34. The file Words.txt contains a list of words. Write a program that displays the words in a list box sorted by the number of different vowels (A, E, I, O and U) in the word. When two words have the same number of different vowels, they should be ordered first by their length (descending) and then alphabetically. The display should show both the word and the number of different vowels in the word. See Fig. 7.11.

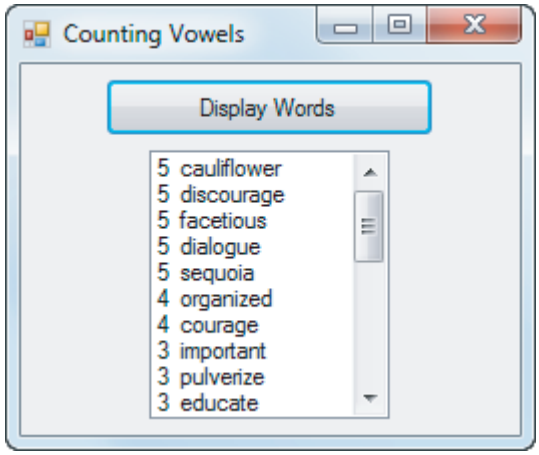


FIGURE 7.11 Output of Exercise 34.

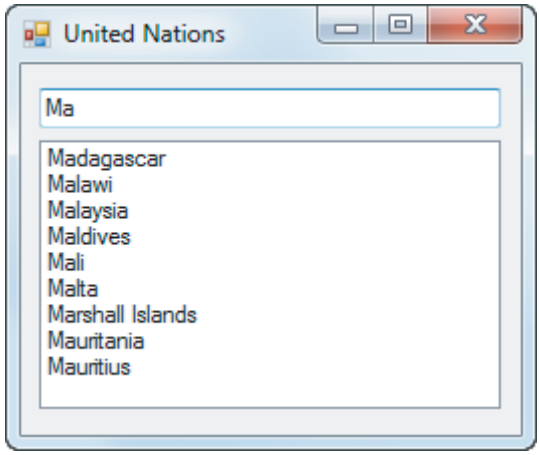


FIGURE 7.12 Output of Exercise 35.

VideoNote
Presidents (Homework)

35. The file Nations.txt contains the names of the 192 member nations of the United Nations. Write a program that initially displays all the nations in a list box. Each time a letter is typed into a text box, the program should reduce the displayed nations to those beginning with the letters in the text box. Figure 7.12 shows the status after the letters "Ma" are typed into the text box. At any time, the user should be able to click on the name of a nation to have it appear in the text box.

36. The **median** of an ordered set of measurements is a number separating the lower half from the upper half. If the number of measurements is odd, the median is the middle measurement. If the number of measurements is even, the median is the average of the two middle measurements. Write a program that requests a number n and a set of n measurements as input and then displays the median of the measurements.

Solutions to Practice Problems 7.2

1.
```
Private Sub btnDisplay_Click(...) Handles btnDisplay.Click
    Dim numbers() As String = IO.File.ReadAllLines("Numbers.txt")
    Dim query = From num In numbers
                Select CDbl(num)
    MessageBox.Show(CStr(query.Sum), "Total")
End Sub
```

2. (a) A text box can be filled only with a string. The value returned by a query is a *sequence* type that contains one string element. Only that element, not the sequence itself can be assigned to the text property of the text box.

   (b) `query(0)`, `query.Last`, `query.Max`, `query.Min`

7.3 Arrays of Structures

Often we work with several pieces of related data. For instance, four related pieces of data about a country are *name*, *continent*, *population*, and *area*. Suppose we are considering this information for the 192 countries in the United Nations. In the early days of programming, the way to work

with such information was to put it into four parallel arrays—one array of type String for names, a second array of type String for continents, a third array of type Double for populations, and a fourth array of type Double for areas. The modern way of dealing with such information is to place it into a single array of a composite data type that you define called a **structure** or a **user-defined data type**.

Structures

A structure contains variables of (possibly) different types, which are known as **members**. A structure is defined in the Declarations section of the Code Editor by a block of the form

```
Structure StructureName
  Dim memberName1 As MemberType1
  Dim memberName2 As MemberType2
  .
  .
End Structure
```

where *StructureName* is the name of the user-defined data type, *memberName1* and *memberName2* are the names of the members of the user-defined type, and *MemberType1* and *MemberType2* are the corresponding member data types.

Some examples of structures are

```
Structure Nation
  Dim name As String
  Dim continent As String
  Dim population As Double   'in millions
  Dim area As Double         'in square miles
End Structure

Structure Employee
  Dim name As String
  Dim dateHired As Date
  Dim hourlyWage As Double
End Structure

Structure College
  Dim name As String
  Dim state As String          'state abbreviation
  Dim yearFounded As Integer
End Structure
```

Variables having a user-defined data type are declared with Dim statements just like ordinary variables. For instance, the statement

```
Dim country As Nation
```

declares a variable of data type Nation.

Dot notation is used to refer to an individual member of a user-defined variable. For instance, the set of statements

```
country.name = "China"
country.continent = "Asia"
country.population = 1332.5
country.area = 3696100
```

assigns values to the members of the variable *country* declared above. After these assignment statements are executed, the statement

```
txtOutput.Text = country.continent
```

will display the string "Asia" in a text box, and the statement

```
txtOutput.Text = CStr(1000000 * country.population / country.area)
```

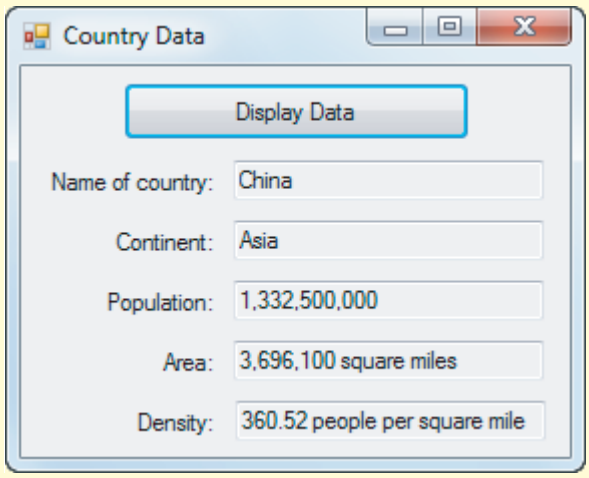
will display the population density of China in a text box.

Although structures are always defined in the Declarations section of the Code Editor, variables of user-defined type can be declared anywhere in a program. Just like ordinary variables, they have class-level, local, or block-level scope depending on where they are declared.

Example 1 The following program uses the Split method to assign values to the members of a variable having a user-defined type. This technique will play a vital role when values are assigned from text files to an array of structures. ***Note:*** Since the population member of the structure Nation is given in terms of millions, the value has to be multiplied by one million when used in a calculation.

```
Structure Nation
  Dim name As String
  Dim continent As String
  Dim population As Double  'in millions
  Dim area As Double        'in square miles
End Structure

Dim country As Nation       'class-level variable

Private Sub frmCountry_Load(...) Handles MyBase.Load
  'Assign values to country's member variables
  Dim line As String = "China,Asia,1332.5,3696100"
  Dim data() As String = line.Split(",c)
  country.name = data(0)
  country.continent = data(1)
  country.population = CDbl(data(2))
  country.area = CDbl(data(3))
End Sub

Private Sub btnDisplay_Click(...) Handles btnDisplay.Click
  'Display data in text boxes
  txtName.Text = country.name
  txtContinent.Text = country.continent
  txtPop.Text = FormatNumber(1000000 * country.population, 0)
  txtArea.Text = FormatNumber(country.area, 0) & " square miles"
  txtDensity.Text = FormatNumber(1000000 * country.population / country.area) &
                    " people per square mile"
End Sub
```

[Run, and click on the button.]



