PREV NEXT

[Page 307]

Chapter 7. Arrays

(This item omitted from WebBook edition)

PREV PREV NEXT NEXT

[Page 308]

7.1. Creating and Accessing Arrays

A variable (or simple variable) is a name to which Visual Basic can assign a single value. An array variable is a collection of simple variables of the same type to which Visual Basic can efficiently assign a list of values.

Consider the following situation: Suppose that you want to evaluate the exam grades for 30 students. Not only do you want to compute the average score, but you also want to display the names of the students whose scores are above average. You might place the 30 pairs of student names and scores in a text file and run the program outlined:

```
Private Sub btnDisplay_Click(...) Handles btnDisplay.Click
  Dim student1 As String, score1 As Double
  Dim student2 As String, score2 As Double
  Dim student3 As String, score3 As Double
    .
    .
  Dim student30 As String, score30 As Double
  'Analyze exam grades
  Dim sr As IO.StreamReader = IO.File.OpenText("SCORES.TXT")
  student1 = sr.ReadLine
  score1 = CDbl(sr.ReadLine)
  student2 = sr.ReadLine
  score2 = CDbl(sr.ReadLine)
    .
    .
  student30 = sr.ReadLine
  score30 = CDbl(sr.ReadLine)
  sr.Close()
  'Compute the average grade
    .
    .
  'Display names of above average students
    .
```

```
    .
End Sub
```

This program is going to be uncomfortably long. What's most frustrating is that the 30 Dim statements and 30 pairs of statements reading from the file are very similar and look as if they should be condensed into a short loop. A shorthand notation for the many related variables would be welcome. It would be nice if we could just write

```
For i As Integer = 1 To 30
  studenti = sr.ReadLine
  scorei = CDbl(sr.ReadLine)
Next
```

Of course, this will not work. Visual Basic will treat studenti and scorei as two variables and keep reassigning new values to them. At the end of the loop, they will have the values of the thirtieth student.

[Page 309]

Declaring an Array Variable

Visual Basic provides a data structure called an array that lets us do what we tried to accomplish in the loop. The variable names, similar to those in the preceding program, will be

```
student(0), student(1), student(2), student(3), ..., student(29)
```

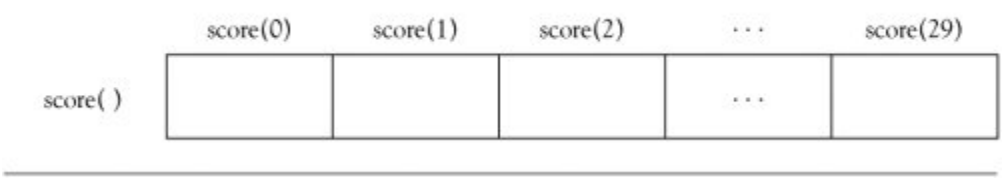
and

```
score(0), score(1), score(2), score(3), ..., score(29)
```

We refer to these collections of variables as the array variables student() and score(). The numbers inside the parentheses of the individual variables are called subscripts, and each individual variable is called a subscripted variable or element . For instance, student(3) is the fourth subscripted variable of the array student(), and score(20) is the 21st subscripted variable of the array score(). The elements of an array are assigned successive memory locations. Figure 7.1 shows the memory locations for the array score().

Figure 7.1. The array score().

[View full size image]



Array variables have the same kinds of names as simple variables. If arrayName is the name of an array variable and n is an Integer literal, variable, or expression, then the declaration statement

```
Dim arrayName(n) As varType
```

reserves space in memory to hold the values of the subscripted variables arrayName(0), arrayName(1), arrayName(2), ..., arrayName(n). The value of n is called the upper bound of the array. The number of elements in the array,n+1, is called the size of the array. The subscripted variables will all have the same data type; namely, the type specified by varType. For instance, they could be all String variables or all Integer variables. In particular, the statements

```
Dim score(29) As Double
Dim score(29) As Double
```

declare the arrays needed for the preceding program.

Values can be assigned to individual subscripted variables with assignment statements and displayed in text boxes and list boxes just as values of ordinary variables. The default initial value of each subscripted variable is the same as with an ordinary variable; that is, the keyword Nothing for String types and 0 for numeric types. The statement

```
Dim score(29) As Double
```

[Page 310]

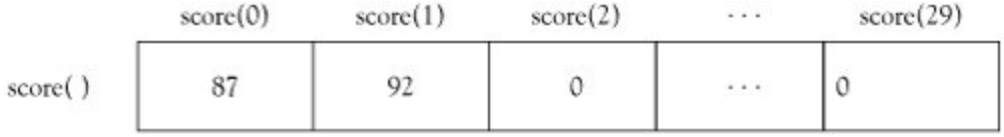
sets aside a portion of memory for the Integer array score() and assigns the default value 0 to each element.

[View full size image]

The statements

```
score(0) = 87
score(1) = 92
```

assign values to the zeroth and first elements.



The statements

```
For i As Integer = 0 To 2
  lstBox.Items.Add(score(i))
Next
```

then produce the following output in the list box:

```
87
92
0
```

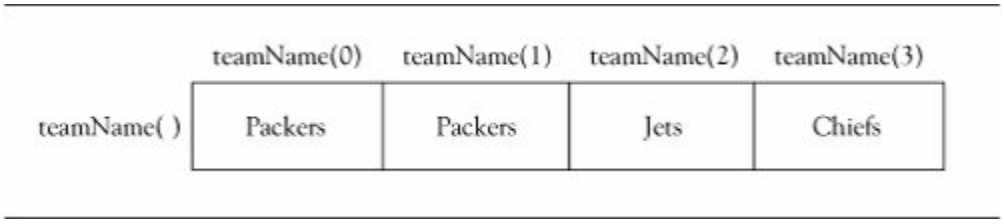
As with an ordinary variable, an array declared in the Declarations section of the Code window is class-level. That is, it will be visible to all procedures in the form, and any value assigned to it in a procedure will persist when the procedure terminates. Array variables declared inside a procedure are local to that procedure and cease to exist when the procedure is exited.

Example 1.

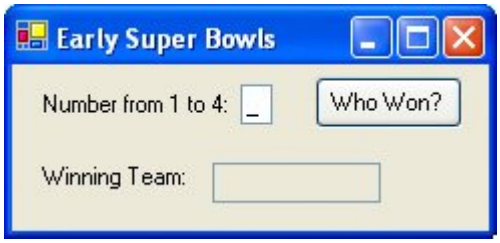
(This item is displayed on pages 310 - 311 in the print version)

The following program creates a string array consisting of the names of the first four Super Bowl winners. Figure 7.2 shows the array created by the program.

Figure 7.2. The array teamName() of Example 1.



[Page 311]

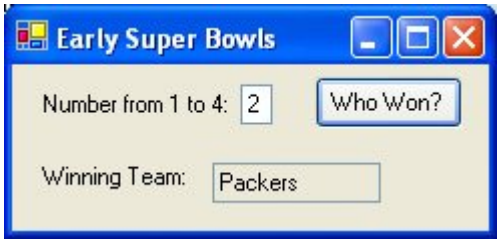


Object	Property	Setting
frmBowl	Text	Early Super Bowls
lblNumber	Text	Number from 1 to 4:
mtxtNumber	Mask	0

btnWhoWon	Text	Who Won?
lblWinner	Text	Winning Team:
txtWinner	ReadOnly	True

```
Private Sub btnWhoWon_Click(...) Handles btnWhoWon.Click
  Dim teamName(3) As String
  Dim n As Integer
  'Place Super Bowl Winners into the array
  teamName(0) = "Packers"
  teamName(1) = "Packers"
  teamName(2) = "Jets"
  teamName(3) = "Chiefs"
  'Access array
  n = CInt(mtxtNumber.Text)
  txtWinner.Text = teamName(n - 1)
End Sub
```

[Run, type 2 into the masked text box, and click the button.]



The Load Event Procedure

In Example 1, the array teamName was assigned values within the btnWhoWon_Click event procedure. Every time the button is clicked, the values are reassigned to the array. This manner of assigning values to an array can be very inefficient, especially in programs with large arrays where the task of the program (in Example 1, looking up a fact) may be repeated numerous times for different user input. When, as in Example 1, the data to be placed in an array are known at the time the program begins to run, a more efficient location for the statements that fill the array is in the form's Load event procedure. The Load event occurs automatically when the form is loaded into memory, before it becomes visible on the screen. The header and End Sub statements for the Load event procedure are placed in the Code window when you double-click on the form. A typical header looks like

```
Private Sub frmName_Load(...) Handles MyBase.Load
```

The keyword MyBase is similar to the Me keyword and refers to the form being loaded. Example 2 uses the frmBowl_Load procedure to improve on Example 1.

[Page 312]

Example 2.

