8

Text Files

8.1 Managing Text Files

This section presents efficient ways to sort, search, reorganize, combine, and retrieve information from text files. The section begins with some preliminaries. You can skip them if you have read Section 7.3.

Preliminaries

The text files considered in Sections 7.1 and 7.2 have a single piece of data per line. For instance, each line of the file States.txt contains the name of a state, and each line of the file USPres.txt contains the name of a president. Another type of text file, called a CSV formatted file, has several items of data on each line with the items separated by commas. (CSV stands for *Comma Separated Values*.) An example is the file USStates.txt, where each line contains four items for each state—*name*, *abbreviation*, *land area* (in square miles), and *population in the year 2000*. The first four lines of the file are

```
Delaware,DE,1954,759000
Pennsylvania,PA,44817,12296000
New Jersey,NJ,7417,8135000
Georgia,GA,57906,7637000
```

Each line of the file is called a **record**, and each of the four categories of data is a **field.** That is, each record consists of four fields—a name field, an abbreviation field, an area field, and a population field.

The Split method is used to access the fields of CSV formatted files. For instance, if the string variable *line* holds the first record of the file USStates.txt, then the value of `line.Split(",c)(0)` is Delaware, the value of `line.Split(",c)(1)` is DE, the value of `CInt(line.Split(",c)(2))` is the number 1954, and the value of `CInt(line.Split(",c)(3))` is the number 759000.

The following code can be used to create a LINQ query holding the contents of the file USStates.txt:

```
Dim states() As String = IO.File.ReadAllLines("USStates.txt")
Dim query = From line In states
            Let data = line.Split(",c)
            Let name = data(0)
            Let abbr = data(1)
            Let area = CInt(data(2))
            Let pop = CInt(data(3))
            Select name, abbr, area, pop
```

After this code is executed, *query* will be a sequence of elements, each element consisting of four components. If the variable *state* is assigned one of the elements, then the four components associated with *state* are denoted *state.name*, *state.abbr*, *state.area*, and *state.pop*. For instance, the following code fills a list box with the names of the states.

```
For Each state In query
  lstBox.Items.Add(state.name)
Next
```

The Select clause in the query above contains four items. A Select clause can contain any number of items. When a Select clause contains just one item, the sequence returned by the query can be displayed in a list box. If a Select clause of a query contains two or more items, all the values returned by the query can be displayed in a table by using a DataGridView control. (The DataGridView control is found in the Toolbox's *All Windows Forms* and *Data* groups.)

The standard prefix for the name of a DataGridView control is *dgv*. With the query above, the statements

```
dgvStates.DataSource = query.ToList
dgvStates.CurrentCell = Nothing
```

display the values returned by the query in the grid shown in Fig. 8.1. (***Note:*** The second statement is optional. It prevents having a shaded cell in the grid.) Visual Basic automatically generates the column headings.

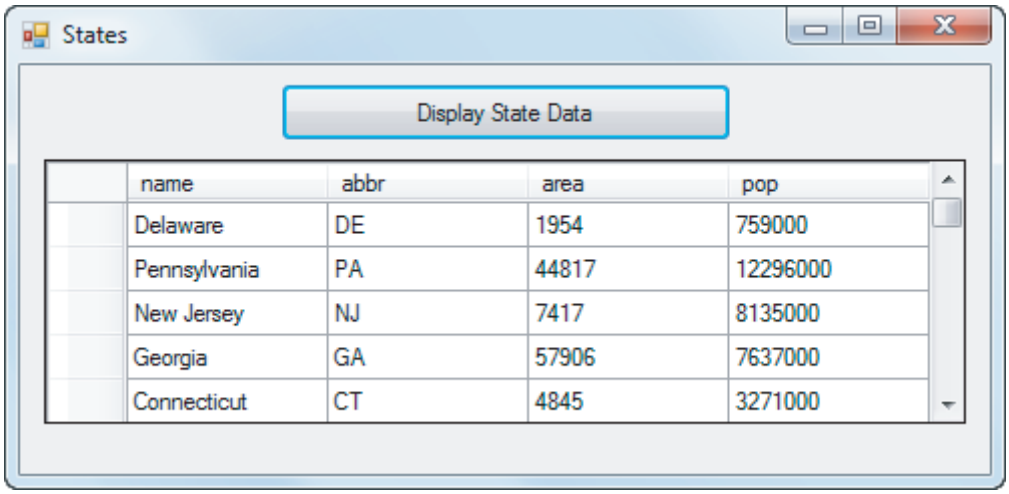


FIGURE 8.1 **Displaying a table in a DataGridView control.**

The blank column at the left side of the DataGridView control can be removed by setting the RowHeadersVisible property of the control to False. The column headers in the DataGridView control can be customized with code that sets the HeaderText property. Figure 8.2 results when the RowHeadersVisible property is set to False at design time and the following four lines of code are added to the two lines of code above:

```
dgvStates.Columns("name").HeaderText = "State"
dgvStates.Columns("abbr").HeaderText = "State Abbreviation"
dgvStates.Columns("area").HeaderText = "Land Area"
dgvStates.Columns("pop").HeaderText = "Population"
```

