgBoxResult.Yes Then  
            SaveList()  
        End  
        End If  
        If Answer = MsgBoxResult.No Then End  
        'Do nothing if they cancel!  
    Else  
        End  
    End If  
End Sub 'AskToSave  
  
Private Sub mnuExit_Click(ByVal sender As Object, ByVal e As System.EventArgs) _  
    Handles mnuExit.Click  
    AskToSave()  
End Sub  
End Class
```

This will work fine as long as the user clicks the Exit button to save. What happens if they close the program using the X on the right, or some other way?

OnClosing

The OnClosing procedure is invoked whenever the form is closed any other way. If the argument **e.Cancel** is set to True, the form is not closed.

1. Add a global variable canceled:

```
Inherits System.Windows.Forms.Form
Dim Changed As Boolean
Dim Canceled As Boolean
```

2. In AskToSave replace the line that says **'Do nothing if they cancel!** With:


```
If Answer = MsgBoxResult.Cancel Then Canceled = True
```



3. Write the OnClosing procedure:



```
Protected Overrides Sub OnClosing(ByVal e As System.ComponentModel.CancelEventArgs)
    Canceled = False
    AskToSave()
    If Canceled Then e.Cancel = True 'Program does NOT end!
End Sub 'OnClosing
```

Run, test, save!

Experiment: Add a toolbar.


You can find the Save picture  in ...Graphics/Bitmaps/Tlbr_W95/Save.bmp.

You can find  and  in ...Graphics/Icons/Misc as MISC05.ICO and MISC06.ICO.

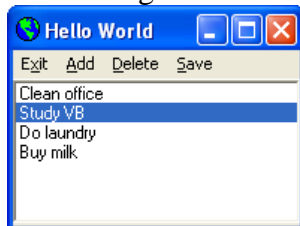
You can find  and  in ...Graphics/Icons/Traffic as TRFFC13.ICO and TRFFC14.ICO.

ListBox

The ListBox  is almost identical to the ComboBox except the ComboBox shows just one item at a time and the ListBox shows several.

Delete the ComboBox and add a ListBox .

In the code, change every instance of ComboBox1 with ListBox1. Absolutely nothing else needs to be changed.







If you want, you can make the list box fill the form, make sure the location is 0,0:

```
Protected Overrides Sub OnSizeChanged(ByVal e As System.EventArgs)
    Me.ListBox1.Width = Me.Width
    Me.ListBox1.Height = Me.Height
End Sub 'OnSizeChanged
```

Experiment: Both ListBoxes and ComboBoxes have a Sorted property. Try setting Sorted to true.

RichTextBox

The RichTextBox  control has built-in methods to read and write a file.

1. Start a new Windows application and name it MyNotes.
2. Add a MainMenu  control and create top level menu items to mnuExit, mnuOpen and mnuSave.
3. Add a RichTextBox .
4. Add an OpenFileDialog .
5. A bare minimum of code is provided for you. Try adding the same features that were included in the ToDo program.

```
Private Sub mnuOpen_Click(ByVal sender As Object, ByVal e As _
    System.EventArgs) Handles mnuOpen.Click
    OpenFileDialog1.Filter = "Text|*.txt"
    OpenFileDialog1.ShowDialog()
    If OpenFileDialog1.FileName <> "" Then
        Me.RichTextBox1.LoadFile(OpenFileDialog1.FileName, _
            RichTextBoxStreamType.PlainText)
    End If
End Sub 'mnuOpen_Click

Private Sub mnuSave_Click(ByVal sender As Object, ByVal e As _
    System.EventArgs) Handles mnuSave.Click
    Me.RichTextBox1.SaveFile(Me.OpenFileDialog1.FileName, _
        RichTextBoxStreamType.PlainText)
End Sub 'mnuSave_Click
```

With

In the code for mnuOpen the object OpenFileDialog1 is referred to several times. The with statement lets you refer to an object just once. After naming the object in the With statement, you can type a dot to open a drop-down list just as though you have typed the name and then the dot. The code would look like this:

```
Private Sub mnuOpen_Click(ByVal sender As Object, ByVal e As _
    System.EventArgs) Handles mnuOpen.Click
    With OpenFileDialog1
        .Filter = "Text|*.txt"
        .ShowDialog()
        If .FileName <> "" Then
            Me.RichTextBox1.LoadFile(.FileName, _
                RichTextBoxStreamType.PlainText)
        End If
    End With
End Sub 'mnuOpen_Click
```

Experiment: Add a SaveFileDialog  and implement a “Save As” procedure.