The following improvement on Example 1 makes teamName () a class-level array and assigns values to the elements of the array in the frmBowl_Load procedure:

```
Dim teamName(3) AsString
Private Sub btnWhoWon_Click(...) Handles btnWhoWon.Click
  Dim n As Integer
  n = CInt(mtxtNumber.Text)
  txtWinner.Text = teamName(n - 1)
End Sub

Private Sub frmBowl_Load(...) HandlesMyBase.Load
  'Place Super Bowl Winners into the array
  teamName(0) = "Packers"
  teamName(1) = "Packers"
  teamName(2) = "Jets"
  teamName(3) = "Chiefs"
End Sub
```

Like ordinary variables, array variables can be declared and assigned initial values at the same time. A statement of the form

```
Dim arrayName() As varType = {value0, value1, value2, ..., valueN}
```

declares an array having upper bound N and assigns value0 to arrayName(0), value1 to arrayName(1), value2 to arrayName(2), ..., and valueN to arrayName(N). For instance, in Example 4, the Dim statement and frmBowl_Load event procedure can be replaced by the single line

```
Dim teamName() As String = {"Packers", "Packers", "Jets", "Chiefs"}
```

The GetUpperBound Method

The value of arrayName.GetUpperBound(0) is the upper bound of the array. For instance, in Example 1, the value of `teamName.GetUpperBound(0)` is 3.

ReDim Statement

After an array has been declared, its size (but not its type) can be changed with a statement of the form

```
ReDim arrayName(m)
```

where arrayName is the name of the already declared array and m is an Integer literal, variable, or expression. Note: Since the type cannot be changed, there is no need for an "As dataType" clause at the end of the ReDim statement.

Visual Basic allows you to declare an array without specifying an upper bound with a statement of the form

```
Dim arrayName() As varType
```

Later, the size of the array can be stipulated with a ReDim statement.

[Page 313]

The ReDim statement has one shortcoming: It causes the array to lose its current contents. That is, it resets all String values to Nothing and resets all numeric values to 0. This situation can be remedied by following ReDim with the word Preserve. The general form of a ReDim Preserve statement is

```
ReDim Preserve arrayName(m)
```

Of course, if you make an array smaller than it was, data in the eliminated elements will be lost.

Example 3.

(This item is displayed on pages 313 - 314 in the print version)

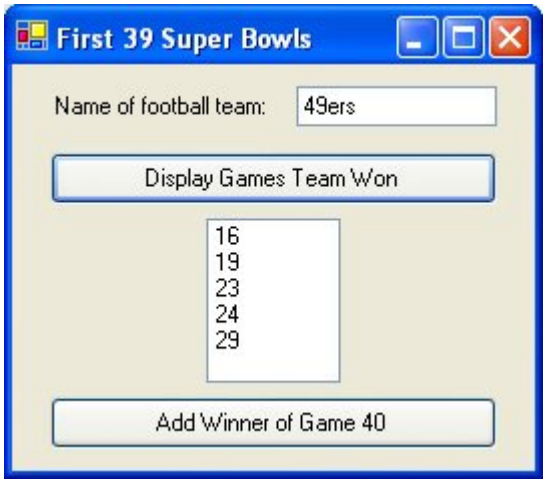
The following program reads the names of the winners of Super Bowl games from a file and places them into an array. The user can type a team's name into a text box and then display the numbers of the Super Bowl games won by that team. The user has the option of adding winners of subsequent games to the array of winners. The program uses the text file SBWINNERS.TXT whose lines contain the names of the winners in order. That is, the first four lines of the file contain the names Packers, Packers, Jets, and Chiefs.

```
Dim teamName() AsString
Dim upperBound As Integer

Private Sub frmBowl_Load(...) HandlesMyBase.Load
  Dim sr As IO.StreamReader = IO.File.OpenText("SBWINNERS.TXT")
  Dim name As String, numLines As Integer
  'Count number of lines in the file and assign it minus one to upperBound
  numLines = 0
  Do While (sr.Peek <> -1)
    name = sr.ReadLine
    numLines += 1
  Loop
  upperBound = numLines -1
  sr.Close()
  'Specify the title bar of the form
  Me.Text = "First "  & upperBound + 1  & " Super Bowls"
  'Specify the caption of the Next Winner button
  btnNextWinner.Text = "Add Winner of Game "  & upperBound + 2
  'Specify the size of the array and fill it with the entries of the file
  ReDim teamName(upperBound)
  sr = IO.File.OpenText("SBWINNERS.TXT")
  For i As Integer = 0 To upperBound
    teamName(i) = sr.ReadLine
  Next
  sr.Close()
End Sub
```

[Page 314]

```
Private Sub btnDisplay_Click(...) Handles btnDisplay.Click
  'Display the numbers of the games won by the team in the text box
  lstGamesWon.Items.Clear()
  For i As Integer = 0 To upperBound
    If teamName(i) = txtName.Text Then
      lstGamesWon.Items.Add(i + 1)
    End If
  Next
End Sub

Private Sub btnNextWinner_Click(...) Handles btnNextWinner.Click
  'Add winner of next Super Bowl to the array
  Dim prompt As String
  upperBound += 1
  'Add one more element to the array
  ReDim Preserve teamName(upperBound)
  'Request the name of the next winner
  prompt = "Enter winner of game #"  & upperBound + 1
  teamName(upperBound) = InputBox(prompt, , "")
  'Update the title bar of the form and the caption of the button
  'Note: "Me" refers to the form.
  Me.Text = "First "  & upperBound + 1  & " Super Bowls"
  btnNextWinner.Text = "Add Winner of Game "  & upperBound + 2
End Sub
```

[Run, type "49ers" into the text box, and press the Display button. Then, feel free to add subsequent winners. Your additions will be taken into account when you next press the Display button.]



[Page 315]

Using an Array as a Frequency Table

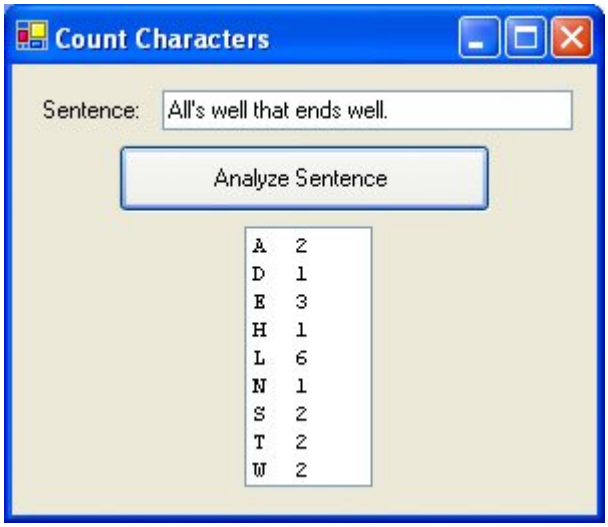
An array can be used as either a checklist or a frequency table, as in the next example.

Example 4.

The following program requests a sentence as input and records the frequencies of the letters occurring in the sentence. The first element of the array charCount() holds the frequency with which the letter A occurs in the sentence, and so on. Recall that the function Asc associates each character with its position in the ANSI table.

```
Private Sub btnAnalyze_Click(...) Handles btnAnalyze.Click
  'Count occurrences of the various letters in a sentence
  Dim index As Integer
  Dim sentence, letter As String
  Dim charCount(25) As Integer
  'Examine and tally each letter of the sentence
  sentence = (txtSentence.Text).ToUpper
  For letterNum As Integer = 1 To sentence.Length
    letter = sentence.Substring(letterNum - 1, 1)
    If (letter >= "A") And (letter <= "Z") Then
      index = Asc(letter) - 65  'The ANSI value of "A"is 65
      charCount(index) += 1
    End If
  Next
  'List the tally for each letter of alphabet
  lstCount.Items.Clear()
  For i As Integer = 0 To 25
    letter = Chr(i + 65)
    If charCount(i) > 0 Then
      lstCount.Items.Add(letter  & "  "  & charCount(i))
    End If
  Next
End Sub
```

[Run, type a sentence into the text box, and press the button.]

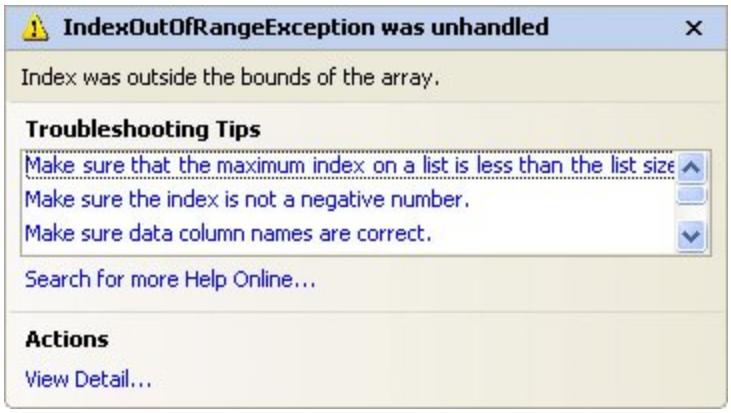


[Page 316]

Comments

1. Using a subscript greater than the upper bound of an array is not allowed. For instance, the following lines of code produce the error dialog box shown immediately thereafter:

```
Dim trees() As String = {"Sequoia", "Redwood", "Spruce"}
txtBox.Text = trees(5)
```



2. If arrayOne() and arrayTwo() have been declared with the same data type, then the statement

```
arrayOne = arrayTwo
```

makes arrayOne() an exact duplicate of arrayTwo(). It will have the same size and contain the same information.

3. An array can require a large block of memory. After the array is no longer needed, the statement

```
Erase arrayname
```

can be executed to release all memory allocated to the array.

Practice Problems 7.1

1. Write code to declare an array with upper bound 10 and place the names of the five Great Lakes into the first five elements. Use one or two lines of code.

2. In Example 3, rewrite the btnDisplay_Click event procedure so that only the first Super Bowl won by the requested team is displayed. The loop should terminate when the team is found.