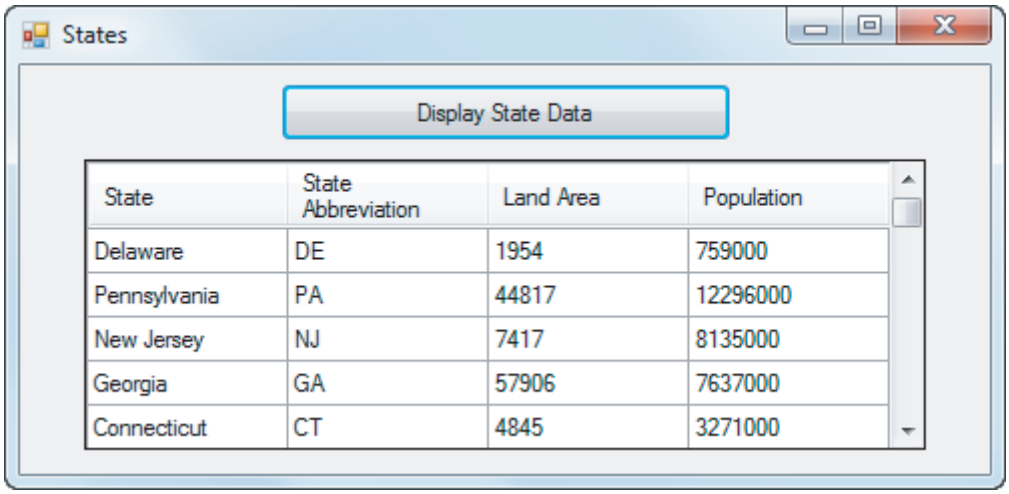


FIGURE 8.2 **States table with modified headers.**

Note: The DataGridView controls appearing in this textbook have been carefully sized to exactly fit the data. Comments 2 and 3 of Section 7.3 explain how this was accomplished. There is no need for you to strive for such precision when working the exercises.

VideoNote
Managing text files

WriteAllLines Method

In Chapter 7 we copied the contents of files into arrays with the ReadAllLines method. We can reverse the process with the WriteAllLines method. The following line of code creates a new text file and copies the contents of a string array (or a LINQ query that returns string values) into the file, placing one element on each line.

```
IO.File.WriteAllLines("fileName.txt", strArrayOrQueryName)
```

A simple numeric array can be copied into a text file by first converting the array to a string array and then copying the string array into a file using either of the following two sets of code:

```
Dim upperBound As Integer = numArray.Count - 1
Dim strArray(upperBound) As String
For i As Integer = 0 To upperBound
  strArray(i) = CStr(numArray(i))
Next
IO.File.WriteAllLines("fileName.txt", strArray)
```

```
Dim query = From num In numArray
            Select CStr(num)
IO.File.WriteAllLines("fileName.txt", query)
```

Note: If the WriteAllLines method references an existing file, the file will be overwritten.

Sorting a Text File

Any text file can easily be sorted with a LINQ query.

Example 1 The first four lines of the file AgeAtInaug.txt are

```
George Washington,57
John Adams,61
Thomas Jefferson,57
James Madison,57
```

Each of the 44 lines in the file contains a president's name and his age at inauguration. The following program orders the data in the file by the age at inauguration and creates a new sorted file.

```
Private Sub btnSort_Click(...) Handles btnSort.Click
  'Sort the file AgeAtInaug.txt by ages
  Dim agesAtInaug() As String = IO.File.ReadAllLines("AgeAtInaug.txt")
  Dim query = From line In agesAtInaug
              Let age = CInt(line.Split(","c)(1))
              Order By age
              Select line
  IO.File.WriteAllLines("Sorted.txt", query)
End Sub
```

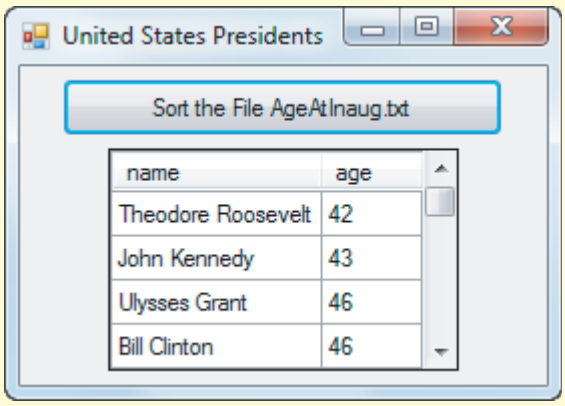
[Run, click on the button, and terminate the program. Then click on the *Refresh* button in the Solution Explorer window, click on the *View All Files* button, and double-click on the new text file Sorted.txt in the *bin**Debug* subfolder. The first four lines of the file are as follows.]

```
Theodore Roosevelt,42
John Kennedy,43
Ulysses Grant,46
Bill Clinton,46
```

Example 2 The following variation of Example 1 displays the data in the file AgeAtInaug.txt in a table ordered by the ages at inauguration.

```
Private Sub btnSort_Click(...) Handles btnSort.Click
  'Sort the file AgeAtInaug.txt by ages
  Dim agesAtInaug() As String = IO.File.ReadAllLines("AgeAtInaug.txt")
  Dim query = From line In agesAtInaug
              Let name = line.Split(","c)(0)
              Let age = CInt(line.Split(","c)(1))
              Order By age
              Select name, age
  dgvOutput.DataSource = query.ToList
  dgvOutput.CurrentCell = Nothing
End Sub
```

[Run, and click on the button.]