Arrays of Structures

Since a structure is a data type, an array can be declared with a structure as its data type. For instance, the statement

```
Dim nations(191) As Nation
```

declares *nations* to be an array of 192 elements, where each element has data type Nation. For each index *i*, *nations*(*i*) will be a variable of type Nation, and the values of its members will be *nations*(*i*).*name*, *nations*(*i*).*continent*, *nations*(*i*).*population*, and *nations*(*i*).*area*. Filling this 192-element array requires 4 × 192 = 768 pieces of data. This amount of data is best supplied by a text file. The optimum design for this text file is to have 192 lines of text, each consisting of 4 pieces of data delimited by commas. The next example uses the file UN.txt that gives data about the 192 members of the United Nations with the countries listed in alphabetical order. Some lines of the file are

```
Canada,North America,32.9,3855000
France,Europe,63.5,211209
New Zealand,Australia/Oceania,4.18,103738
Nigeria,Africa,146.5,356669
Pakistan,Asia,164,310403
Peru,South America,27.9,496226
```

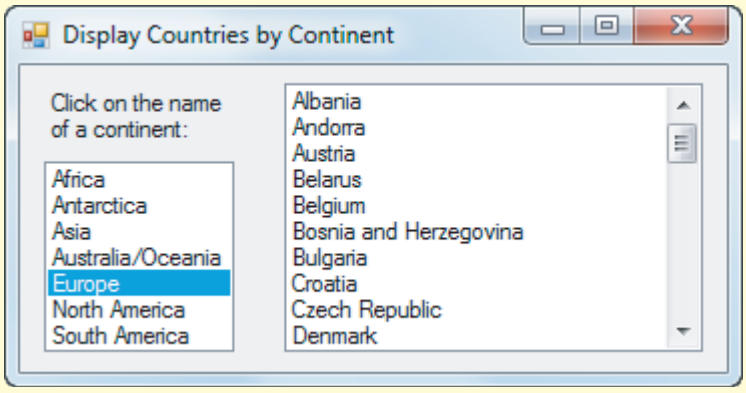
Each line of this text file is called a **record** and each record is said to contain four **fields**—a name field, a continent field, a population field, and an area field. The text file is said to use a **CSV format**. (CSV stands for "Comma Separated Values.")

Example 2 The following program uses the text file UN.txt to fill an array of structures and then uses the array to display the names of the countries in the continent selected by the user. The program uses two list boxes. Assume the String Collection Editor for lstContinents has been filled at design time with the names of the seven continents. The countries in the selected continent are displayed in lstCountries.

```
Structure Nation
  Dim name As String
  Dim continent As String
```

```
  Dim population As Double  'in millions
  Dim area As Double        'in square miles
End Structure

Dim nations(191) As Nation

Private Sub frmCountry_Load(...) Handles MyBase.Load
  'Place the contents of UN.txt into the array nations.
  Dim line As String
  Dim data() As String
  Dim countries() As String = IO.File.ReadAllLines("UN.txt")
  For i As Integer = 0 To 191
    line = countries(i)
    data = line.Split(",c)
    nations(i).name = data(0)
    nations(i).continent = data(1)
    nations(i).population = CDbl(data(2))
    nations(i).area = CDbl(data(3))
  Next
End Sub

Private Sub lstContinents_SelectedIndexChanged(...) Handles _
                              lstContinents.SelectedIndexChanged
  Dim selectedContinent As String = lstContinents.Text
  lstCountries.Items.Clear()
  If selectedContinent = "Antarctica" Then
    MessageBox.Show("There are no countries in Antarctica.")
  Else
    For i As Integer = 0 To 191
      If nations(i).continent = selectedContinent Then
        lstCountries.Items.Add(nations(i).name)
      End If
    Next
  End If
End Sub
```

[Run, and click on the name of a continent.]

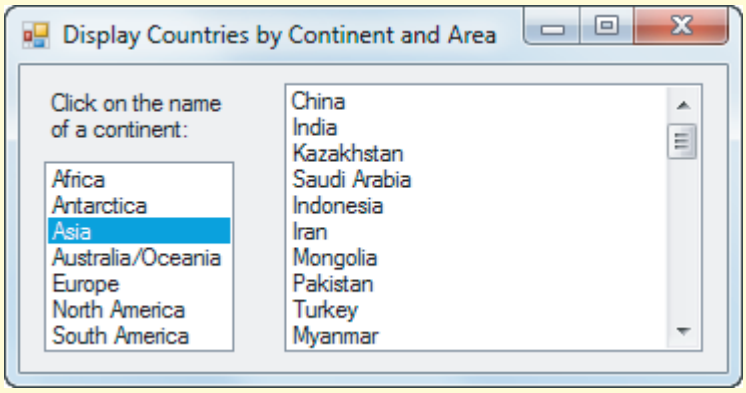


Queries can be used with arrays of structures in much the same way they are used with ordinary arrays.

Example 3 In the following variation of Example 2, the countries are displayed in descending order by their areas. LINQ is used both to filter the countries and to sort them by area. The query returns a sequence of names of countries.

```
Structure Nation
  Dim name As String
  Dim continent As String
  Dim population As Double  'in millions
  Dim area As Double        'in square miles
End Structure

Dim nations(191) As Nation

Private Sub frmCountry_Load(...) Handles MyBase.Load
  Dim line As String
  Dim data() As String
  Dim countries() As String = IO.File.ReadAllLines("UN.txt")
  For i As Integer = 0 To 191
    line = countries(i)
    data = line.Split(",c)
    nations(i).name = data(0)
    nations(i).continent = data(1)
    nations(i).population = CDbl(data(2))
    nations(i).area = CDbl(data(3))
  Next
End Sub

Private Sub lstContinents_SelectedIndexChanged(...) Handles _
                                lstContinents.SelectedIndexChanged
  Dim selectedContinent As String = lstContinents.Text
  Dim query = From country In nations
              Where country.continent = selectedContinent
              Order By country.area Descending
              Select country.name
  lstCountries.Items.Clear()
  If selectedContinent = "Antarctica" Then
    MessageBox.Show("There are no countries in Antarctica.")
  Else
    For Each countryName In query
      lstCountries.Items.Add(countryName)
    Next
  End If
End Sub
```

[Run, and click on the name of a continent.]



So far, LINQ Select clauses have contained a single item. However, Select clauses can contain multiple items. In that case the query returns a sequence of structures.