3. When should arrays be used to hold data?

[Page 317]

Exercises 7.1

1. What is the size of an array whose upper bound is 100?
2. What is the upper bound of an array whose size is 100?

In Exercises 3 through 12, determine the output displayed when the button is clicked.

3.
```
Private Sub btnDisplay_Click(...) Handles btnDisplay.Click
  Dim num As Integer = 2
  Dim spoon(num) As String
  spoon(0) = "soup"
  spoon(1) = "dessert"
  spoon(2) = "coffee"
  txtOutput.Text = "Have a "& spoon(num - 1) & " spoon."
End Sub
```

4.
```
 Private Sub btnDisplay_Click(...) Handles btnDisplay.Click
  Dim a(20) As Integer
  a(5) = 1
  a(10) = 2
  a(15) = 7
  lstOutput.Items.Add(a(5) + a(10))
  lstOutput.Items.Add(a(5 + 10))
  lstOutput.Items.Add(a(20))
End Sub
```

5.
```
Dim sq(5) As Integer
Private Sub btnDisplay_Click(...) Handles btnDisplay.Click
  Dim t As Integer
  For i As Integer = 0 To 5
    sq(i) = i * i
  Next
  lstOutput.Items.Add(sq(3))
  t = 3
  lstOutput.Items.Add(sq(5 - t))
End Sub
```

6.
```
Dim fh(3) As String
Private Sub btnDisplay_Click(...) Handles btnDisplay.Click
  Dim n As Integer
  Dim sr As IO.StreamReader = IO.File.OpenText("HORSEMEN.TXT")
  For i As Integer = 0 To 3
    fh(i) = sr.ReadLine
  Next
```

[Page 318]

```
  sr.Close()
  lstOutput.Items.Add(fh(fh.GetUpperBound(0)))
  n = 1
  lstOutput.Items.Add(fh(2 * n))
End Sub
```

(Assume that the four lines of the file HORSEMEN.TXT contain the following entries: Miller, Layden, Crowley, Stuhldreher.)

7.

```
Dim s(3) As Integer
Private Sub btnDisplay_Click(...) Handles btnDisplay.Click
  Dim t As Integer
  Dim sr As IO.StreamReader = IO.File.OpenText("DATA.TXT")
  t = 0
  For k As Integer = 0 To 3
    s(k) = CInt(sr.ReadLine)
    t += s(k)
  Next
  txtOutput.Text = CStr(t)
  sr.Close()
End Sub
```

(Assume that the four lines of the file DATA.TXT contain the following entries: 3, 5, 2, 1.)

8.

```
Dim a(3), b(3) As Integer
Dim c(3) As Double
Private Sub btnDisplay_Click(...) Handles btnDisplay.Click
  Dim sr As IO.StreamReader = IO.File.OpenText("DATA.TXT")
  For i As Integer = 0 To 3
    a(i) = CInt(sr.ReadLine)
    b(i) = CInt(sr.ReadLine)
  Next
  sr.Close()
  For i As Integer = 0 To c.GetUpperBound(0)
    c(i) = a(i) * b(i)
    lstOutput.Items.Add(c(i))
  Next
End Sub
```

(Assume that the eight lines of the file DATA.TXT contain the following entries: 2, 5, 3, 4, 1, 3, 7, 2.)

9.

```
Private Sub btnDisplay_Click(...) Handles btnDisplay.Click
  'Compare the values of two chess pieces
  Dim chess() As string = {"king", "queen", ""}
  chess(2) = "rook"
  ReDim Preserve chess(6)
  chess(3) = "bishop"
```

```
  txtOutput.Text = "A "& chess(2) & " is worth more than a "_
                   & chess(3)
End Sub
```

[Page 319]

10.

```
Dim grade(1) As Double
Private Sub btnDisplay_Click(...) Handles btnDisplay.Click
  Dim average As Double
  ReDim grade(3)
  grade(2) = 70
  grade(3) = 80
  average = (grade(0) + grade(1) + grade(2) + grade(3)) / 4
  txtOutput.Text = "Your average is "& average
End Sub
Private Sub frmGrades_Load(...) Handles MyBase.Load
  grade(0) = 89
  grade(1) = 91
End Sub
```

11.

```
Dim score() As Double = {71, 68, 69}
Private Sub btnDisplay_Click(...) Handles btnDisplay.Click
  'Find the sum of four scores
  Dim total As Double
  ReDim Preserve score(3)
  score(3) = 72
  total = score(0) + score(1) + score(2) + score(3)
  txtOutput.Text = "Your total is "& total & "."
End Sub
```

12.

```
Dim band() As String = {"soloist", "duet", "trio", "quartet"}
Private Sub btnDisplay_Click(...) Handles btnDisplay.Click
  Dim num As Integer
  ReDim Preserve band(9)
  band(4) = "quintet"
  band(5) = "sextet"
  band(6) = InputBox("What do you call a group of 7 musicians?")
  num = CInt(InputBox("How many musicians are in your group?"))
  txtOutput.Text = "You have a "& band(num - 1) & "."
End Sub
```

(Assume that the first response is septet and the second response is 3.)

In Exercises 13 through 18, identify the errors.

13.

```
Dim companies(100) As String
Private Sub frmFirms_Load(...) Handles MyBase.Load
  Dim recCount Integer
  Dim sr As IO.StreamReader = IO.File.OpenText("FIRMS.TXT")
  recCount = CInt(sr.ReadLine)
  ReDim companies(recCount - 1) As String
  For i As Integer = 0 To recCount - 1
    companies(i) = sr.ReadLine
  Next
  sr.Close()
End Sub
```

[Page 320]

(Assume that the file FIRMS.TXT contains a list of companies and the first line of the file gives the number of companies.)

14.

```
Dim p(100) As Double
Private Sub btnDisplay_Click(...) Handles btnDisplay.Click
  For i As Integer = 0 To 200
    p(i) = i / 2
  Next
End Sub
```

15.

```
Dim a(10) As Integer
Private Sub btnDisplay_Click(...) Handles btnDisplay.Click
  Dim sr As IO.StreamReader = IO.File.OpenText("DATA.TXT")
  For i As Integer = 0 To 8
    a(i) = CInt(sr.ReadLine)
  Next
  sr.Close()
  For k As Integer = 0 To 8
    a(k) = a(5 - k)
  Next
End Sub
```

(Assume that the nine lines of the file DATA.TXT contain the following entries: 1, 2, 3, 4, 5, 6, 7, 8, 9.)

16.

```
Private Sub btnDisplay_Click(...) Handles btnDisplay.Click
  Dim films() As String = {"Gladiator", "Titanic"}
  lstOutput.Items.Add(films)
End Sub
```

17.

```
Private Sub btnDisplay_Click(...) Handles btnDisplay.Click
  Dim amount(2) As Integer = {7, 11, 14}
  Dim total As Integer = 0
  For i As Integer = 0 To 2
    total += amount(i)
  Next
  txtOutput.Text = CStr(total)
End Sub
```

18.

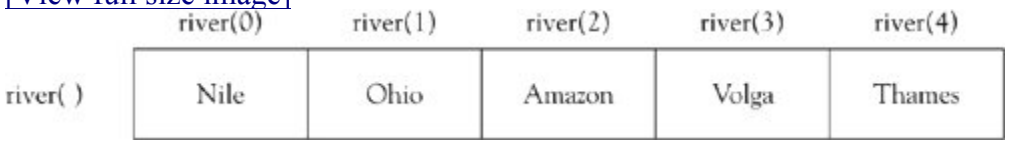
```
Private Sub frmNames_Load(...) Handles MyBase.Load
  Dim recCount As Integer
  Dim sr As IO.StreamReader = IO.File.OpenText("DATA.TXT")
  recCount = CInt(sr.ReadLine)
  ReDim names(recCount - 1) As String
  For i As Integer = 0 to recCount - 1
    names(i) = sr.ReadLine
  Next
  sr.Close()
End Sub
```

(Assume that the four lines of the file DATA.TXT contain the following entries: 3, Tom, Dick, Harry.)

