35. You are offered two salary options for ten days of work. Option 1: $100 per day. Option 2: $1 the first day, $2 the second day, $4 the third day, and so on, with the amount doubling each day. Determine which option pays better.

Exercises 36 and 37 should use the following Function procedure.

```
Function DayOfWeek(ByVal d As Date) As String
  Dim str As String = FormatDateTime(d, DateFormat.LongDate)
  Dim n As Integer = str.IndexOf(",")
  Return str.Substring(0, n)
End Function
```

36. Request a year as input and then display the date of the first Tuesday of that year.

37. Request a year as input and then display the dates of the first Tuesdays of each month of that year.

Solutions to Practice Problem 6.2

1. The loop will never be entered because 15 is greater than 1. The intended first line might have been

   ```
   For i As Integer = 15 To 1 Step -1
   ```

   or

   ```
   For i As Integer = 1 To 15
   ```

2. If the exact number of times the loop will be executed is known before entering the loop, then a For . . . Next loop should be used. Otherwise, a Do loop is more appropriate.

6.3 List Boxes and Loops

In previous sections we used list boxes to display output and to facilitate selection. In this section we explore some additional features of list boxes and use loops to analyze data in list boxes.

Some Properties, Methods, and Events of List Boxes

VideoNote
List boxes and loops

During run time, the value of

```
lstBox.Items.Count
```

is the number of items currently in the list box. Each item in lstBox is identified by an **index number** ranging from 0 through lstBox.Items.Count − 1. For instance, if the list box contains 10 items, then the first item has index 0, the second item has index 1, . . . , and the last item has index 9. In general, the *n*th item in a list box has index $n - 1$.

During run time, the user can highlight an item in a list box by clicking on the item with the mouse or by moving to it with the up- and down-arrow keys when the list box has the focus. The SelectedIndexChanged event occurs each time an item of a list box is clicked on or each time an arrow key is used to change the highlighted item. It is the default event for list box controls.

The value of

```
lstBox.SelectedIndex
```

is the index number of the item currently highlighted in lstBox. If no item is highlighted, the value of SelectedIndex is −1. The statement

```
lstBox.SelectedIndex = -1
```

will unhighlight any highlighted item in the list. ***Note:*** This statement also raises the SelectedIndexChanged event.

The value of

```
lstBox.Items(n)
```

is the item of lstBox having index *n*. The elements of the list are of a data type called Object. A value of any type may be added to the list. However, type casting must take place whenever an element of the list is assigned to a numeric or string variable or is concatenated with another variable or literal. For instance, the statement

```
txtBox.Text = CStr(lstBox.Items(0))
```

displays the first item of lstBox in a text box.

The value of

```
lstBox.Text
```

is the currently highlighted item of lstBox converted to a string.

The Sorted property is perhaps the most interesting list box property. When it is set to True (at either design time or run time), items will automatically be displayed in alphabetical (i.e., ANSI) order. The default value of the Sorted property is False.

After the SelectedIndexChanged event, the two most important events for list boxes are the Click and DoubleClick events. However, if a program contains procedures for both of these events and the user double-clicks on the list box, only the Click event will be raised.

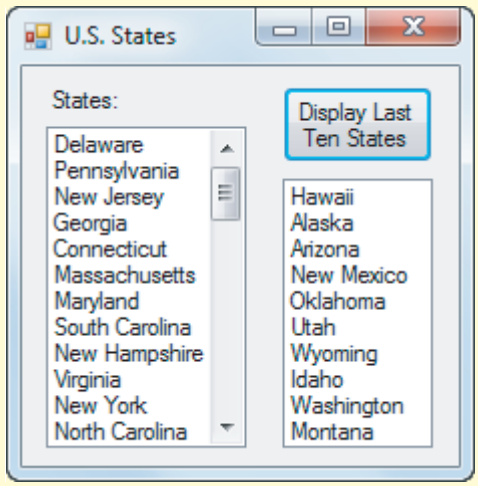
The items in a list box are usually all strings or all numbers. When the items are all strings, we use loops to search for items and to extract information. When the items are all numbers, we use loops to perform calculations.