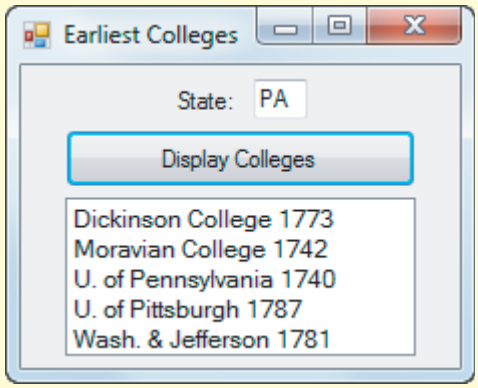
The next example uses the file Colleges.txt that contains data (name, state, and year founded) about colleges founded before 1800. The first four lines of the file are

```
Harvard U.,MA,1636
William and Mary,VA,1693
Yale U.,CT,1701
U. of Pennsylvania,PA,1740
```

Example 4 The following program displays colleges alphabetically ordered (along with their year founded) that are in the state specified in a masked text box. In this program we do not assume that the number of colleges in the text file is known in advance. Also, the Select clause returns a sequence of values whose data type is a structure having two members—a name member and a yearFounded member.

```
Structure College
  Dim name As String
  Dim state As String          'state abbreviation
  Dim yearFounded As Integer
End Structure

Dim colleges() As College

Private Sub frmColleges_Load(...) Handles MyBase.Load
  'Place the data for each college into the array schools.
  Dim schools() = IO.File.ReadAllLines("Colleges.txt")
  Dim n As Integer = schools.Count - 1
  ReDim colleges(n)
  Dim line As String   'holds data for a single college
  Dim data() As String
  For i As Integer = 0 To n
    line = schools(i)
    data = line.Split(","c)
    colleges(i).name = data(0)
    colleges(i).state = data(1)
    colleges(i).yearFounded = CInt(data(2))
  Next
End Sub

Private Sub btnDisplay_Click(...) Handles btnDisplay.Click
  Dim query = From col In colleges
              Where col.state = mtbState.Text.ToUpper
              Order By col.name Ascending
              Select col.name, col.yearFounded
  lstColleges.Items.Clear()
  For Each institution In query
    lstColleges.Items.Add(institution.name & " " & institution.yearFounded)
  Next
End Sub
```

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



In the program above, the query returned a sequence of values whose data type is a structure having two members. (The number of members was determined by the number of items in the Select clause.) The new structure type has no declared name, and thus it is said to have an **anonymous type**. Local type inference (provided by having Option Infer set to On) spares us from having to know the names of anonymous data types.

The DataGridView Control

In Section 7.2, the DataSource property was used to display (as a list) the values returned by a query having a single item in its Select clause. The DataSource property also can be used to display (as a table) the structure values returned by a query having two or more expressions in its Select clause. Instead of a list box, the values are displayed in 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*.)

If the Select clause of a query contains two or more items, then a pair of statements of the form

```
dgvOutput.DataSource = queryName.ToList
dgvOutput.CurrentCell = Nothing
```

displays the values returned by the query in the DataGridView control dgvOutput. (***Note:*** The second statement is optional. It prevents having a shaded cell in the table.) For instance, consider the program in the previous example. If the list box is replaced by the DataGridView control dgvColleges, the statement `lstColleges.Items.Clear()` is deleted, and the For Each loop is replaced by the statements

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

then the outcome will be as shown in Fig. 7.13 on the next page.

The blank column at the left side of the DataGridView control can be removed by setting the RowHeadersVisible property of the DataGridView control to False at design time. By default, the column headers contain the member names from the query's Select clause. The column header can be customized with the HeaderText property. Figure 7.14 results when the

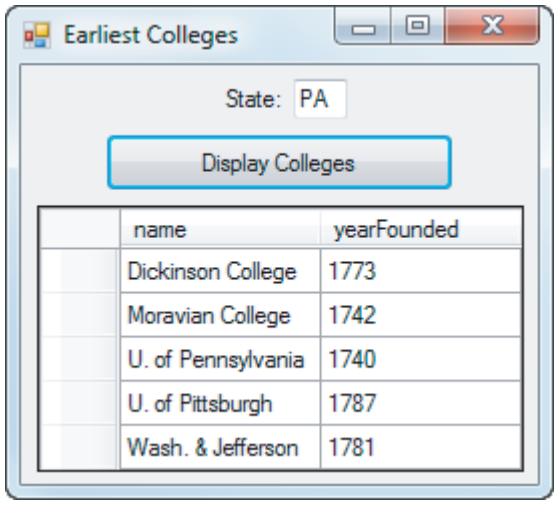


FIGURE 7.13 Use of a DataGridView in Example 4.

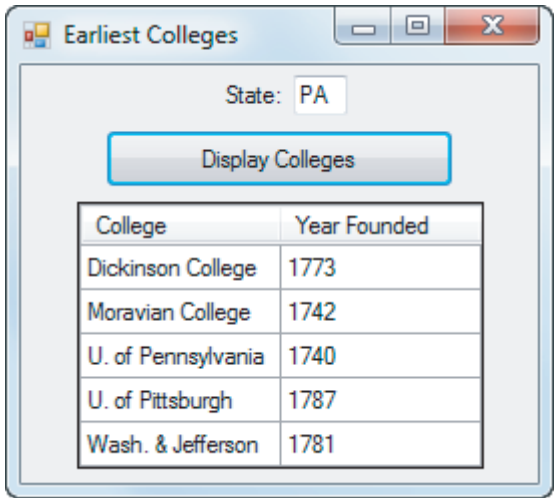


FIGURE 7.14 Customizing the headers in Example 4.

RowHeadersVisible property is set to False and the following two lines of code are added to the btnDisplay_Click event procedure:

```
dgvColleges.Columns("name").HeaderText = "College"
dgvColleges.Columns("yearFounded").HeaderText = "Year Founded"
```

Note: The DataGridView controls appearing in this textbook have been carefully sized to exactly fit the data. Comment 3 explains how this was accomplished. There is no need for you to strive for such precision when working the exercises.

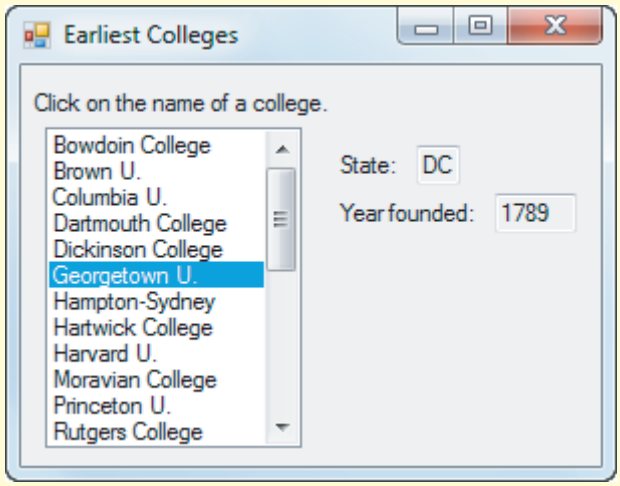
Searching an Array of Structures

Often one member (or pair of members) serves to uniquely identify each element in an array of structures. Such a member (or pair of members) is called a **key**. In Example 4, the *name* member is a key for the array of College structures. Often an array of structures is searched with a LINQ query for the sole element having a specific key value. If so, the query returns a sequence consisting of a single item, and a method such as the First method is used to display the value in a text box.

Example 5 The following program provides information about a college selected from a list box by the user. (***Note:*** In this program, the method First can be replaced by other methods, such as Last, Max, or Min.)

```
Structure College
  Dim name As String
  Dim state As String
  Dim yearFounded As Integer
End Structure

Dim colleges() As College

Private Sub frmColleges_Load(...) Handles MyBase.Load
  Dim schools() = IO.File.ReadAllLines("Colleges.txt")
  Dim n = schools.Count - 1
  ReDim colleges(n)
  Dim line As String    'holds data for a single college
  Dim data() As String
  For i As Integer = 0 To n
    line = schools(i)
    data = line.Split(",")c)
    colleges(i).name = data(0)
    colleges(i).state = data(1)
    colleges(i).yearFounded = CInt(data(2))
  Next
  Dim query = From institution In colleges
              Order By institution.name
              Select institution
  For Each institution In query
    lstColleges.Items.Add(institution.name)
  Next
End Sub

Private Sub lstColleges_SelectedIndexChanged(...) Handles _
              lstColleges.SelectedIndexChanged
  Dim query = From institution In colleges
              Where institution.name = lstColleges.Text
              Select institution
  txtState.Text = query.First.state
  txtYear.Text = CStr(query.First.yearFounded)
End Sub
```

[Run, and click on a college.]