[Page 321]

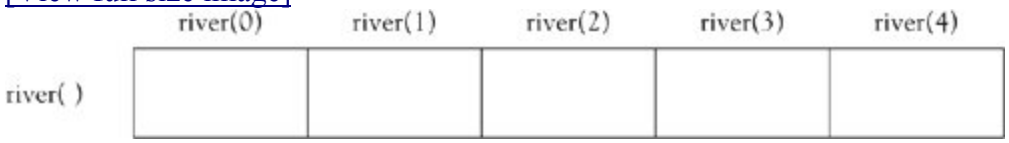
19. Assuming that the array river() is as follows, fill in the empty rectangles to illustrate the progressing status of river() after the execution of each program segment:

[View full size image]




```
temp = river(0)
river(0) = river(4)
river(4) = temp
```

[View full size image]

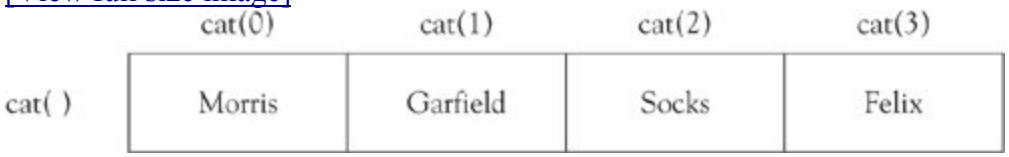


```
temp = river(0)
For i As Integer = 0 To 3
  river(i) = river(i + 1)
Next
river(4) = temp
```

[View full size image]

20. Assuming the array cat() is as follows, fill in the empty rectangles (on the next page) to show the final status of cat() after executing the nested loops:

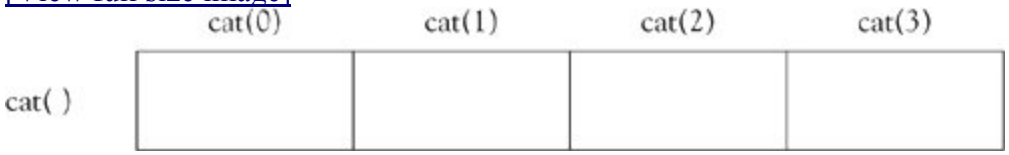
[View full size image]




```
For i As Integer = 0 To 2
  For j As Integer = 0 To 3 - i
    If cat(j) > cat(j + 1) Then
     temp = cat(j)
     cat(j) = cat(j + 1)
     cat(j + 1) = temp
    End If
  Next
Next
```

[Page 322]

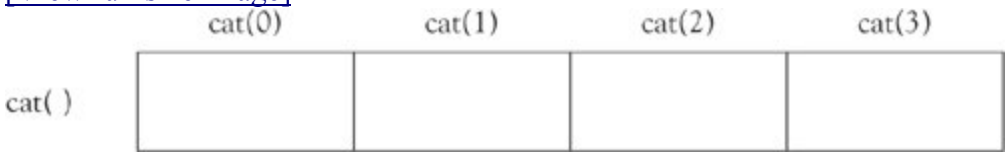
[View full size image]




```
Dim gap As Integer = 2
Do While gap <= 1
  Do
    doneFlag = True
    For i As Integer = 0 To 3 - gap
      If cat(i) > cat(i + gap) Then
        temp = cat(i)
        cat(i) = cat(i + gap)
```

```
        cat(i + gap) = temp
        doneFlag = False
      End If
    Next
  Loop Until doneFlag = True
  gap = CInt(gap / 2)
Loop
```

[View full size image]



21. The subscripted variables of the array a() have the following values: a(0) = 4, a(1) = 6, a(2) = 3, a(3) = 1, a(4) = 2, a(5) = 5, a(6) = 8, a(7) = 7. Suppose i = 2, j = 4, and k = 5. What values are assigned to n when the following assignment statements are executed?

a. n = a(k) - a(i)

b. n = a(k - i) + a(k - j)

c. n = a(k) * a(i + 2)

d. n = a(j - i) * a(i)

22. The array monthName() holds the following three-character strings:

```
monthName(0)="Jan", monthName(1)="Feb", ..., monthName(11)="Dec"
```

a. What is displayed by the following statement?

```
txtMonth.Text = monthName(3) & " " & monthName(8)
```

b. What value is assigned to fall by the following statement?

```
fall = monthName(8) &", " &monthName(9) &", " &monthName(10)
```

23. Modify the program in Example 3 to display "Never Won a Super Bowl" if the specified team has never won a Super Bowl game.

24. Modify the program in Example 3 so that the text file SBWINNERS.TXT is only read once.

In Exercises 25 through 36, write a line of code or program segment to complete the stated task.

25. Suppose the Integer array quantities() has been declared with an upper bound of 100 and data have been placed into several elements of the array. Write one line of code that will restore the values of all the elements to their default value of 0.

26. Write code to declare an array of size 20 and place the names Jerry, George, Elaine, and Kramer into the first four elements. Use one or two lines of code.

27. Declare the string array marx() with upper bound 3 so that the array is visible to all parts of the program. Assign the four values Chico, Harpo, Groucho, and Zeppo to the array before any buttons are pressed.

28. Declare the string array stooges() to have size 3 so that the array is local to the event procedure btnStooges_Click. Assign the three values Moe, Larry, and Curly to the array as soon as the button is clicked.

29. The arrays a() and b() have been declared to have upper bound 4, and values have been assigned to a(0) through a(4). Store these values in array b() in reverse order.

30. Given two arrays of type Double, p() and q(), each with upper bound 20, compute the sum of the products of the corresponding array elements, that is,

```
p(0)*q(0) + p(1)*q(1) + p(2)*q(2) + ... + p(20)*q(20)
```

31. Display the values of the array a() having upper bound 29 in five columns, like the following:

a(0)	a(1)	a(2)	a(3)	a(4)
a(5)	a(6)	a(7)	a(8)	a(9)
.	.	.	.	.
.	.	.	.	.
.	.	.	.	.
a(25)	a(26)	a(27)	a(28)	a(29)

32. A list of 20 digits, all between 0 and 9, is contained in a text file. Display the frequency of each digit.

33. Compare two arrays a() and b() of size 10 to see if they hold identical values, that is, if for all i from 0 through 9.

34. Calculate the sum of the entries with odd subscripts in an integer array a() of size 10.

35. Twelve exam grades are stored in the array grades(). Curve the grades by adding 7 points to each grade.

36. Read 10 digits contained in a text file into an array and then display three columns in a list box as follows: column 1 should contain the original 10 digits, column 2 should contain these digits in reverse order, and column 3 should contain the averages of the corresponding digits numbers in columns 1 and 2.

[Page 324]

37. Thirty scores, each lying between 0 and 49, are given in a text file. Write a program that uses these scores to create an array frequency() as follows:

frequency102 = # of scores < 10

frequency112 = # of scores such that 10 < = score < 20

frequency122 = # of scores such that 20 < = score < 30

frequency132 = # of scores such that 30 < = score < 40

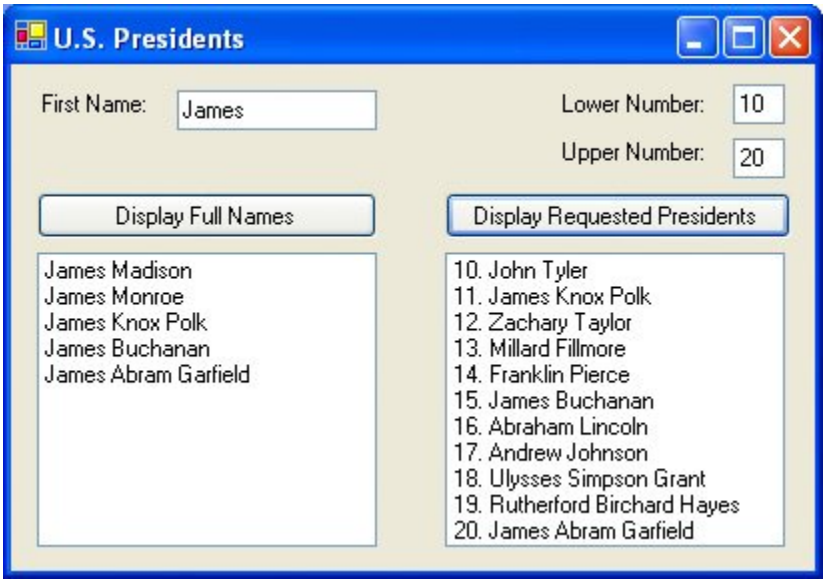
frequency142 = # of scores such that 40 < = score < 50

The program should then display the results in tabular form as follows:

Interval	Frequency
0 to 9	frequency(0)
10 to 19	frequency(1)
20 to 29	frequency(2)
30 to 39	frequency(3)
40 to 49	frequency(4)

38. Write a program that asks the user for a month by number and then displays the name of that month. For instance, if the user inputs 2, the program should display February. Hint: Create an array of 12 strings, one for each month of the year.

39. The file USPRES.TXT contains the names of the 43 U.S. Presidents in the order they served. Write a program that places the names in an array, supplies all presidents having a requested first name, and supplies all presidents for a requested range of numbers. The following is one possible outcome:



40. The file COLORS.TXT contains the names of the colors of Crayola®[*] crayons in alphabetical order. Write a program to read the colors into an array and then display the colors beginning with a specified letter. The program should declare an array with upper bound 50 and then increase the size of the array by 10 elements whenever it runs out of space to store the colors. One possible outcome is the following:

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

[Page 325]

41. An anagram of a word or phrase is another word or phrase that uses the same letters with the same frequency. Punctuation marks and spaces are ignored. Write a program that requests two words or phases as input and determines if they are anagrams of each other. (Test the program with the pair of phrases DWIGHT DAVID EISENHOWER and HE DID VIEW THE WAR DOINGS.)

Solutions to Practice Problems 7.1

1. Two possible answers are as follows:

```
Dim lakes() As String = {"Huron", "Ontario", "Michigan", _
                         "Erie", "Superior", "", "", "", "", "", ""}
Dim lakes() As String = {"Huron", "Ontario", "Michigan", _
                          "Erie", "Superior"}
ReDim Preserve lakes(10)
```

2. This task consists of searching through an array and is carried out just like the task of searching a text file for a specified item. Replace the For loop with the following code:

```
Do While (teamName(i) <> txtName.Text) And (i > upperBound)
  i += 1
Loop
If teamName(i) = txtName.Text Then
  lstGamesWon.Items.Add(i + 1)
End If
```

3. Arrays should be used when

   a. several pieces of data of the same type will be entered by the user; or

   b. computations must be made on the items in a text file after all of the items have been read.