Reorganizing the Data in a CSV Text File

LINQ can retrieve specific data from a file and use it to create a new file containing that data.

Example 3 The first four lines of the file Justices.txt are

```
Samuel,Alito,George W. Bush,NJ,2006,0
Henry,Baldwin,Andrew Jackson,PA,1830,1844
Philip,Barbour,Andrew Jackson,VA,1836,1841
Hugo,Black,Franklin Roosevelt,AL,1937,1971
```

Each line of the file contains the following information about a Supreme Court justice: first name, last name, appointing president, state from which they were appointed, year appointed, and year they left the court. (For sitting justices, the last field is set to 0.) The following program creates a new file, where each line is the full name of a justice and the year they were appointed. The justices are sorted by the year appointed in ascending order. Justices appointed during the same year are sorted by their first name in ascending order.

```
Private Sub btnReorganize_Click(...) Handles btnReorganize.Click
  'Take data from a file. Sort and restructure the data,
  'and write it to a new file.
  Dim justices() As String = IO.File.ReadAllLines("Justices.txt")
  Dim query = From line In justices
              Let data = line.Split(",c)
              Let firstName = data(0)
              Let lastName = data(1)
              Let yrAppointed = CInt(data(4))
              Order By yrAppointed, firstName
              Let newLine = firstName & " " & lastName & "," & yrAppointed
              Select newLine
  IO.File.WriteAllLines("NewFile.txt", query)
End Sub
```

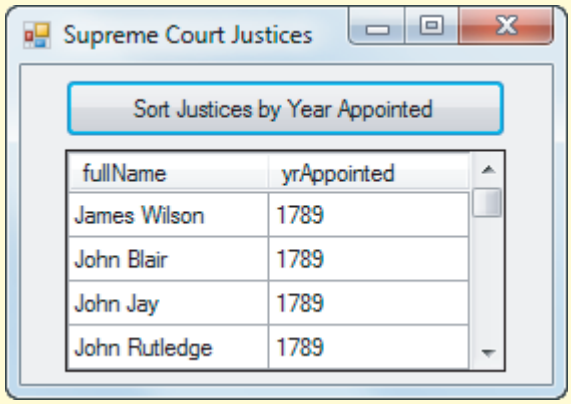
[Run, click on the button, and terminate the program. Then click on the *Refresh* button in the Solution Explorer window, click on the *View All Files* button, and double-click the new text file in the *bin**Debug* subfolder. The first four lines of the file are as follows.]

```
James Wilson,1789
John Blair,1789
John Jay,1789
John Rutledge,1789
```

Example 4 The following variation of Example 3 displays the requested information in a table.

```
Private Sub btnReorganize_Click(...) Handles btnReorganize.Click
  'Take data from a text file. Sort and restructure the data,
  'and display it in a table.
  Dim justices() As String = IO.File.ReadAllLines("Justices.txt")
  Dim query = From line In justices
              Let data = line.Split(",c)
              Let firstName = data(0)
              Let lastName = data(1)
              Let yrAppointed = CInt(data(4))
              Let fullName = firstName & " " & lastName
              Order By yrAppointed, firstName
              Select fullName, yrAppointed
  dgvOutput.DataSource = query.ToList
  dgvOutput.CurrentCell = Nothing
End Sub
```

[Run, and click on the button.]



Set Operations

Often we want to create a new text file from two existing text files. For instance, we might want to merge the two files (with or without duplications). Or, we might want to update one file by deleting the items that also appear in the other file. Or, we might want the new file to contain the items that appear in both of the existing files. The steps we will use to carry out such operations are as follows:

1. Use the ReadAllLines method to fill two arrays with the contents of the two existing text files.
2. Apply a set operation such as Concat, Union, Intersect, or Except to the arrays or to LINQ queries derived from the arrays.
3. Use the WriteAllLines method to write the resulting array into a new text file.

If *array1* and *array2* are string arrays, then

array1.Concat(array2).ToArray is an array containing the elements of *array1* with *array2* appended, possibly with duplications.

array1.Union(array2).ToArray is an array containing the elements of *array1* with *array2* appended, without duplications.

array1.Intersect(array2).ToArray is an array containing the elements that are in both *array1* and *array2*.

array1.Except(array2).ToArray is an array containing the elements of *array1* with the elements of *array2* removed.

Note: When one of the four operations above is used as the second parameter of a WriteAllLines method, the ToArray method can be omitted.

These four set operations are demonstrated with two simple files in Example 5 and then with more complex files in Example 6.

Example 5 The contents of two files are as follows:

File1.txt	File2.txt
Alpha	Bravo
Bravo	Delta
Charlie	

The following program combines these two files in four ways. Figure 8.3 shows the form for the program.