List Boxes Populated with Strings

Example 1 The following program uses two list boxes, named lstStates and lstLastTen. We assume that the String Collection Editor of lstStates contains the names of the 50 U.S. states in the order they joined the union. The program displays the last 10 states to join the union beginning with the most recent. ***Note:*** If *n* is the number of items in lstStates, then the last item in lstStates has index *n* − 1.

```
Private Sub btnDisplay_Click(...) Handles btnDisplay.Click
  Dim n As Integer = lstStates.Items.Count
  For i As Integer = (n - 1) To (n - 10) Step -1
    lstLastTen.Items.Add(lstStates.Items(i))
  Next
End Sub
```

[Run, and click on the button.]



When a list is searched, we often use a Boolean variable called a **flag** to tell us whether or not the sought-after item has been found. The value of the flag is set to False initially and then is changed to True if and when the sought-after item is found.

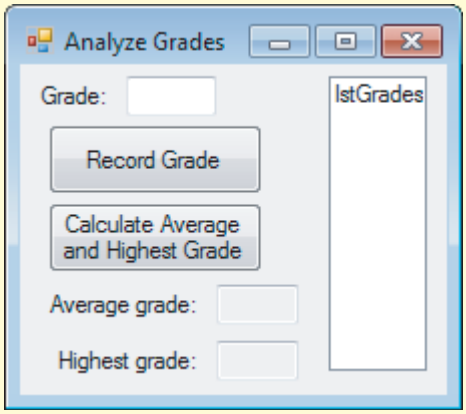
Example 2 The following program uses a list box named lstStates whose String Collection Editor contains the names of the 50 U.S. states in the order they joined the union. The program also uses a masked text box with Mask "LL". After the user enters two letters into the masked text box, the program uses a Do loop to search the list box for a state beginning with those letters. The Do loop terminates when the state is found or when the last item in the list box has been examined. If a state is found, the program reports its full name and the order in which it joined the union. If there is no state beginning with the pair of letters, the program so reports.

```
Private Sub btnSearch_Click(...) Handles btnSearch.Click
  Dim letters As String = mtbFirstTwoLetters.Text.ToUpper
  Dim foundFlag As Boolean = False   'indicates whether state has been found
  Dim i As Integer = -1   'index of the state currently considered
  Do Until (foundFlag) Or (i = lstStates.Items.Count - 1)
    i += 1
    If CStr(lstStates.Items(i)).ToUpper.StartsWith(letters) Then
      foundFlag = True
    End If
  Loop
  If foundFlag Then
    txtOutput.Text = CStr(lstStates.Items(i)) & " is state #" & (i + 1) & "."
  Else
    txtOutput.Text = "No state begins with " & mtbFirstTwo.Text & "."
  End If
End Sub
```

[Run, enter two letters into the masked text box, and click on the button.]

List Boxes Populated with Numbers

Example 3 The following program evaluates exam grades. The user inserts a grade into the list box by typing it into the txtGrade text box and then clicking on the *Record* button. After all the grades have been entered, the user clicks on the *Calculate* button to determine the average grade and the highest grade for the exam. The average grade is calculated as [sum of grades] / [number of grades]. The variable *sum* adds up the grades during a loop through the grades. The number of grades is just the number of items in the list box. The variable *maxGrade* starts out set to 0. It is then adjusted during each pass through the loop. ***Note:*** To prevent the program from crashing, the btnCalculate_Click event procedure checks that the list box contains some items.