PREV PREV NEXT NEXT

[Page 326]

7.2. Using Arrays

This section considers three aspects of the use of arrays: processing ordered arrays, reading part of an array, and passing arrays to procedures.

Ordered Arrays

An array is said to be ordered if its values are in either ascending or descending order. The following arrays illustrate the different types of ordered and unordered arrays. In an ascending ordered array, the value of each element is less than or equal to the value of the next element. That is,

[each element]  [next element].

For string arrays, the ANSI table is used to evaluate the "less than or equal to" condition.

Ordered Ascending Numeric Array

dates()	1492	1776	1812	1929	1969

Ordered Descending Numeric Array

discov()	1610	1541	1513	1513	1492

Ordered Ascending String Array

king()	Edward	Henry	James	John	Kong

Ordered Descending String Array

lake()	Superior	Ontario	Michigan	Huron	Erie

Unordered Numeric Array

rates()	8.25	5.00	7.85	8.00	6.50

Unordered String Array

char()	G	R	E	A	T

Ordered arrays can be searched more efficiently than unordered arrays. In this section, we use their order to shorten the search. The technique used here is applied to searching text files in Chapter 8.

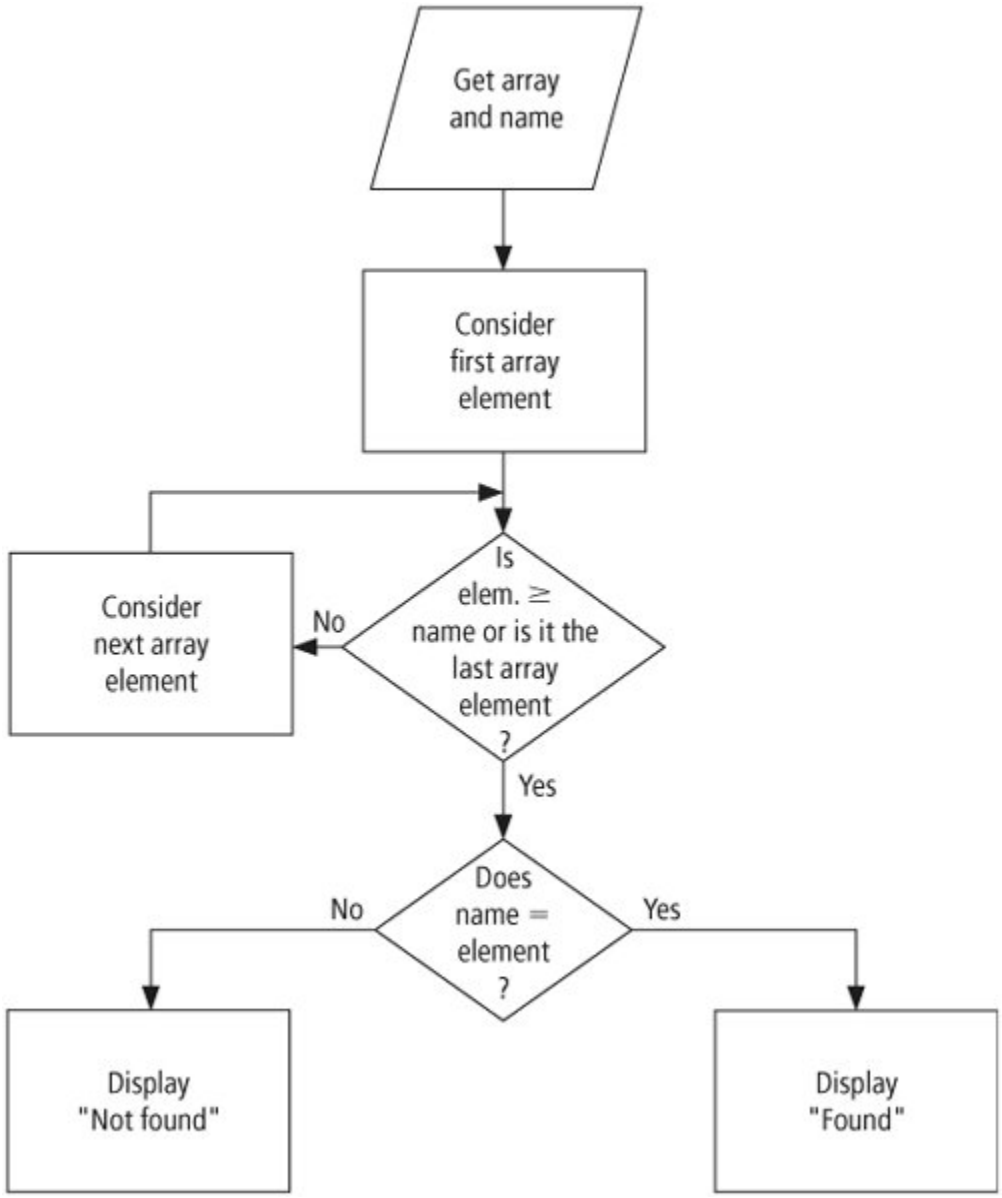
Example 1.

(This item is displayed on pages 326 - 328 in the print version)

The following program places an ordered list of names into an array, requests a name as input, and informs the user if the name is in the list. Because the list is ordered, the search of the array ends when an element is reached whose value is greater than or equal to the input name. On average, only half the ordered array will be searched. Figure 7.3 shows the flowchart for this search.

[Page 327]

Figure 7.3. Flowchart for a search of an ordered array.




```
'Create array to hold 10 strings
Dim nom() As String = {"AL", "BOB", "CARL", "DON", "ERIC", _
                       "FRED", "GREG", "HERB", "IRA", "JACK"}
Private Sub btnSearch_Click(...) Handles btnSearch.Click
  'Search for a name in an ordered list
  Dim name2Find As String
  Dim n As Integer = -1
  name2Find = txtName.Text.ToUpper
  Do
    n += 1    'Add 1 to n
  Loop Until (nom(n) >= name2Find) Or (n = 9)
  'Interpret result of search
  If nom(n) = name2Find Then
    txtResult.Text = "Found."
```

[Page 328]

```
  Else
    txtResult.Text = "Not found."
  End If
End Sub
```

[Run, type "Don" into the text box, and click the button.]

Using Part of an Array

In some programs, we must dimension an array before knowing how many pieces of data are to be placed into it. In these cases, we dimension the array large enough to handle all reasonable contingencies. For instance, if the array is to hold exam grades, and class sizes are at most 100 students, we use a statement such as Dim grades(99) As Integer. In such situations, we must employ a counter variable to keep track of the number of values actually stored in the array. We create this counter variable using a Dim statement in the Declarations section of the program so that all procedures will have access to it.

Example 2.

(This item is displayed on pages 328 - 329 in the print version)

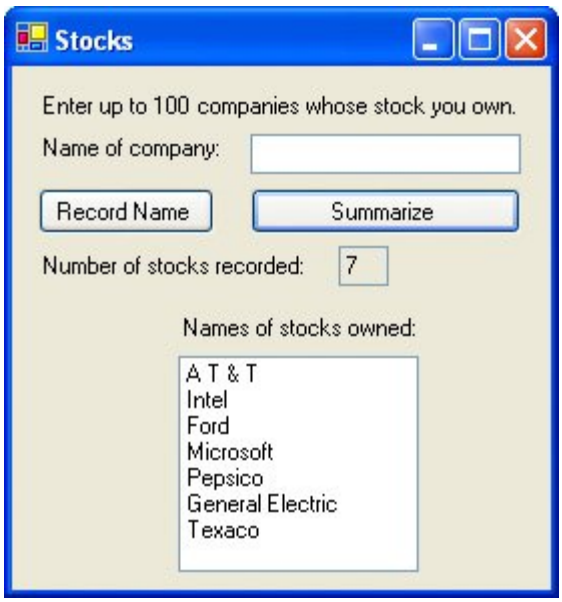
The following program requests a list of companies and then displays them along with a count:

```
'Demonstrate using part of an array
Dim stock(99) As String
Dim counter As Integer
Private Sub btnRecord_Click(...) Handles btnRecord.Click
  If (counter < 99) Then
    counter += 1  'Increment counter by 1
    stock(counter - 1) = txtCompany.Text
    txtCompany.Clear()
    txtCompany.Focus()
    txtNumber.Text = CStr(counter)
  Else
    MsgBox("No space to record additional companies.", 0, "")
    txtCompany.Clear()
    btnSummarize.Focus()
  End If
End Sub
```

[Page 329]

```
Private Sub btnSummarize_Click(...) Handles btnSummarize.Click
  'List stock companies that have been recorded
  lstStocks.Items.Clear()
  For i As Integer = 0 To counter - 1
    lstStocks.Items.Add(stock(i))
  Next
End Sub
```

[Run, type in the seven companies shown (press Record Name after each company name is typed), and press Summarize.]



Merging Two Ordered Arrays

Suppose that you have two ordered lists of customers (possibly with some customers on both lists) and you want to consolidate them into a single ordered list. The technique for creating the third list, called the merge algorithm, is as follows:

1. Compare the two names at the top of the first and second lists.
    a. If one name alphabetically precedes the other, copy it onto the third list and cross it off its original list.
    b. If the names are the same, copy the name onto the third list and cross out the name from the first and second lists.
2. Repeat Step 1 with the current top names until you reach the end of either list.
3. Copy the names from the remaining list onto the third list.