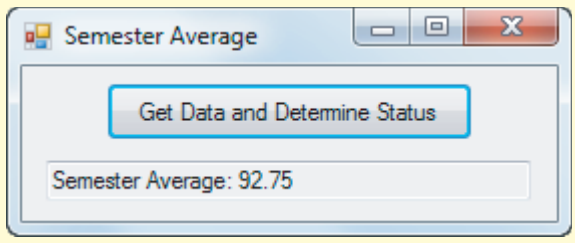
Using General Procedures with Structures

Variables whose type is a structure can be passed to Sub procedures and returned by Function procedures in the same way as variables of other data types.

Example 6 The following program uses general procedures to input grades and to curve the grades.

```
Structure Grades
  Dim exam1 As Double
  Dim exam2 As Double
  Dim final As Double
End Structure

Private Sub btnCalculate_Click(...) Handles btnCalculate.Click
  Dim scores As Grades
  Dim semesterAverage As Double
  GetGrades(scores)
  scores = CurveGrades(scores)
  semesterAverage = (scores.exam1 + scores.exam2 + 2 * scores.final) / 4
  txtOutput.Text = "Semester Average: " & FormatNumber(semesterAverage, 2)
End Sub

Sub GetGrades(ByRef scores As Grades)
  scores.exam1 = 80
  scores.exam2 = 90
  scores.final = 95
End Sub

Function CurveGrades(ByVal scores As Grades) As Grades
  scores.exam1 += 3
  scores.exam2 += 4
  scores.final += 2
  Return scores
End Function
```

[Run, and click on the button.]



Displaying and Comparing Structure Values

Statements of the form

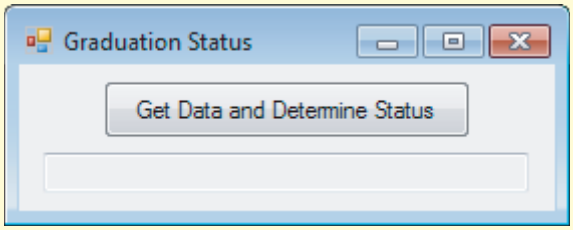
```
lstBox.Items.Add(structureVar)
```

where *structureVar* is a structure variable, do not perform as intended. Each member of a structure should appear separately in a lstBox.Items.Add statement. Also, comparisons involving structures using the relational operators <, >, =, <>, <=, and >= are valid only with individual members of the structures, not with the structures themselves.

Complex Structures (Optional)

So far, the members of structures have had elementary types; such as String or Integer. However, the type for a member can be another structure or an array. When a member is given an array type, the defining Dim statement must not specify the upper bound; this task must be left to a ReDim statement. Example 7 demonstrates the use of both of these nonelementary types of members.

Example 7 The following program totals a person's college credits and determines whether that person has enough credits for graduation. ***Notes:*** The structure variable *person* is local to the btnGet_Click event procedure. In the fifth line of the procedure, **`person.name.firstName`** should be thought of as **`(person.name).firstName`**.

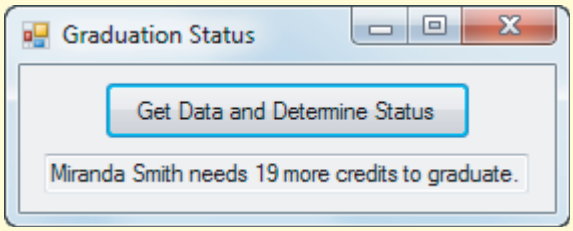


OBJECT	PROPERTY	SETTING
frmStatus	Text	Graduation Status
btnGet	Text	Get Data and Determine Status
txtResult	ReadOnly	True

```
Structure FullName
  Dim firstName As String
  Dim lastName As String
End Structure

Structure Student
  Dim name As FullName
  Dim credits() As Integer
End Structure

Private Sub btnGet_Click(...) Handles btnGet.Click
  Dim numYears As Integer
  Dim person As Student
  txtResult.Clear()
  person.name.firstName = InputBox("First Name:")
  person.name.lastName = InputBox("Last Name:")
  numYears = CInt(InputBox("Number of years completed:"))
  ReDim person.credits(numYears - 1)
  For i As Integer = 0 To numYears - 1
    person.credits(i) = CInt(InputBox("Credits in year " & i + 1))
  Next
  DetermineStatus(person)
End Sub

Sub DetermineStatus(ByVal person As Student)
  Dim query = From num In person.credits
              Select num
  Dim total As Integer = query.Sum
  If (total >= 120) Then
    txtResult.Text = person.name.firstName & " " &
             person.name.lastName & " has enough credits to graduate."
```

```
  Else
    txtResult.Text = person.name.firstName & " " &
                     person.name.lastName & " needs " &
                     (120 - total) & " more credits to graduate."
  End If
End Sub
```

[Run, click on the button, and respond to requests for input with *Miranda*, *Smith*, *3*, *34*, *33*, *34*.]



Comments

1. When a Select clause contains two or more items, none of the items can involve computed values. For instance, consider the query in Example 3 and suppose we were interested in the age of each college in 2010. The following query would not be valid:

```
Dim collegeQuery = From col In colleges
                   Where col.state = mtbState.Text.ToUpper
                   Order By col.name Ascending
                   Select col.name, 2010 - col.yearFounded
```

   Instead, the query must be written

```
Dim collegeQuery = From col In colleges
                   Where col.state = mtbState.Text.ToUpper
                   Order By col.name Ascending
                   Let age = 2010 - col.yearFounded
                   Select col.name, age
```

2. If a column header cell in a DataGridView control is too narrow to accommodate its text, the text is displayed in two or more lines. However, if an ordinary cell is too narrow to accommodate its text, part of the text will be cut off. If you suspect that some cells will be too narrow, set the AutoSizeColumn property to *AllCells*. Then, each column's width will automatically adjust to accommodate the longest entries in the column.
3. When the AutoSizeColumn property of a DataGridView control is left at its default setting of *None*, the control's size is specified by its Size property. The setting for the Size property has the form w, h, where w is the width of the grid in pixels and h is its height in pixels. After the RowHeadersVisible property has been set to False, good values for the Size property are $w = 100 \cdot$ [number of columns] $+ 3$, and $h = 22 \cdot$ [number of rows] $+ 1$. ***Note:*** If there will be a scroll bar on the right side of the grid, add 17 to w. If any column header will occupy more than one line, add 13 to h for each additional line.

Practice Problems 7.3

1. Find the errors in the following event procedure.

```
Sub btnDisplay_Click(...) Handles btnDisplay.Click
  Structure Team
```

```
    Dim school As String
    Dim mascot As String
  End Structure
  Team.school = "Rice"
  Team.mascot = "Owls"
  txtOutput.Text = Team.school & " " & Team.mascot
End Sub
```

2. Correct the code in Practice Problem 1.

EXERCISES 7.3

In Exercises 1 through 10, determine the output displayed when the button is clicked.