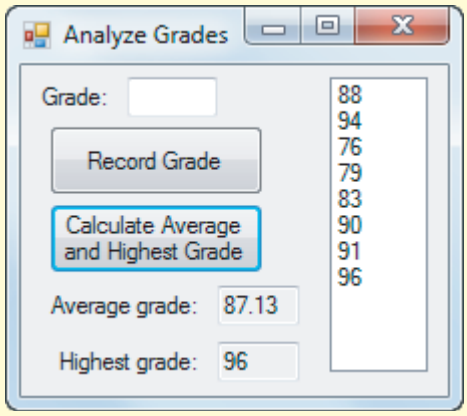


OBJECT	PROPERTY	SETTING
frmGrades	Text	Analyze Grades
lblGrade	Text	Grade:
txtGrade		
btnRecord	Text	Record Grade
btnCalculate	Text	Calculate Average and Highest Grade
lblAverage	Text	Average grade:
txtAverage	ReadOnly	True
lblHighest	Text	Highest grade:
txtHighest	ReadOnly	True
lstGrades		

```
Private Sub btnRecord_Click(...) Handles btnRecord.Click
  lstGrades.Items.Add(txtGrade.Text)
  txtGrade.Clear()
  txtGrade.Focus()
End Sub

Private Sub btnCalculate_Click(...) Handles btnCalculate.Click
  Dim sum As Double = 0
  Dim maxGrade As Double = 0
  If lstGrades.Items.Count > 0 Then 'condition is true when list box is nonempty
    For i As Integer = 0 To lstGrades.Items.Count - 1
      sum += CDbl(lstGrades.Items(i))
      If CDbl(lstGrades.Items(i)) > maxGrade Then
        maxGrade = CDbl(lstGrades.Items(i))
      End If
    Next
    txtAverage.Text = FormatNumber(sum / lstGrades.Items.Count, 2)
    txtHighest.Text = CStr(maxGrade)
  Else
    MessageBox.Show("You must first enter some grades.")
  End If
End Sub
```

[Run, enter some grades, and then click on the *Calculate* button.]



Note: In Example 3, the average grade and the highest grade could have been calculated without the grades being stored in a list box. Some calculations, however, such as the standard deviation, do require the grades to be stored.

Searching an Ordered List

When the items in a list of strings are in alphabetical order, the search can be shortened. For instance, if you are searching an ordered list of words for one that begins with the letter *D*, you can certainly stop the search when you reach words beginning with *E*. Consider Example 2. Whenever the pair of letters entered into the masked text box were not the first two letters of a state, the entire list was searched. Such searches can be shortened considerably if the states are first ordered.

Example 4 The following program has the same controls and settings as Example 2, except that the Sorted property of the list box is set to True. The program begins by looking at the items one at a time until it locates a state whose name exceeds the sought-after letters alphabetically. If that state doesn't begin with the sought-after letters, we can assume that no state in the list does.

```
Private Sub btnSearch_Click(...) Handles btnSearch.Click
  Dim letters As String = mtbFirstTwoLetters.Text.ToUpper
  Dim i As Integer = 0    'index of the state currently considered
  Do Until (CStr(lstStates.Items(i)).ToUpper > letters) Or
           (i = lstStates.Items.Count - 1)
    i += 1
  Loop
  If CStr(lstStates.Items(i)).ToUpper.StartsWith(letters) Then
    txtOutput.Text = CStr(lstStates.Items(i)) & " begins with " &
                     mtbFirstTwoLetters.Text & "."
  Else
    txtOutput.Text = "No state begins with " & mtbFirstTwoLetters.Text & "."
  End If
End Sub
```

Comments

1. A list box containing numbers might not be numerically in increasing order even when the Sorted property is set to True. For instance, since the ANSI table determines order, the number 88 will precede the number 9.
2. Example 4 presents one way to search a list of strings that are in alphabetical order. A more efficient technique, called a *binary search*, is discussed in Programming Project 8.

Practice Problems 6.3

1. Write a program that displays a message box telling you whether a SelectedIndexChanged event was caused by the pressing of an arrow key or was caused by clicking on an item.
2. Consider Example 3. Why couldn't the maximum grade be calculated with the following code?

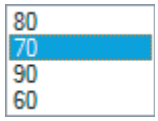
```
lstGrades.Sorted = True
maxGrade = CDbl(lstGrades.Items(lstGrades.Items.Count - 1))
```

EXERCISES 6.3

In Exercises 1 through 6, assume that lstBox is as shown below. Determine the contents of the text box after the code is executed.

1. `txtOutput.Text = lstBox.Text`
2. `txtOutput.Text = CStr(lstBox.Items(2))`
3. `txtOutput.Text = CStr(lstBox.Items(lstBox.Items.Count - 1))`
4. `txtOutput.Text = CStr(lstBox.Items(lstBox.SelectedIndex))`
5. `txtOutput.Text = CStr(lstBox.SelectedIndex)`
6.
```
Dim total As Integer = 0
For n As Integer = 0 To lstBox.Items.Count - 1
  If CStr(lstBox.Items(n)).Length = 6 Then
    total += 1
  End If
Next
txtOutput.Text = CStr(total)
```

In Exercises 7 through 12, assume that lstBox is as shown below. Determine the contents of the text box after the code is executed.