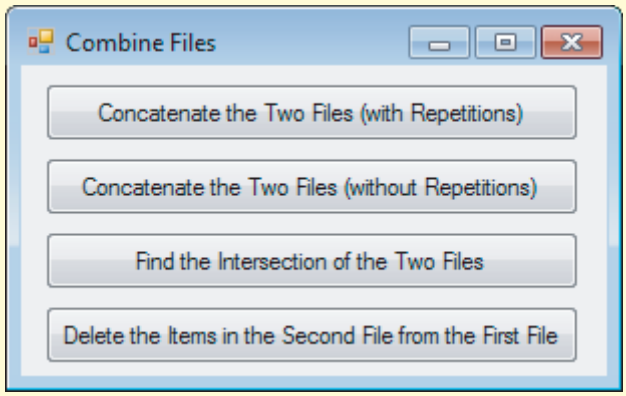


FIGURE 8.3 Form for Example 5.

```
Dim firstSet() As String = IO.File.ReadAllLines("File1.txt")
Dim secondSet() As String = IO.File.ReadAllLines("File2.txt")

Private Sub btnConcat_Click(...) Handles btnConcat.Click
  IO.File.WriteAllLines("Concat.txt", firstSet.Concat(secondSet))
End Sub

Private Sub btnUnion_Click(...) Handles btnUnion.Click
  IO.File.WriteAllLines("Union.txt", firstSet.Union(secondSet))
End Sub

Private Sub btnIntersect_Click(...) Handles btnIntersect.Click
  IO.File.WriteAllLines("Intersect.txt", firstSet.Intersect(secondSet))
End Sub

Private Sub btnExcept_Click(...) Handles btnExcept.Click
  IO.File.WriteAllLines("Except.txt", firstSet.Except(secondSet))
End Sub
```

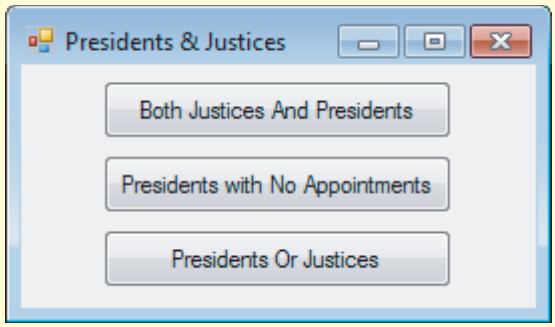
[Run, click on each button, and terminate the program. Then click on the *Refresh* button in the Solution Explorer window, click on the *View All Files* button, and look at the new text files in the *bin**Debug* subfolder.]

The file Concat.txt contains the five words Alpha, Bravo, Charlie, Bravo, and Delta.
The file Union.txt contains the four words Alpha, Bravo, Charlie, and Delta.
The file Intersect.txt contains the single word Bravo.
The file Except.txt contains the two words Alpha and Charlie.

In the four set operations, one or both of the arrays can be replaced with LINQ queries. This allows the programmer to order, filter, and project the data before combining files.

Example 6 The following program demonstrates the set operations with the two files Justices.txt and USPres.txt. The file USPres.txt is a single-field text file containing the names of the 44 presidents. The first four lines of USPres.txt contain the names George Washington,

John Adams, Thomas Jefferson, and James Madison. ***Note:*** The file Justices.txt shows all Supreme Court appointments as of January 1, 2010.



OBJECT	PROPERTY	SETTING
frmPnJ	Text	Presidents & Justices
btnBoth	Text	Both Justices And Presidents
btnNo	Text	Presidents with No Appointments
btnOr	Text	Presidents Or Justices

```
Dim justices() As String = IO.File.ReadAllLines("Justices.txt")
Dim presidents() As String = IO.File.ReadAllLines("USPres.txt")

Private Sub btnBoth_Click(...) Handles btnBoth.Click
  'Display justices who were also presidents
  Dim queryJustices = From line In justices
                      Let firstName = line.Split(","c)(0)
                      Let lastName = line.Split(","c)(1)
                      Let fullName = firstName & " " & lastName
                      Select fullName
  IO.File.WriteAllLines("Both.txt", queryJustices.Intersect(presidents))
End Sub

Private Sub btnNoAppointments_Click(...) Handles btnNoAppointments.Click
  'Display presidents who made no Supreme Court appointments
  Dim queryAppointers = From line In justices
                        Let president = line.Split(","c)(2)
                        Select president
  Dim queryNoAppoint = From pres In presidents.Except(queryAppointers)
                       Order By pres
                       Select pres
  IO.File.WriteAllLines("NoAppointments.txt", queryNoAppoint)
End Sub

Private Sub btnOr_Click(...) Handles btnOr.Click
  'Display a combined list of presidents and justices
  Dim queryJustices = From line In justices
                      Let firstName = line.Split(","c)(0)
                      Let lastName = line.Split(","c)(1)
                      Let fullName = firstName & " " & lastName
                      Select fullName
  Dim queryEither = From person In presidents.Union(queryJustices)
                    Order By person
                    Select person
  IO.File.WriteAllLines("PresOrJustice.txt", queryEither)
End Sub
```

[Run, click on each button, and terminate the program. Then click on the *Refresh* button in the Solution Explorer window, click on the *View All Files* button, and look at the new text files in the *bin**Debug* subfolder.]