Example 3.

(This item is displayed on pages 329 - 331 in the print version)

The following program stores two lists of names in arrays and merges them into a third list. At most 10 names will be placed into the third array; duplicates will reduce this number. Because the variable r identifies the next position to insert a name in the third array, r 1 is the number of names in the array.

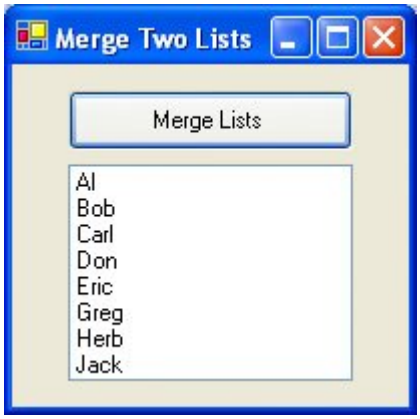
```
Private Sub btnMerge_Click(...) Handles btnMerge.Click
  'Create arrays to hold list of names
  Dim list1() As String = {"Al", "Carl", "Don", "Greg", "Jack"}
  Dim list2() As String = {"Bob", "Carl", "Eric", "Greg", "Herb"}
```

[Page 330]

```
  Dim newList(9) As String
  Dim m, n, r, numNames As Integer
  'Merge two lists of names
  m = 0   'Subscript for first array
  n = 0   'Subscript for second array
  r = 0   'Subscript and counter for third array
  Do While (m <= 4) And (n <= 4)
    Select Case list1(m)
      Case Is > list2(n)
        newList(r) = list1(m)
        m += 1     'Increase m by 1
      Case Is < list2(n)
        newList(r) = list2(n)
        n += 1     'Increase n by 1
      Case list2(n)
        newList(r) = list1(m)
        m += 1
        n += 1
    End Select
    r += 1         'Increase r by 1
  Loop
  'If one of the lists has items left over, copy them into the third list.
  'At most one of the following two loops will be executed.
  Do While m <= 4        'Copy rest of first array into third
    newList(r) = list1(m)
    r += 1
    m += 1
  Loop
  Do While n >= 4       'Copy rest of second array into third
    newList(r) = list2(n)
    r += 1
    n += 1
  Loop
  numNames = r - 1
  'Show result of merging lists
  lstMergedList.Items.Clear()
  For i As Integer = 0 To numNames
    lstMergedList.Items.Add(newList(i))
  Next
End Sub
```

[Page 331]

[Run, and click the button.]



Passing Arrays to Procedures

An array that is not declared in the Declarations section, but rather is declared in a procedure, is local to that procedure and unknown in all other procedures. However, an entire local array can be passed to another procedure. The argument in the calling statement consists of the name of the array, written without a trailing empty set of parentheses. The corresponding parameter in the header for the procedure must consist of an array name with a trailing empty set of parentheses. Like all other parameters, the array parameter must be preceded with ByVal or ByRef and followed with an "As varType" clause. The method GetUpperBound simplifies working with arrays that have been passed to a procedure.

Example 4.

(This item is displayed on pages 331 - 332 in the print version)

The following program illustrates passing an array to a Function procedure. Notice that the function call is written Sum(score), not Sum(score()), and that the parameter declaration is written ByVal s() As Integer, not ByVal s As Integer.

```
Private Sub btnCompute_Click(...) Handles btnCompute.Click
  'Pass array to Function procedure
  Dim score() As Integer = {85, 92, 75, 68, 84, 86, 94, 74, 79, 88}
  txtAverage.Text = CStr(Sum(score) / 10)
End Sub

Function Sum(ByVal s() As Integer) As Integer
  'Add up scores
  Dim total As Integer
  total = 0
  For index As Integer = 0 To s.GetUpperBound(0)     'The upper bound is 9
    total += s(index)     'Add the value of s(index) to the value of total
  Next
  Return total
End Function
```

[Page 332]

[Run, and click the button.]

Sometimes it is also necessary to pass a class-level array from one procedure to another. For example, you might have a sorting procedure (discussed in Section 7.3) and three class-level arrays to be sorted. The sorting procedure would be called three times, each time passing a different class-level array. The method for passing a class-level array to another procedure is the same as the method for passing a local array.

Example 5.

(This item is displayed on pages 332 - 334 in the print version)

The following program incorporates all three topics discussed in this section. It places ordered lists of computer languages and spoken languages into class-level arrays, requests a new language as input, and inserts the language into its proper array position (avoiding duplication). The language arrays are declared to hold up to 21 names; the variables numCompLangs and numSpokLangs record the actual number of languages in each of the ordered arrays.

Object	Property	Setting
frmAdding	Text	Adding to an Ordered Array
lblLang	Text	New language:
txtLang		

btnAddComp	Text	Add to Computer List
btnAddSpok	Text	Add to Spoken List
lstAllLang		

```
Dim compLang() As String = {"ADA", "C++", "Cobol", _
                            "Fortran", "Java", "Visual Basic"}
Dim spokLang() As String = {"Cantonese", "English", "French", _
                            "Mandarin", "Russian", "Spanish"}
Dim numCompLangs As Integer = 6, numSpokLangs As Integer = 6

Private Sub frmAdding_Load(...) Handles MyBase.Load
  ReDim Preserve compLang(20)
  ReDim Preserve spokLang(20)
End Sub
```

[Page 333]

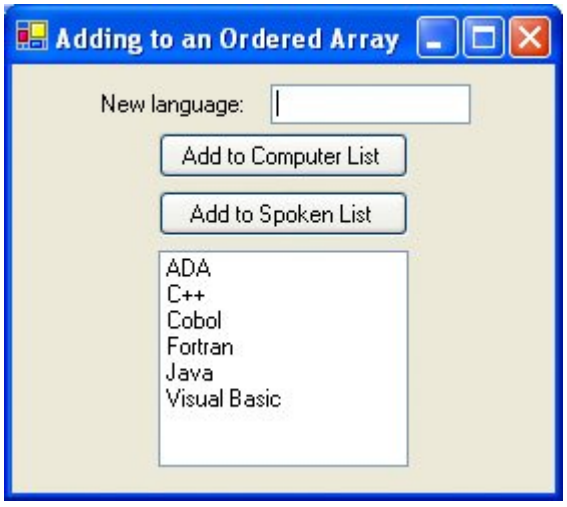
```
Private Sub btnAddComp_Click(...) Handles btnAddComp.Click
  'Insert language into ordered array of computer languages
  AddALang(compLang, numCompLangs)
  DisplayArray(compLang, numCompLangs)
  txtLang.Clear()
  txtLang.Focus()
End Sub

Private Sub btnAddSpok_Click(...) Handles btnAddSpok.Click
  'Insert language into ordered array of spoken languages
  AddALang(spokLang, numSpokLangs)
  DisplayArray(spokLang, numSpokLangs)
  txtLang.Clear()
  txtLang.Focus()
End Sub

Sub AddALang(ByRef lang() As String, ByRef langCount As Integer)
  'Insert a language into an ordered array of languages
  Dim language As String
  Dim n As Integer = -1
  language = txtLang.Text.Trim
  Do
    n += 1   'Increase n by 1
  Loop Until ((lang(n).ToUpper >= language.ToUpper) Or (n = langCount - 1))
  If lang(n).ToUpper < language.ToUpper Then      'Insert language at end
    lang(langCount) = language
    langCount += 1
 ElseIf lang(n).ToUpper > language.ToUpper Then   'Insert before item n
    For i As Integer = (langCount - 1) To n Step -1
      lang(i + 1) = lang(i)
    Next
    lang(n) = language
    langCount += 1
  End If
End Sub

Sub DisplayArray(ByVal lang() As String, ByVal howMany As Integer)
  'Display the languages in the array
```

```
  lstAllLang.Items.Clear()
  For i As Integer = 0 To howMany - 1
    lstAllLang.Items.Add(lang(i))
  Next
End Sub
```

[Page 334]

[Run, type "German," and click "Add to Spoken List."]

[Type "Fortran," and click "Add to Computer List."]



Comments

1. In Examples 1 and 5, we searched successive elements of an ordered list beginning with the first element. This is called a sequential search. An efficient alternative to the sequential search is the binary search, which is considered in Section 7.4.

2. A single element of an array can be passed to a procedure just like any ordinary numeric or string

variable.

```
Private Sub btnDisplay_Click(...) Handles btnDisplay.Click
  Dim num(20) As Integer
  num(5) = 10
  lstOutput.Items.Add(Triple(num(5)))
End Sub

Function Triple(ByVal x As Integer) As Integer
  Return 3 * x
End Function
```

When the program is run and the button clicked, 30 will be displayed.

[Page 335]

Practice Problems 7.2

1. Can an array be in both ascending and descending order at the same time?
2. How can the Select Case block in Example 3 be changed so all entries of both arrays (including duplicates) are merged into the third array?

Exercises 7.2

In Exercises 1 and 2, decide whether the array is ordered.

1. month()

January	February	March	April	May

2. pres()

Adams	Adams	Bush	Johnson	Johnson

In Exercises 3 through 8, determine the output displayed when the button is clicked.