7. `txtOutput.Text = CStr(lstBox.Items(0))`

8. `txtOutput.Text = lstBox.Text`

9. `txtOutput.Text = CStr(lstBox.Items(lstBox.SelectedIndex))`

10. `txtOutput.Text = CStr(lstBox.SelectedIndex)`

11.
```
Dim num As Integer = 0
For n As Integer = 0 To lstBox.Items.Count - 1
  num += CInt(lstBox.Items(n))
Next
txtOutput.Text = CStr(num)
```

12.
```
Dim min As Double = 100
For n As Integer = 0 To lstBox.Items.Count - 1
  If CDbl(lstBox.Items(n)) < min Then
    min = CDbl(lstBox.Items(n))
  End If
Next
txtOutput.Text = CStr(min)
```

In Exercises 13 through 18, fill the String Collection Editor of lstBox at design time with the winners of the nearly 100 Rose Bowl games that have been played.[1] The first three items in the list box will be Michigan, Washington State, and Oregon. Some colleges appear many times in the list. Write a program that performs the indicated task.

13. Count the number of times USC has won the Rose Bowl.

14. After the user clicks on the name of a college in lstBox, count the number of times the college has won the Rose Bowl.

15. Determine if a college input by the user in a text box has won the Rose Bowl. Assume that the Sorted property of lstBox is set to False. The procedure should terminate the search if and when the college is found.

16. Determine if a college input by the user in a text box has won the Rose Bowl. Assume that the Sorted property of lstBox is set to True. The procedure should terminate the search as soon as possible.

17. Fill lstBox2 with the entries of lstBox, but in reverse order.

18. Fill lstBox2 with the colleges (in alphabetical order) that have won the Rose Bowl, with each winner appearing just once.

Suppose lstBox has been filled with the 50 U.S. states in the order they joined the union.[2] In Exercises 19 through 34, write a program to perform the indicated task.

19. Display in lstBox2 the states in alphabetical order.

20. Display in lstBox2 the states in reverse alphabetical order.

21. Display in lstBox2 the states whose names (including spaces) are seven letters long.

22. Determine the first state in lstBox whose name is seven letters long. The program should terminate the search as soon as the state is found.

---

[1]The file Rosebowl.txt (found in the folder Programs\Ch06\Text_Files_for_Exercises) contains the names of the Rose Bowl winners in the order the games were played. Copy the contents of the text file and Paste them into the String Collection Editor of lstBox.

[2]The file States.txt (found in the folder Programs\Ch06\Text_Files_for_Exercises) contains the names of the states in the order they joined the union.

23. Determine the first state in lstBox whose name begins with "New". The program should terminate the search as soon as the state is found.

24. Display in lstBox2 the states whose names begin with "New".

25. Determine the length of the longest state name, and display in lstBox2 the states having that length.

26. Determine the length of the shortest state name, and display in lstBox2 the states having that length.

27. Display in lstBox2 the states whose names have four letters that are vowels. The program should call a Function procedure NumberOfVowels that counts the number of letters in a string that are vowels.

28. After the user clicks on a state, determine the number of letters in the state's name that are vowels.

29. Determine the maximum number of letters that are vowels for the names of the 50 states. The program should call a Function procedure NumberOfVowels that counts the number of letters in a string that are vowels.

30. Determine the number of states whose names consist of two words.

31. Display the name of the first state to join the union.

32. Display the name of the last state to join the union.

33. Display the name of the fifth state to join the union.

34. Display in lstBox2 the names of the original thirteen states.

35. Alter Example 3 so that the btnCalculate_Click event procedure calculates the lowest grade instead of the highest grade.