The file Both.txt contains the single name "William Taft".
The file NoAppointments.txt contains the names of four presidents.
The file PresOrJustice.txt contains 154 names.

Searching a CSV Text File

In Section 7.3 we searched files having several fields by loading the data into arrays of structures and searching the arrays. The next example provides a more direct way of searching for data—no user-defined structure is used.

Example 7 Each record of the file USStates.txt contains a name field, an abbreviation field, an area field, and a population field. The first two records of the file are

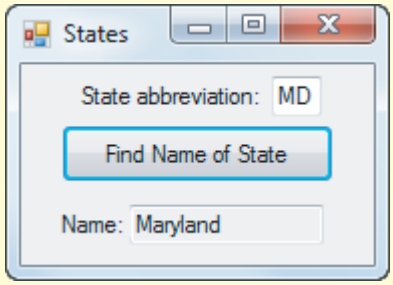
```
Delaware,DE,1954,759000
Pennsylvania,PA,44817,12296000
```

The following program looks up the name of the state whose abbreviation is given. ***Note:*** The method First is required in the line `txtName.Text = query.First` because query is a sequence—namely, a sequence of one item.

```
Dim states() As String = IO.File.ReadAllLines("USStates.txt")

Private Sub btnFind_Click(...) Handles btnFind.Click
  'Note: mtbAbbr has the mask LL
  Dim query = From line In states
              Let name = line.Split(","c)(0)
              Let abbreviation = line.Split(","c)(1)
              Where abbreviation = mtbAbbr.Text.ToUpper
              Select name
  If query.Count = 1 Then
    txtName.Text = query.First
  Else
    Dim str As String = " is not a valid state abbreviation."
    MessageBox.Show(mtbAbbr.Text.ToUpper & str, "Error")
    mtbAbbr.Clear()
    mtbAbbr.Focus()
  End If
End Sub
```

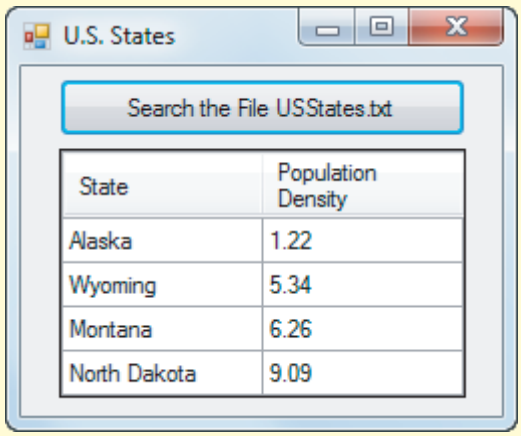
[Run, enter a state abbreviation into the masked text box, and click on the button.]



Example 8 The following program searches the file USStates.txt for the states having population density less than 9.5 people per square mile and displays the names of the states and their population densities ordered by their population densities (in ascending order) in a table.

```
Private Sub btnSearch_Click(...) Handles btnSearch.Click
  Dim states() As String = IO.File.ReadAllLines("USStates.txt")
  Dim query = From line In states
              Let data = line.Split(","c)
              Let name = data(0)
              Let popDensity = CDbl(data(3)) / CDbl(data(2))
              Let formattedDensity = FormatNumber(popDensity)
              Where popDensity < 9.5
              Order By popDensity Ascending
              Select name, formattedDensity
  dgvOutput.DataSource = query.ToList
  dgvOutput.CurrentCell = Nothing
  dgvOutput.Columns("name").HeaderText = "State"
  dgvOutput.Columns("formattedDensity").HeaderText = "Population Density"
End Sub
```

[Run, and click on the button.]



The OpenFileDialog Control

Windows applications, such as Word, Excel, and Notepad, all provide the same standard Open dialog box to help you specify the file you want to open. Figure 8.4 shows an Open dialog box that could be used with Notepad to open a text file. The same Open dialog box, with all its functionality, is available to Visual Basic programs courtesy of the OpenFileDialog control.

The OpenFileDialog control is in the *Dialogs* section of the Toolbox. When you double-click on the icon in the Toolbox, the control will appear in the component tray below the Form Designer with the default name OpenFileDialog1. (We will not change the name, since the default name completely describes the control's function.) The only property we will set for the control is the Filter property, which determines the text that appears in the combo box above