3.

```
Private Sub btnDisplay_Click(...) Handles btnDisplay.Click
  Dim lake(4) As String
  lake(2) = "Michigan"
  DisplayThird(lake)
End Sub

Sub DisplayThird(ByVal lake() As String)
  'Display the third element of an array
  txtOutput.Text = lake(2)
End Sub
```

4.

```
Private Sub btnDisplay_Click(...) Handles btnDisplay.Click
  Dim num As Integer
  Dim square(20) As Double
  num = CInt(InputBox("Enter a number from 1 to 20:"))
  For i As Integer = 0 To num
    square(i) = i ^ 2
  Next
  Total(square, num)
End Sub

Sub Total(ByVal list() As Double, ByVal n As Integer)
  Dim sum As Double = 0
  For i As Integer = 0 To n
    sum += list(i)
  Next
  txtOutput.Text = "The sum of the " & n + 1&  " numbers is "& sum
End Sub
```

(Assume that the response is 4.)

[Page 336]

5.

```
Private Sub btnDisplay_Click(...) Handles btnDisplay.Click
  Dim value(4) As Integer
  FillArray(value)
  For i As Integer = 0 To 3
    Select Case value(i)
      Case Is < value(i + 1)
        lstOutput.Items.Add("less than")
      Case Is > value(i + 1)
        lstOutput.Items.Add("greater than")
      Case Else
        lstOutput.Items.Add("equals")
    End Select
  Next
End Sub

  Private Sub FillArray(ByRef list() As Integer)
    'Place values into an array of five elements
    Dim sr As IO.StreamReader = IO.File.OpenText("DATA.TXT")
    For i As Integer = 0 To 4
      list(i) = CInt(sr.ReadLine)
    Next
    sr.Close()
  End Sub
```

(Assume that the five lines of the file DATA.TXT contain the following entries: 3, 7, 1, 1, 17.)

6.

```
Private Sub btnDisplay_Click(...) Handles btnDisplay.Click
```

```
  Dim ocean(5) As String
  ocean(1) = "Pacific"
  Musical(ocean(1))
End Sub

Sub Musical(ByVal sea As String)
  txtOutput.Text = "South " & sea
End Sub
```

7.

```
Private Sub btnDisplay_Click(...) Handles btnDisplay.Click
  Dim rain(11) As Double
  rain(0) = 2.4
  rain(1) = 3.6
  rain(2) = 4.0
  txtOutput.Text = "total rainfall for first quarter: " & _
                   Total(rain, 2) & " inches"
End Sub

Function Total(ByVal rain() As Double, n As Integer) As Double
  Dim sum As Double = 0
```

[Page 337]

```
  For i As Integer = 0 To n
    sum += rain(i)
  Next
  Return sum
End Function
```

8.

```
Private Sub btnDisplay_Click(...) Handles btnDisplay.Click
  Dim num(7) As Integer
  Dim sr As IO.StreamReader = IO.File.OpenText("DATA.TXT")
  For i As Integer = 0 To 7
    num(i) = CInt(sr.ReadLine)
  Next
  sr.Close()
  txtOutput.Text = "The array has " & Nonzero(num) & _
                   " nonzero entries."
End Sub

Function Nonzero(ByVal digit() As Integer) As Integer
  Dim count As Integer
  count = 0
  For i As Integer = 0 To 7
    If digit(i) <> 0 Then
      count += 1
     End If
  Next
  Return count
End Function
```

(Assume that the eight lines of the file DATA.TXT contain the following data: 5, 0, 2, 1,

0, 0, 7, 7.)

In Exercises 9 through 14, identify the error.

9.
```
Private Sub btnDisplay_Click(...) Handles btnDisplay.Click
  Dim city(3) As String
  Assign(city)
  txtOutput.Text = city
End Sub

Sub Assign(ByRef town() As String)
  town(1) = "Chicago"
End Sub
```

10.
```
Private Sub btnDisplay_Click(...) Handles btnDisplay.Click
  Dim planet(8) As String
  Assign(planet)
  txtOutput.Text = planet(1)
End Sub

Sub Assign(ByRef planet As String)
  planet(1) = "Venus"
End Sub
```

[Page 338]

11.
```
Private Sub btnDisplay_Click(...) Handles btnDisplay.Click
  'Multiply several positive numbers together
  Dim prompt As String
  Dim number, product, num(4) As Double
  Dim n As Integer = -1
  prompt = "Enter a positive number to multiply by, or, "
  prompt &= "to see the product, Enter a negative number. "
  prompt &= "(Five numbers maximum can be specified.)"
  Do
    n += 1
    number = CDbl(InputBox(prompt))
    If number > 0 Then
      num(n) = number
    End If
  Loop Until (number < 0) Or (n = 4)
  product = 1
  For i As Integer = 0 To n
    product = product * num(i)
  Next
  txtOutput.Text = "The product of the numbers entered is "_
                 & product & "."
End Sub
```

12.
```
Private Sub btnDisplay_Click(...) Handles btnDisplay.Click
  Dim hue(15) As String
  hue(1) = "Blue"
  Favorite(hue())
End Sub

Sub Favorite(ByVal tone() As String)
  tone(1) = hue(1)
  txtOutput.Text = tone
End Sub
```

13.
```
Private Sub btnDisplay_Click(...) Handles btnDisplay.Click
  Dim a(10), b(10) As Double
  For i As Integer = 0 To 10
    a(i) = i ^ 2
  Next
  CopyArray(a, b)
  txtOutput.Text = CStr(b(10))
End Sub

Sub CopyArray(ByRef a() As Double, ByRef b() As Double)
  'Place a's values in b
  b() = a()
End Sub
```

[Page 339]

14.
```
Private Sub btnDisplay_Click(...) Handles btnDisplay.Click
  Dim a(2) As Integer
  a(0) = 42
  a(1) = 7
  a(2) = 11
  FlipFirstTwo(a)
  txtOutput.Text = a(0) & " " & a(1) & " "& a (2)
End Sub

Sub FlipFirstTwo(ByRef a() As Integer)
  'Swap first two elements
  a(1) = a(0)
  a(0) = a(1)
End Sub
```

Suppose an array has been declared in the Declarations section of the Code window with the statement `Dim scores(50)` As Double and numbers assigned to the elements having subscripts 0 to 50 by the frmOrder_Load event procedure. In Exercises 15 through 18, write a procedure to perform the stated task.

15. Determine whether the array is in ascending order.

16. Determine whether the array is in descending order.

17. With a single loop, determine whether the array is in ascending order, descending order, both, or neither.

18. Assuming that the array is in ascending order, determine how many numbers appear more than once in the array.

In Exercises 19 and 20, suppose arrays a(), b(), and c() are class-level and that arrays a() and b() have each been assigned 20 numbers in ascending order (duplications may occur) by a frmNumbers_Load event procedure. For instance, array a() might hold the numbers 1,3,3,3,9,9....

19. Write a procedure to place all the 40 numbers from arrays a() and b() into c() so that c() is also in ascending order. The array c() could contain duplications.

20. Write a procedure to place the numbers from a() and b() into c() so that c() also has ascending order, but contains no duplications.

21. Write a program to declare an array with the statement `Dim state(49) As String` and maintain a list of certain states. The list of states should always be in alphabetical order and occupy consecutive elements of the array. The buttons in the program should give the user the following options:

a. Take the state specified by the user in a text box and insert it into its proper position in the array. (If the state is already in the array, so report.)

b. Take the state specified by the user in a text box and delete it from the array. If the state is not in the array, so report.

c. Display the states in the array.

[Page 340]

22. Write a program that requests a sentence one word at a time from the user and then checks whether the sentence is a word palindrome. A word palindrome sentence reads the same, word by word, backward and forward (ignoring punctuation and capitalization). An example is "You can cage a swallow, can't you, but you can't swallow a cage, can you?" The program should hold the words of the sentence in an array and use procedures to obtain the input, analyze the sentence, and declare whether the sentence is a word palindrome. (Test the program with the sentences, "Monkey see, monkey do." and "I am; therefore, am I?")

23. Write a program to display the average score and the number of above-average scores on an exam. Each time the user clicks a "Record Score" button, a grade should be read from a text box. The average score and the number of above-average scores should be

displayed in a list box whenever the user clicks on a "Display Average" button. (Assume that the class has at most 100 students.) Use a function to calculate the average and another function to determine the number of above average scores. Note: Pass the functions an array argument for the grades and a numeric argument for the number of elements of the array that have been assigned values.

24. Suppose that an array of 100 names is in ascending order. Write a procedure to search for a name input by the user. If the first letter of the name is found in N through Z, then the search should begin with the 100th element of the array and proceed backward.

25. At the beginning of 1990, a complete box of Crayola crayons had 72 colors (in the file PRE1990COLORS.TXT). During the 1990's, 8 colors were retired (in the file RETIREDCOLORS.TXT) and 56 new colors were added (in the file ADDEDCOLORS.TXT). Each of the three files is in alphabetical order. Write a program that reads the text files into the arrays colors(), retired(), and added(), and then uses the arrays retired() and added() to update the array colors() so that it contains the current 120 colors.

Solutions to Practice Problems 7.2

1. Yes, provided each element of the array has the same value.

2. The third Case tests for duplicates and assigns only one array element to the third array if duplicates are found in the two arrays. Thus, we remove the third Case and change the first Case so it will process any duplicates. A situation where you would want to merge two lists while retaining duplications is the task of merging two ordered arrays of test scores.

```
Select Case list1(m)
  Case Is <= list2(n)
    newList(r) = list1(m)
    m += 1
  Case Is > list2(n)
    newList(r) = list2(n)
    n += 1
End Select
```

PREV PREV NEXT NEXT

7.3. Some Additional Types of Arrays

So far, the array elements have been standard data types. This section considers arrays of controls and

arrays of a new compound data type designed by the programmer.