1.
```
Structure Rectangle
  Dim length As Integer
  Dim width As Integer
End Structure

Private Sub btnDisplay_Click(...) Handles btnDisplay.Click
  Dim footballField As Rectangle
  footballField.length = 120   'yards
  footballField.width = 160    'yards
  Dim area As Integer = footballField.length * footballField.width
  txtOutput.Text = "The area of a football field is " & area &
                   " square yards."
End Sub
```

2.
```
Structure College
  Dim name As String
  Dim state As String
  Dim yearFounded As Integer
End Structure

Private Sub btnDisplay_Click(...) Handles btnDisplay.Click
  Dim school As College
  school.name = "USC"
  school.state = "CA"
  school.yearFounded = 1880
  'Now.Year is the current year
  Dim age As Integer = Now.Year - school.yearFounded
  txtOutput.Text = school.name & " is " & age & " years old."
End Sub
```

3.
```
Structure College
  Dim name As String
  Dim state As String
  Dim yearFounded As Integer
End Structure

Private Sub btnDisplay_Click(...) Handles btnDisplay.Click
  Dim school As College
```

```
    Dim line As String = "Duke,NC,1838"
    Dim data() As String = line.Split(","c)
    school.name = data(0)
    school.state = data(1)
    school.yearFounded = CInt(data(2))
    txtOutput.Text = school.name & " was founded in " & school.state &
                     " in " & school.yearFounded & "."
End Sub
```

4.
```
Structure College
    Dim name As String
    Dim state As String
    Dim yearFounded As Integer
End Structure

Private Sub btnDisplay_Click(...) Handles btnDisplay.Click
    Dim school As College
    Dim data() As String = {"Stanford", "CA", "1885"}
    school.name = data(0)
    school.state = data(1)
    school.yearFounded = CInt(data(2))
    txtOutput.Text = school.name & " was founded in " & school.state &
                     " in " & school.yearFounded & "."
End Sub
```

5.
```
Structure Appearance
    Dim height As Double
    Dim weight As Double
End Structure

Private Sub btnDisplay_Click(...) Handles btnDisplay.Click
    Dim person1, person2 As Appearance
    person1.height = 72
    person1.weight = 170
    person2.height = 12 * 6
    If person1.height = person2.height Then
      lstOutput.Items.Add("heights are same")
    End If
    person2 = person1
    lstOutput.Items.Add(person2.weight)
End Sub
```

6.
```
Structure Employee
    Dim name As String
    Dim hoursWorked As Double
    Dim hourlyWage As Double
    Dim eligibleForBonus As Boolean
End Structure

Dim worker As Employee

Private Sub frmWages_Load(...) Handles Me.Load
    worker.name = "John Q. Public"
```

```
  worker.hoursWorked = 40
  worker.hourlyWage = 25
  worker.eligibleForBonus = True
End Sub

Private Sub btnDetermine_Click(...) Handles btnDetermine.Click
  Dim wage As Double
  wage = worker.hoursWorked * worker.hourlyWage
  If worker.eligibleForBonus Then
    wage = wage + 0.1 * wage
  End If
  MessageBox.Show("Wage for " & worker.name & ": " & FormatCurrency(wage))
End Sub
```

7.
```
Structure TestData
  Dim name As String
  Dim score As Double
End Structure

Dim students() As String = IO.File.ReadAllLines("Scores.txt")

Private Sub btnDisplay_Click() Handles btnDisplay.Click
  Dim student As TestData
  For i As Integer = 0 To students.Count - 1
    student = GetScore(i)
    DisplayScore(student)
  Next
End Sub

Function GetScore(ByVal i As Integer) As TestData
  Dim student As TestData
  Dim line As String = students(i)
  Dim data() As String = line.Split(","c)
  student.name = data(0)
  student.score = CDbl(data(1))
  Return student
End Function

Sub DisplayScore(ByVal student As TestData)
  lstOutput.Items.Add(student.name & ": " & student.score)
End Sub
```

(Assume that the three lines of the file Scores.txt contain the following data: Joe,88; Moe,90; Roe,95.)

8.
```
Structure Employee
  Dim name As String
  Dim dateHired As Date
  Dim hasDependents As Boolean
End Structure

Private Sub btnDisplay_Click(...) Handles btnDisplay.Click
  Dim worker As Employee
```

```
  worker.name = "John Jones"
  worker.dateHired = #9/20/2010#
  worker.hasDependents = True
  If DateDiff(DateInterval.Day, worker.dateHired, Today) < 180 Then
    MessageBox.Show("Not eligible to participate in the health plan.")
  Else
    MessageBox.Show("The monthly cost of your health plan is " &
                    HealthPlanCost(worker.hasDependents) & ".")
  End If
End Sub

Function HealthPlanCost(ByVal hasDependents As Boolean) As String
  If hasDependents Then
    Return FormatCurrency(75)
  Else
    Return FormatCurrency(50)
  End If
End Function
```

(Assume that today is 1/1/2011.)

9.
```
Structure Address
  Dim street As String
  Dim city As String
  Dim state As String
End Structure

Structure Citizen
  Dim name As String
  Dim dayOfBirth As Date
  Dim residence As Address
End Structure

Private Sub btnDisplay_Click(...) Handles btnDisplay.Click
  Dim person As Citizen
  person.name = "Mr. President"
  person.dayOfBirth = #8/4/1961#
  person.residence.street = "1600 Pennsylvania Avenue"
  person.residence.city = "Washington"
  person.residence.state = "DC"
  txtOutput.Text = person.name & " lives in " &
         person.residence.city & ", " & person.residence.state
End Sub
```

10.
```
Structure TaxData
  Dim socSecNum As String
  Dim numWithAllow As Integer  'number of withholding allowances
  Dim maritalStatus As String
  Dim hourlyWage As Double
End Structure

Structure Employee
  Dim name As String
```

```
  Dim hrsWorked As Double
  Dim taxInfo As TaxData
End Structure

Private Sub btnDisplay_Click(...) Handles btnDisplay.Click
  Dim worker As Employee
  worker.name = "Hannah Jones"
  worker.hrsWorked = 40
  worker.taxInfo.hourlyWage = 20
  txtOutput.Text = worker.name & " earned " &
      FormatCurrency(worker.hrsWorked * worker.taxInfo.hourlyWage)
End Sub
```

In Exercises 11 through 13, determine the errors.

11.

```
Structure Nobel
  Dim peace As String
  Dim yr As Integer
End Structure

Private Sub btnDisplay_Click(...) Handles btnDisplay.Click
  Dim prize As Nobel
  peace = "Martti Ahtisaari"
  yr = 2008
  txtOutput.Text = peace & " won the " & yr & " Nobel Peace Prize."
End Sub
```

12.

```
Structure Vitamins
  Dim a As Double
  Dim c As Double
End Structure

Private Sub btnDisplay_Click(...) Handles btnDisplay.Click
  Dim minimum As Vitamins
  minimum.c = 60
  minimum.a = 5000
  lstOutput.Items.Add(minimum)
End Sub
```

13.

```
Structure BallGame
  Dim hits As Double
  Dim runs As Double
End Structure

Private Sub btnDisplay_Click(...) Handles btnDisplay.Click
  Dim game1, game2 As BallGame
  game1.hits = 15
  game1.runs = 8
  game2.hits = 17
  game2.runs = 10
  If game1 > game2 Then
    txtOutput.Text = "The first game was better."
  Else
```

```
    txtOutput.Text = "The second game was at least as good."
  End If
End Sub
```

14. Write lines of code as instructed in Steps (a) through (e) to fill in the missing lines in the following program.

```
Structure Appearance
  Dim height As Double   'inches
  Dim weight As Double   'pounds
End Structure

Structure Person
  Dim name As String
  Dim stats As Appearance
End Structure

Private Sub btnDisplay_Click(...) Handles btnDisplay.Click
  Dim person1, person2 As Person
  (missing lines)
End Sub
```

(a) Give *person1* the name Michael.
(b) Set Michael's height and weight to 71 and 190, respectively.
(c) Give *person2* the name Jacob.
(d) Set Jacob's height and weight to 70 and 175, respectively.
(e) If one person is both taller and heavier than the other, display a sentence of the form "[name of bigger person] is bigger than [name of smaller person]."