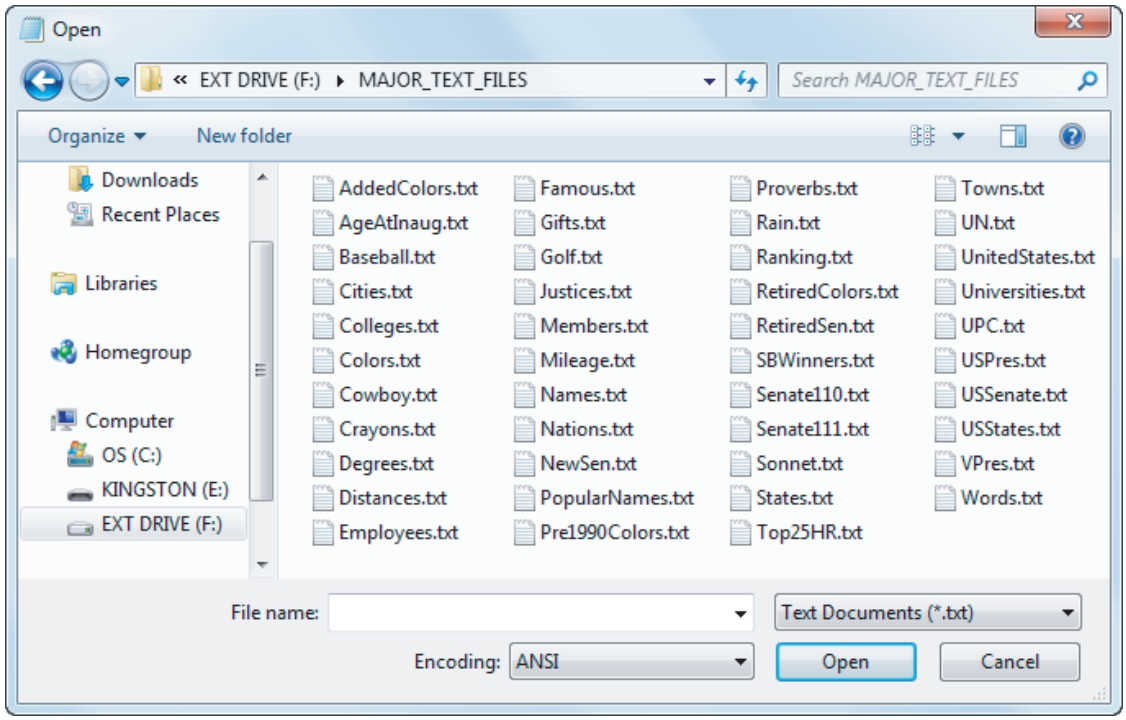


FIGURE 8.4 An Open dialog box.

the *Open* button and the types of files that will be displayed in the dialog box. The simplest setting has the form

text for combo box|*.*ext*

where *ext* is a two- or three-letter extension describing the types of files to display. For our purposes, the most used setting for the Filter property will be

```
Text Documents (*.txt)|*.txt
```

The statement

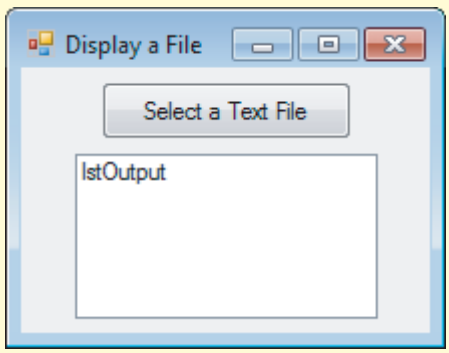
```
OpenFileDialog1.ShowDialog()
```

displays the Open dialog box. After a file has been selected and the *Open* button pressed, the value of

```
OpenFileDialog1.FileName
```

will be the file's filespec—including drive, path, filename, and extension.

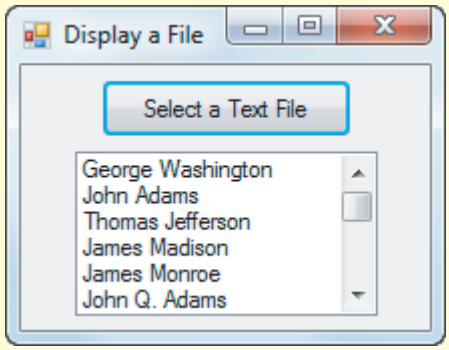
Example 9 The following program displays the contents of a text file selected by the user with an Open dialog box.



OBJECT	PROPERTY	SETTING
frmDisplayFile	Text	Display a File
btnSelect	Text	Select a Text File
lstOutput		
OpenFileDialog1	Filter	Text Documents (*.txt)|*.txt

```
Private Sub btnSelect_Click(...) Handles btnSelect.Click
  Dim textFile As String
  OpenFileDialog1.ShowDialog() 'Open dialog box appears and program
  '                            pauses until a selection is made
  textFile = OpenFileDialog1.FileName
  lstOutput.DataSource = IO.File.ReadAllLines(textFile)
  lstOutput.SelectedItem = Nothing
End Sub
```

[Run, and click on the button. (Assume that the user makes choices leading to the situation in Fig. 8.4.) Select the file USPres.txt, and click on the *Open* button in the dialog box.]



Comments 8.1

1. In Example 7, the array *states* could have been omitted and the first line of the query changed to

   ```
   Dim query = From line In IO.File.ReadAllLines("USStates.txt")
   ```

   However, with this change the text file would be read each time the user requested the name of a state. Ideally, a file should be read only once from a disk each time the program is run.

Practice Problems 8.1

1. Consider Example 7. Suppose the last line of the query were changed to

   ```
   Select name, abbr
   ```

   What change would have to be made to the following line?