36. The **standard deviation** measures the spread or dispersal of a set of numbers about the mean. Formally, if $x_1, x_2, x_3, \ldots, x_n$ is a collection of n numbers with average value m, then

$$\text{standard deviation} = \sqrt{\frac{(x_1 - m)^2 + (x_2 - m)^2 + (x_3 - m)^2 + \cdots + (x_n - m)^2}{n}}.$$

Extend Example 3 so that the btnCalculate_Click event procedure also calculates the standard deviation of the grades.

37. The **range** of a set of numbers is the difference between the highest and the lowest numbers. Modify Example 3 so that the btnCalculate_Click procedure calculates the range of the grades instead of the highest grade.

38. Alter Example 3 so that the btnCalculate_Click event procedure calculates the number of above-average grades instead of the maximum grade.

39. Rewrite Example 4 so that the btnSearch_Click event procedure starts at the last item and searches backwards.

Solutions to Practice Problems 6.3

1.
```
Dim clickFlag As Boolean

Private Sub lstBox_Click(...) Handles lstBox.Click
  clickFlag = True
End Sub

Private Sub lstBox_SelectedIndexChanged(...) Handles _
                        lstBox.SelectedIndexChanged
  Dim msg As String = "The SelectedIndexChanged event was caused by "
```

```
  If clickFlag Then
    MessageBox.Show(msg & "clicking on an item of the list box.")
  Else
    MessageBox.Show(msg & "pressing an arrow key.")
  End If
  clickFlag = False
End Sub
```

2. The ordering in the list box is determined by the ANSI table (where the items are treated as strings), not the numerical value. Therefore the last item in the list box might not have the greatest numerical value.

CHAPTER 6 SUMMARY

1. A *Do loop* repeatedly executes a block of statements either as long as or until a certain condition is true. The condition can be checked either at the top of the loop or at the bottom.
2. A *For . . . Next loop* repeats a block of statements a fixed number of times. The *counter variable* assumes an initial value and increases it by one after each pass through the loop until it reaches the terminating value. Alternative increment values can be specified with the *Step* keyword.
3. Visual Basic uses *local type inference* to infer the data types of local variables declared without an As clause by looking at the data type of the initialization expression.
4. The items in a list box are assigned *index numbers* ranging from 0 to [number of items minus 1]. Loops can use the index numbers to extract information from list boxes.
5. A *flag* is a Boolean variable used to indicate whether a certain event has occurred or a certain situation exists.

CHAPTER 6 PROGRAMMING PROJECTS

1. *Caffeine Absorption*. After caffeine is absorbed into the body, 13% is eliminated from the body each hour. Assume a person drinks an 8-oz cup of brewed coffee containing 130 mg of caffeine, and the caffeine is absorbed immediately into the body. Write a program to compute the following values. See Fig. 6.12.
   (a) The number of hours required until 65 mg (one-half the original amount) remain in the body.

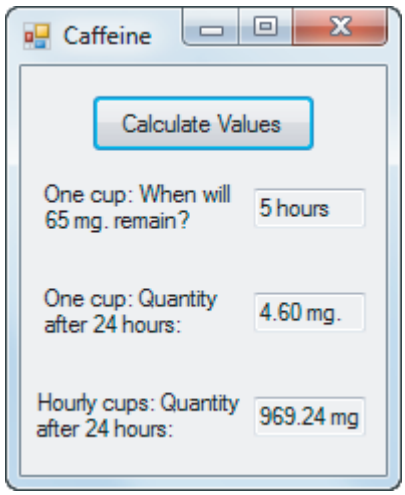


FIGURE 6.12 Output of Programming Project 1.

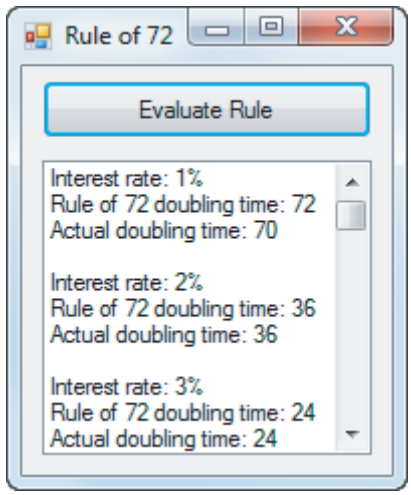


FIGURE 6.13 Output of Programming Project 2.