In Exercises 15 through 18 describe the output that results from clicking on the button. The programs use the file Cities.txt that contains information about the 25 largest cities in the United States. Each record of the file has four fields—*name*, *state*, *population in 2000* (in 100,000s), and *population in 2010* (in 100,000s). The first four lines in the file are as follows:

```
New York,NY,80.1,82.7
Los Angeles,CA,36.9,38.84
Chicago,IL,29.0,28.7
Houston,TX,19.5,22.4
```

Assume that each program contains the following code:

```
Structure City
  Dim name As String
  Dim state As String
  Dim pop2000 As Double
  Dim pop2010 As Double
End Structure

Dim cities() As City

Private Sub frmCities_Load(...) Handles Me.Load
  'Place the data for each city into the array cities.
  Dim cityRecords() = IO.File.ReadAllLines("Cities.txt")
  'Use the array cityRecords to populate the array cities.
  Dim n = cityRecords.Count - 1
```

```
    ReDim cities(n)
    Dim line As String   'holds data for a single city
    Dim data() As String
    For i As Integer = 0 To n
      line = cityRecords(i)
      data = line.Split(",")c)
      cities(i).name = data(0)
      cities(i).state = data(1)
      cities(i).pop2000 = CDbl(data(2))
      cities(i).pop2010 = CDbl(data(3))
    Next
  End Sub
```

15.
```
  Private Sub btnDisplay_Click(...) Handles btnDisplay.Click
    Dim query = From cty In cities
                Where cty.state = "TX"
                Order By cty.pop2010 Descending
                Select cty.name, cty.pop2010
    For Each cty In query
      lstOutput.Items.Add(cty.name & "  " &
                          FormatNumber(100000 * cty.pop2010, 0))
    Next
  End Sub
```

16.
```
  Private Sub btnDisplay_Click(...) Handles btnDisplay.Click
    Dim query = From cty In cities
                Where cty.state = "TX"
                Let growth = (cty.pop2010 - cty.pop2000) / cty.pop2000
                Order By growth Descending
                Select cty.name, cty.state, growth
    For Each cty In query
      lstOutput.Items.Add(cty.name & ",  " & cty.state)
    Next
    lstOutput.Items.Add("Greatest growth: " &
                        FormatPercent(query.First.growth))
  End Sub
```

17.
```
  Private Sub btnDisplay_Click(...) Handles btnDisplay.Click
    Dim query = From cty In cities
                Let increase = cty.pop2010 - cty.pop2000
                Where cty.name = "Phoenix"
                Select increase
    txtOutput.Text = FormatNumber(100000 * query.First, 0)
  End Sub
```

18.
```
  Private Sub btnDisplay_Click(...) Handles btnDisplay.Click
    Dim query = From cty In cities
                Order By cty.pop2010 Descending
                Select cty.pop2010
    Dim pops() As Double = query.ToArray
    ReDim Preserve pops(9)
    txtOutput.Text = FormatNumber(100000 * pops.Sum, 0)
  End Sub
```

In Exercises 19 through 22 use the file USStates.txt that consists of 50 records and four fields. Each field gives a piece of information about a state—*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
```

19. Write a program that accepts a state's abbreviation as input and displays the state's name and its area. See Fig. 7.15.

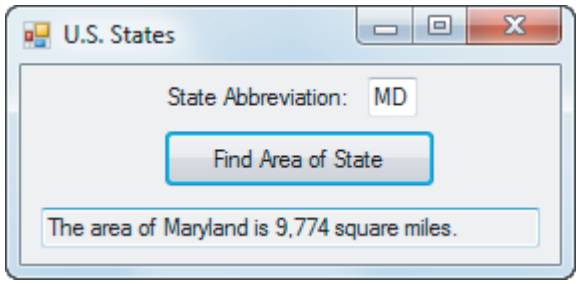


FIGURE 7.15 Possible outcome of Exercise 19.

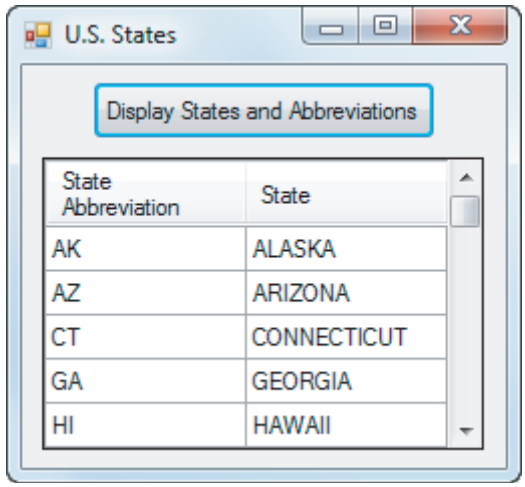


FIGURE 7.16 Outcome of Exercise 20.

20. Write a program that displays the names of the states whose abbreviations are different than the first two letters of their name. Both the abbreviations and the states should be displayed. See Fig. 7.16.

21. Write a program that displays the names of the states sorted by their population densities in descending order. Next to each name should be the state's population density. See Fig. 7.17.

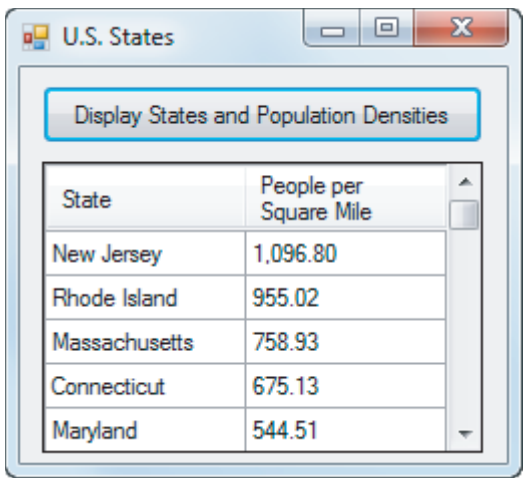


FIGURE 7.17 Outcome of Exercise 21.

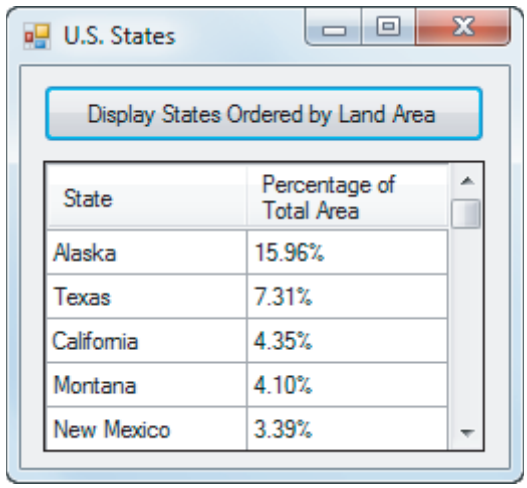


FIGURE 7.18 Outcome of Exercise 22.

22. Write a program that displays the names of the states ordered by land area. Next to each name should be the percentage of the total U.S. land area in that state. See Fig. 7.18.

In Exercises 23 through 26 use the file Baseball.txt that contains data about the performance of major league baseball players during the 2009 regular season. Each record of the file contains four fields—*name*, *team*, *atBats*, and *hits*. Some lines of the file are as follows:

```
Aaron Hill,Blue Jays,682,195
Ichiro Suzuki,Mariners,639,225
Derek Jeter,Yankees,634,212
```

23. Write a program using the file Baseball.txt that requests a team as input from a list and displays the players from that team. The players should be sorted in decreasing order by the number of hits they had during the season. The output should display each player's full name and number of hits. See Fig. 7.19.

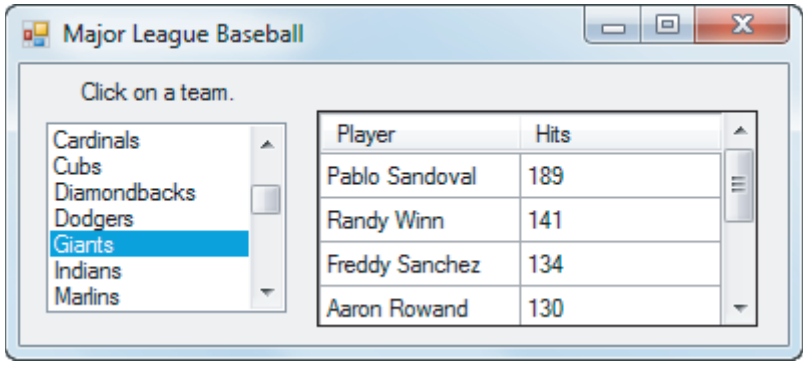


FIGURE 7.19 Outcome of Exercise 23.

24. Write a program that requests a team as input and displays the players from that team. The players should be sorted alphabetically by their last names. Players having the same last name should be ordered secondarily by their first names. (***Note:*** The Split method can be used to extract first and last names from a person's full name. For instance, the value of `"Babe Ruth".Split(" ")(1)` is `Ruth`.) The output should display each player's full name and batting average. See Fig. 7.20.

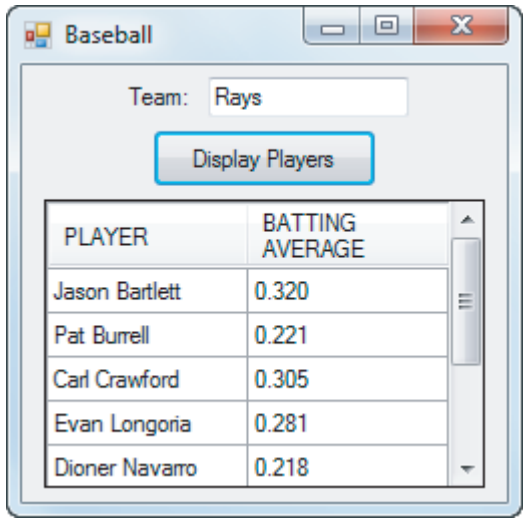


FIGURE 7.20 Outcome of Exercise 24.

25. Write a program that displays the highest batting average and the player (or players) having the highest batting average. The output also should display each player's team. See Fig. 7.21.

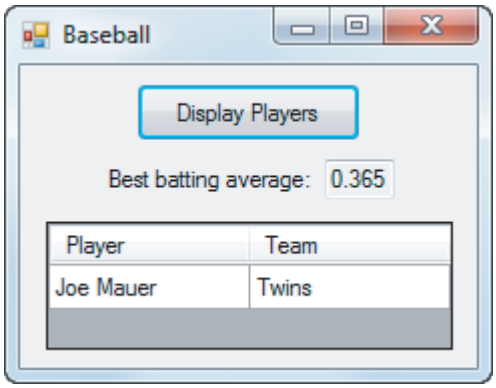


FIGURE 7.21 Outcome of Exercise 25.

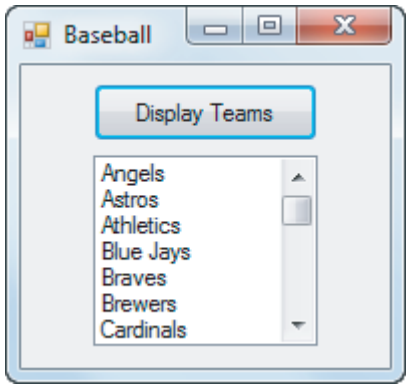


FIGURE 7.22 Outcome of Exercise 26.

26. Write a program using the file Baseball.txt that displays the names of the teams sorted alphabetically. See Fig. 7.22.

In Exercises 27 through 30 use the file Justices.txt that contains data about the Supreme Court justices, past and present. Each record of the file contains six fields—*first name, last name, appointing president, the state from which they were appointed, year appointed,* and *the year they left the court.* (For sitting judges, the last field is set to 0.) The first five lines of the file are as follows:

```
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
Harry,Blackman,Richard Nixon,MN,1970,1994
```

27. Write a program that displays the sitting justices ordered by the year they joined the Supreme Court.

28. Write a program that requests the name of a president as input from a list and then displays the justices appointed by that president. The justices should be ordered by the length of time they served on the court in descending order. (***Note:*** For sitting justices, use `Now.Year - yrAppointed` as their time of service. Otherwise, use `yrLeft - yrAppointed.`) Use the file USPres.txt to fill the presidents list box. That file contains the names of the presidents in the order they served. See Fig. 7.23.

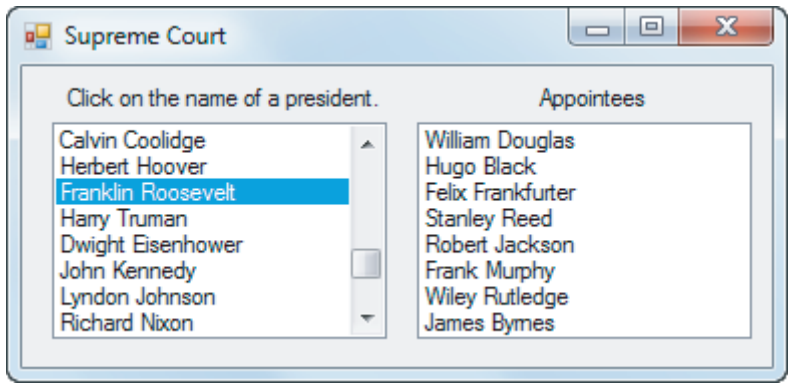


FIGURE 7.23 Possible outcome of Exercise 28.

29. Write a program that requests a state abbreviation as input and displays the justices appointed from that state. The justices should be ordered by their year appointed. The output should also display the last name of the appointing president and the length of time served. (***Note:*** For sitting justices, use `Now.Year - yrAppointed` as their time of service. Otherwise, use `yrLeft - yrAppointed`.) Also, the program should inform the user if no justices have been appointed from the requested state. See Fig. 7.24.

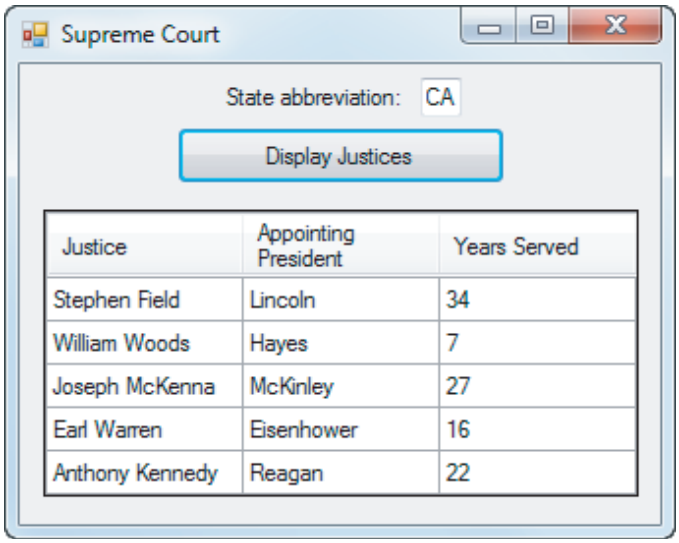


FIGURE 7.24 Possible outcome of Exercise 29.

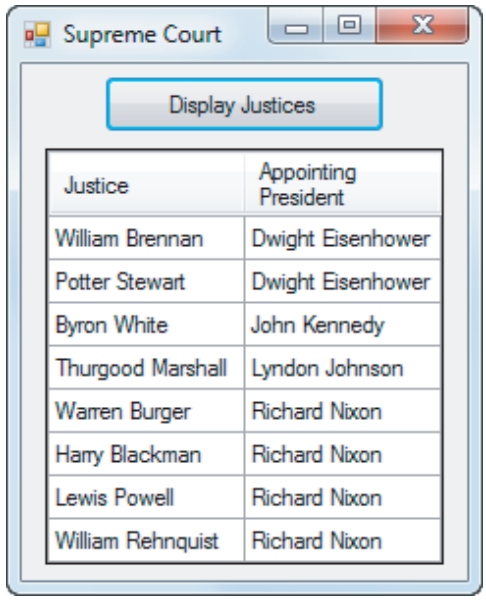


FIGURE 7.25 Outcome of Exercise 30.

30. Write a program that displays the makeup of the Supreme Court at the beginning of 1980. The justices should be ordered by the year they were appointed, and the names of the appointing presidents should be displayed. See Fig. 7.25.

31. *The Twelve Days of Christmas*. Each year, PNC Advisors of Pittsburgh publishes a Christmas price index. See Table 7.3. Write a program that requests an integer from 1 through 12 and then lists the gifts for that day along with that day's cost. On the *n*th day, the *n* gifts are 1 partridge in a pear tree, 2 turtle doves, . . . , *n* of the *n*th gift. The program also should give the total cost up to and including that day. As an example, Fig. 7.26 shows the output in the list box when the user enters 3. The contents of Table 7.3, along with the day corresponding to each gift, are contained in the file Gifts.txt. The first three lines of the file are as follows:

```
1,partridge in a pear tree,159.99
2,turtle doves,27.99
3,French hens,15
```

TABLE 7.3 Christmas price index for 2009.

Item	Cost	Item	Cost
partridge in a pear tree	159.99	swan-a-swimming	750.00
turtle dove	27.99	maid-a-milking	7.25
French hen	15.00	lady dancing	608.11
calling bird	149.99	lord-a-leaping	441.36
gold ring	99.99	piper piping	207.70
goose-a-laying	25.00	drummer drumming	206.26

```
The gifts for day 3 are
1 partridge in a pear tree
2 turtle doves
3 French hens

Cost for day 3: $260.97
Total cost for the first 3 days: $636.93
```

FIGURE 7.26 Sample output for Exercise 31.

The file Famous.txt contains the names of some famous Americans and their birthdays. Use this file in Exercises 32 through 35. The first three lines of the file are

```
Paul Allen,1/21/1953
Lance Armstrong,9/18/1971
Neil Armstrong,8/5/1930
```

32. *Tuesday's child is full of grace.* Display a list of the people in the file Famous.txt born on a Tuesday. (***Hint:*** Use the FormatDateTime function.) See Fig. 7.27.

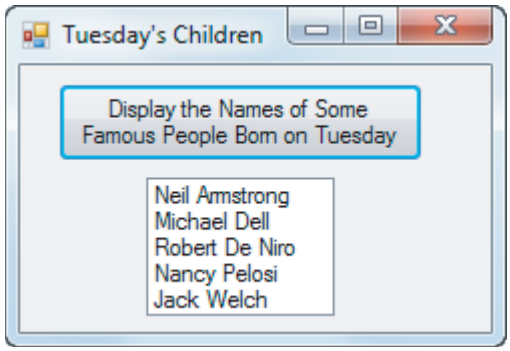


FIGURE 7.27 Output of Exercise 32.

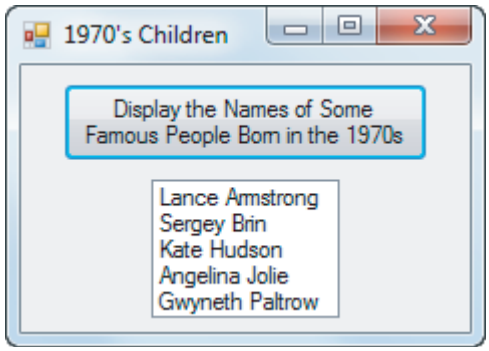


FIGURE 7.28 Output of Exercise 33.

33. Display a list of the people in the file Famous.txt born during the 1970s. See Fig. 7.28.

34. Display a table showing all the people in the file Famous.txt along with their ages. The people should be ordered by their ages in descending order.

35. Display a table showing all the people in the file Famous.txt who are in their forties along with their ages in days and the days of the week they were born.

A campus club has 10 members. The following program stores information about the students into an array of structures. Each structure contains the student's name and a list of the courses he or she is currently taking. Exercises 36 through 39 request that an additional event procedure be written for this program.

```
Structure Student
  Dim name As String
  Dim courses() As String
End Structure

Dim club(9) As Student   'Holds all students in the club

Private Sub frmStudents_Load(...) Handles MyBase.Load
  Dim pupil As Student
  pupil.name = "Juan Santana"
  ReDim pupil.courses(2)
  pupil.courses(0) = "CMSC 100"
```

```
  pupil.courses(1) = "PHIL 200"
  pupil.courses(2) = "ENGL 120"
  club(0) = pupil
  'Enter data for second student
  pupil.name = "Mary Carlson"
  ReDim pupil.courses(3)
  pupil.courses(0) = "BIOL 110"
  pupil.courses(1) = "PHIL 200"
  pupil.courses(2) = "CMSC 100"
  pupil.courses(3) = "MATH 220"
  club(1) = pupil
  pupil.name = "George Hu"
  ReDim pupil.courses(2)
  pupil.courses(0) = "MATH 220"
  pupil.courses(1) = "PSYC 100"
  pupil.courses(2) = "ENGL 200"
  club(2) = pupil
  'Enter names and courses for remaining 7 people in the club
End Sub
```

36. Write the code for a btnDisplay_Click event procedure that displays the names of all the students in the club in a list box.
37. Write the code for a btnDisplay_Click event procedure that displays the names of all the students in the club who are registered for three courses.
38. Write the code for a btnDisplay_Click event procedure that displays the names of all the students in the club who are enrolled in CMSC 100.
39. Write the code for a btnDisplay_Click event procedure that displays the names of all the students in the club who are *not* enrolled in CMSC 100.

Solutions to Practice Problems 7.3

1. The event procedure contains two errors. First, the definition of a structure cannot be inside a procedure; it must be typed into the Declarations section of the Code Editor. Second, the statements `Team.school = "Rice"` and `Team.mascot = "Owls"` are not valid. "Team" should be replaced by a variable of type Team that has previously been declared.
2.
```
Structure Team
  Dim school As String
  Dim mascot As String
End Structure

Private Sub btnDisplay_Click(...) Handles btnDisplay.Click
  Dim squad As Team
  squad.school = "Rice"
  squad.mascot = "Owls"
  txtOutput.Text = squad.school & " " & squad.mascot
End Sub
```

7.4 Two-Dimensional Arrays

Each array discussed so far held a single list of items. Such array variables are called **one-dimensional** or **single-subscripted variables**. An array can also hold the contents of a table with several rows and columns. Such array variables are called **two-dimensional** or **double-subscripted variables**. Two tables follow. Table 7.4 on the next page gives the road mileage between certain cities. It has four rows and four columns. Table 7.5 shows the leading universities in three graduate-school programs. It has three rows and five columns.