   ```
   txtName.Text = query.First
   ```

2. The second clause of the query in Example 1 is

   ```
   Let age = CInt(line.Split(",")(1))
   ```

   The program would produce the correct result even if the CInt function were omitted. Why is that so, and why should CInt be used in general?

3. Does the array `array1.Concat(array2).ToArray` contain the same set of values (disregarding order) as the array `array2.Concat(array1).ToArray`? Is the same true for Union, Intersect, and Except?

EXERCISES 8.1

Exercises 1 through 10 refer to the file Justices.txt that contains data about the Supreme Court justices, past and present. Each record contains six fields—first name, last name, appointing president, home state, year appointed, and year they left the court. (For sitting judges, the last field is set to 0.) The first two lines of the file are as follows:

```
Samuel,Alito,George W. Bush,NJ,2006,0
Henry,Baldwin,Andrew Jackson,PA,1830,1844
```

In Exercises 1 through 6, determine the first two lines of the new file created by the code.

1.
```
Dim query = From line In IO.File.ReadAllLines("Justices.txt")
            Let data = line.Split(","c)
            Let firstName = data(0)
            Let lastName = data(1)
            Let fullName = firstName & " " & lastName
            Let state = data(3)
            Select fullName & "," & state
IO.File.WriteAllLines("NewFile.txt", query)
```

2.
```
Dim query = From line In IO.File.ReadAllLines("Justices.txt")
            Let data = line.Split(","c)
            Let firstName = data(0)
            Let lastName = data(1)
            Let fullName = (firstName & " " & lastName).ToUpper
            Let yrAppointed = CInt(data(4))
            Select fullName & " was appointed in " & yrAppointed & "."
IO.File.WriteAllLines("NewFile.txt", query)
```

3.
```
Dim query = From line In IO.File.ReadAllLines("Justices.txt")
            Let lastName = line.Split(","c)(1)
            Let pres = line.Split(","c)(2)
            Let presLastName = pres.Split(" "c).Last
            Let phrase = lastName & " was appointed by " & presLastName
            Select phrase
IO.File.WriteAllLines("NewFile.txt", query)
```

4.
```
'Note: Today.Year is the current year
Dim query = From line In IO.File.ReadAllLines("Justices.txt")
            Let lastName = line.Split(","c)(1)
            Let yrAppointed = CInt(line.Split(","c)(4))
            Let years = Today.Year - yrAppointed
            Let phrase = lastName & " appointed " & years & " years ago"
            Select phrase
IO.File.WriteAllLines("NewFile.txt", query)
```

5.
```
Dim query = From line In IO.File.ReadAllLines("Justices.txt")
            Let data = line.Split(","c)
            Let firstName = data(0)
            Let lastName = data(1)
            Let yrAppointed = CInt(data(4))
```

```
                Let newLine = lastName & "," & firstName & "," & yrAppointed
                Select newLine
    IO.File.WriteAllLines("NewFile.txt", query)
```

6.
```
Dim query = From line In IO.File.ReadAllLines("Justices.txt")
                Let data = line.Split(","c)
                Let lastName = data(1)
                Let pres = data(2)
                Let state = data(3)
                Let presLastName = pres.Split(" "c).Last
                Let newLine = presLastName & "," & lastName & "," & state
                Select newLine
    IO.File.WriteAllLines("NewFile.txt", query)
```

In Exercises 7 through 10, describe the new file created by the code.

7.
```
Dim query = From line In IO.File.ReadAllLines("Justices.txt")
                Let data = line.Split(","c)
                Let firstName = data(0)
                Let lastName = data(1)
                Let yrAppointed = CInt(data(4))
                Let fullName = firstName & " " & lastName
                Let newLine = fullName & "," & yrAppointed
                Where lastName.StartsWith("B")
                Order By yrAppointed
                Select newLine
    IO.File.WriteAllLines("NewFile.txt", query)
```

8.
```
Dim query = From line In IO.File.ReadAllLines("Justices.txt")
                Let data = line.Split(","c)
                Let firstName = data(0)
                Let lastName = data(1)
                Let state = data(3)
                Let yrAppointed = CInt(data(4))
                Let fullName = firstName & " " & lastName
                Let newLine = state & "," & fullName
                Where (yrAppointed >= 1990) And (yrAppointed < 2000)
                Order By state
                Select newLine
    IO.File.WriteAllLines("NewFile.txt", query)
```

9.
```
Dim query = From line In IO.File.ReadAllLines("Justices.txt")
                Let data = line.Split(","c)
                Let firstName = data(0)
                Let lastName = data(1)
                Let pres = data(2)
                Let newLine = firstName & "," & lastName & "," & pres
                Select newLine
    IO.File.WriteAllLines("NewFile.txt", query)
```

10.
```
Dim query = From line In IO.File.ReadAllLines("Justices.txt")
                Let data = line.Split(","c)
                Let firstName = data(0)
                Let lastName = data(1)
                Let yrAppointed = CInt(data(4))
```

```
                Let yrLeft = CInt(data(5))
                Let fullName = firstName & " " & lastName
                Let newLine = fullName & "," & yrAppointed
                Where yrLeft = 0
                Order By yrAppointed
                Select newLine
    IO.File.WriteAllLines("NewFile.txt", query)
```

In Exercises 11 through 14, describe the new file created by the code. Assume the file NYTimes.txt contains the names of the subscribers to the *New York Times* and the file WSJ.txt contains the names of the subscribers to the *Wall Street Journal*.

11.
```
Dim times() As String = IO.File.ReadAllLines("NYTimes.txt")
Dim wsj() As String = IO.File.ReadAllLines("WSJ.txt")
IO.File.WriteAllLines("NewFile.txt", times.Union(wsj))
```

12.
```
Dim times() As String = IO.File.ReadAllLines("NYTimes.txt")
Dim wsj() As String = IO.File.ReadAllLines("WSJ.txt")
IO.File.WriteAllLines("NewFile.txt", times.Intersect(wsj))
```

13.
```
Dim times() As String = IO.File.ReadAllLines("NYTimes.txt")
Dim wsj() As String = IO.File.ReadAllLines("WSJ.txt")
IO.File.WriteAllLines("NewFile.txt", times.Except(wsj))
```

14.
```
Dim times() As String = IO.File.ReadAllLines("NYTimes.txt")
Dim wsj() As String = IO.File.ReadAllLines("WSJ.txt")
Dim unionArray() As String = times.Union(wsj).ToArray
Dim intersectArray() As String = times.Intersect(wsj).ToArray
IO.File.WriteAllLines("NewFile.txt", unionArray.Except(intersectArray))
```

In Exercises 15 through 18, use the file USPres.txt that contains the names of all the presidents of the United States and the file VPres.txt that contains the names of all the vice-presidents.

15. Write a program that creates a file containing the names of every president who also served as vice-president. The program also should display the number of those presidents in a message box.

16. Write a program that creates a file containing the names of every person who served as either vice-president or president. The program also should display the number of those names in a message box.

17. Write a program that creates a file containing the names of every person who served as either vice-president or president, but not both. The program also should display the number of those names in a message box.

18. Write a program that creates a file containing the names of every president who did not also serve as vice-president. The program also should display the number of those presidents in a message box.

In Exercises 19 through 26, write a program for the stated example or exercise from Section 7.3 without using a structure.

19. Section 7.3, Example 2

20. Section 7.3, Example 4

21. Section 7.3, Exercise 18

22. Section 7.3, Exercise 19

23. Section 7.3, Exercise 20

24. Section 7.3, Exercise 21

25. Section 7.3, Exercise 22

26. Section 7.3, Exercise 24

27. At the beginning of 1990, a complete box of Crayola[1] crayons had 72 colors (in the file Pre1990.txt). During the 1990s, 8 colors were retired (in the file Retired.txt) and 56 new colors were added (in the file Added.txt). Write a program that creates a text file listing the post-1990s set of 120 colors in alphabetical order.

Exercises 28 through 34 should use the file Justices.txt discussed at the beginning of this exercise set.

28. Write a program to create a file in which each record consists of two fields—the full name of a Supreme Court justice, and the justice's state.

29. Write a program to create a file similar to Justices.txt, but with the state field deleted.

30. Write a program that displays the entire contents of the file Justices.txt in a DataGridView control.

31. The file USStates.txt contains information about each of the 50 states. Each record contains four fields—name, abbreviation, land area (in square miles), population in the year 2000. The records are ordered by the states' date of entry into the union. The first four lines of the file are

```
Delaware,DE,1954,759000
Pennsylvania,PA,44817,12296000
New Jersey,NJ,7417,8135000
Georgia,GA,57906,7637000
```

Write a program that creates a file consisting of the states that have not produced any Supreme Court justices.

32. The first Supreme Court justice was appointed in 1789. Write a program to create a file that lists the years from 1789 through 2009 in which no Supreme Court justices were appointed. The first four lines of the file will be `1792`, `1794`, `1795`, and `1797`.

33. Write a program to create a file that lists the states that have produced Supreme Court justices, along with the number of justices produced. The states should be in alphabetical order by their abbreviations. The first four lines of the file will be `AL,3`; `AZ,2`; `CA,5`; `CO,1`.

34. Write a program to create a file that lists the presidents who have appointed Supreme Court justices, along with the number of justices appointed. The presidents should be in alphabetical order by their full names. The first two lines of the file will be `Abraham Lincoln,5`; `Andrew Jackson,5`.

35. Consider the file USStates.txt described in Exercise 31. Write a program to display the entire contents of the file in a DataGridView control with the states in alphabetical order.

Solutions to Practice Problems 8.1

1. The line would have to be changed to

```
txtName.Text = query.First.name
```

2. The ages of the presidents are all two-digit numbers and therefore will be ordered correctly when sorted as strings. In general, however, with arbitary ages, the query would not produce a correct result. For instance, a nine-year-old would be considered to be older than an eighty-year-old, and a centenarian would be considered to be younger than a nine-year-old.

3. The answer to the first question is "Yes." The same is true for Union and Intersect. However, the two arrays are different when Except is used. For instance, if the two arrays were reversed in the last event procedure of Example 5, then the contents of the file Except.txt would be the single word Delta.

[1] Crayola is a registered trademark of Binney & Smith.