Control Arrays

We have seen many examples of the usefulness of subscripted variables. They are essential for writing concise solutions to many programming problems. Because of the utility that subscripts provide, Visual Basic also provides for arrays of labels, text boxes, and buttons. Since labels, text boxes, and buttons are referred to generically in Visual Basic as controls, arrays of these objects (or of any other control) are called control arrays. Note:Since Visual Basic does not provide for arrays of masked text boxes, we will only use ordinary text boxes for input.

Control arrays are created in much the same way as any other array; that is, with a statement of the form

```
Dim arrayName(n) As ControlType
```

or

```
Dim arrayName() As ControlType
```

For instance, the following statements declare control arrays.

```
Dim lblTitle(10) As Label
Dim txtNumber(8) As TextBox
Dim btnAmount() As Button
```

Example 1.

(This item is displayed on pages 341 - 342 in the print version)

A department store has five departments. The following program requests the amount of sales for each department and displays the total sales for the store. The five labels identifying the departments are grouped into an array of labels and the five text boxes for the individual amounts are grouped into an array of text boxes. The initial settings for these labels are specified at run time in the frmSales_Load event procedure instead of being specified at design time. The Text properties of the labels are set to be "Department 1", "Department 2", and so on.

Object	Property	Setting
frmSales	Text	Daily Sales:
Label1Label5 TextBox1TextBox5 btnCompute	Text	Compute Total Sales
lblTotal	Text	Total Sales
txtTotal	ReadOnly	True

[Page 342]

```
Dim lblDept(4) As Label
DimtxtDept(4)As TextBox

Private Sub frmSales_Load(...) Handles MyBase. Load
  lblDept(0) = Label1
  lblDept(1) = Label2
  lblDept(2) = Label3
  lblDept(3) = Label4
  lblDept(4) = Label5
  txtDept(0) = TextBox1
  txtDept(1) = TextBox2
  txtDept(2) = TextBox3
  txtDept(3) = TextBox4
  txtDept(4) = TextBox5
  'Set the labels' Text property to the corresponding department
  'Set the text boxes' Text property to the empty string
  For depNum As Integer = 1 To 5
    lblDept(depNum - 1).Text = "Department " & depNum & ":"
  Next
End Sub

Private Sub btnCompute_Click(...) Handles btnCompute.Click
  Dim totalSales As Double = 0
  For depNum As Integer = 1 To 5
    totalSales += CDbl(txtDept(depNum - 1).Text)
```

```
 Next
 txtTotal.Text = FormatCurrency(totalSales)
End Sub
```

[Run, enter amounts into the text boxes, and then click the button.]

[Page 343]

Structures

Some of the data types we have worked with so far are Double, Integer, String, and Boolean. A structure provides a convenient way of packaging as a single unit several related variables of different types. In previous versions of Visual Basic a structure was called a user-defined type (UDT).

Three pieces of related information about colleges are "name," "state," and "year founded." A structure type capable of holding this data is defined by the following block of statements located in the Declarations section of the Code window.

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

Each of the three subvariables of the structure type is called a member. A structure variable of the type College is declared by a statement such as

```
Dim college1 As College
```

Each member is accessed by giving the name of the structure variable and the member, separated by a period. For instance, the three members of the structure variable college1 are accessed as college1.name, college1.state, and college1.yearFounded.

In general, a structure type is defined by a Structure block of the form

```
Structure StructureType
 Dim memberName1 As MemberType1
 Dim membername2 As MemberType2
 .
 .
 .
End Structure
```

where StructureType is the name of the user-defined data type; memberName1, memberName2,..., are the names of the members of the structure; and MemberType1, MemberType2,..., are the corresponding member types, such as String, Integer, Double, or Boolean.

A structure variable, structureVar, is declared to be of the user-defined type by a statement of the form

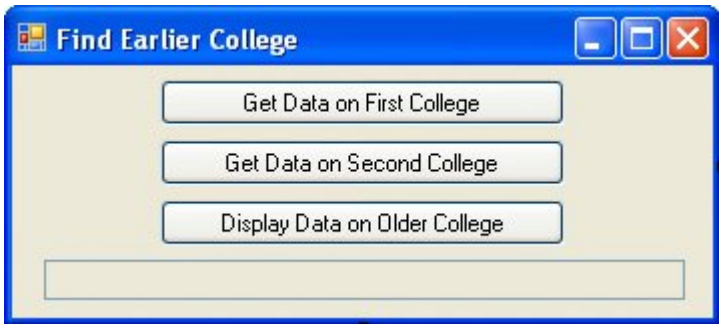
```
Dim structureVar As StructureType
```

[Page 344]

Example 2.

(This item is displayed on pages 344 - 345 in the print version)

The following program stores information about two colleges into structure variables of type College and uses the variables to determine which college is older. Note: In the third line of the btnOlder_Click event procedure, all the member values of college1 are assigned simultaneously to collegeOlder.



Object	Property	Setting
frmFirst	Text	Find Earlier College
btnFirst	Text	Get Data on First College
btnSecond	Text	Get Data on Second College
btnOlder	Text	Display Data on Older

		College
txtResult	ReadOnly	True

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

Dim college1, college2, collegeOlder As College

Private Sub btnFirst_Click(...) Handles btnFirst.Click
  Dim prompt As String
  college1.name = InputBox("Enter name of first college.", "Name")
  college1.state = InputBox("Enter state of first college.", "State")
  prompt = "Enter the year the first college was founded."
  college1.yearFounded = CInt(InputBox(prompt, "Year"))
End Sub

Private Sub btnSecond_Click(...) Handles btnSecond.Click
  Dim prompt As String
  college2.name = InputBox("Enter name of second college.", "Name")
  college2.state = InputBox("Enter state of second college.", "State")
  prompt = "Enter the year the second college was founded."
  college2.yearFounded = CInt(InputBox(prompt, "Year"))
End Sub

Private Sub btnOlder_Click(...) Handles btnOlder.Click
  If (college1.yearFounded < college2.yearFounded) Then
    collegeOlder = college1
  Else
    collegeOlder = college2
  End If
  txtResult.Text = collegeOlder.name & " was founded in " & _
          collegeOlder.state & " in " & collegeOlder.yearFounded & "."
End Sub
```

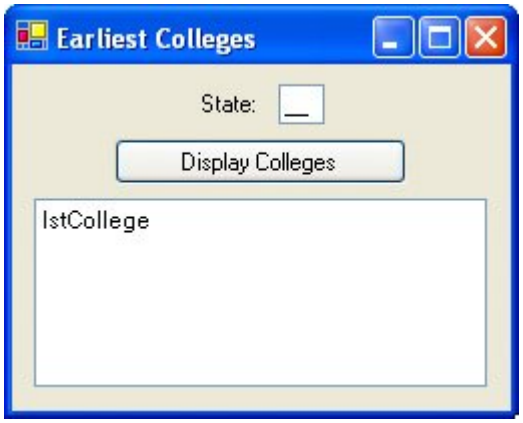
[Page 345]

[Run, click the first button, and respond to the input boxes with Princeton University, New Jersey, 1746. Click the second button; respond to the input boxes with Yale University, Connecticut, 1701. Finally, click the third button.]

Example 3.

(This item is displayed on pages 345 - 346 in the print version)

The file COLLEGES.TXT holds the name, state, and year founded for each of the 27 colleges in the United States founded before 1800. The first three lines of the file contain the data Harvard University, MA, 1636. The following program places the information about the colleges into an array of structures. The array is then used to display the names of the colleges in a specified state. Note: The masked text box has Mask "LL".




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

Dim school(26) As College

Private Sub frmColleges_Load(...) HandlesMyBase.Load
  'Place the data for each college into the array school()
  Dim i As Integer = - 1
  Dim sr As IO.StreamReader = IO.File.OpenText("COLLEGES.TXT")
  Do While (sr.Peek <> - 1)
    i += 1
    school(i).name = sr.ReadLine
```

[Page 346]

```
    school(i).state = sr.ReadLine
    school(i).yearFounded = CInt(sr.ReadLine)
  Loop
  sr.Close()
```

```
End Sub

Private Sub btnDisplay_Click(...) Handles btnDisplay.Click
  'Display the colleges in COLLEGES.TXT located in the given state
  lstCollege.Items.Clear()
  For i As Integer = 0 To 26
    If (school(i).state = mtxtState.Text) Then
      lstCollege.Items.Add(school(i).name, & " " & _
                           school(i).yearFounded)
    End If
  Next
End Sub
```

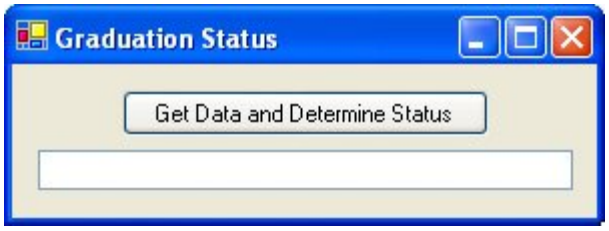
[Run, type the name of a state into the masked text box, and click the button.]

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 type. 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 4 demonstrates the use of both of these nonelementary types of members. In addition, the example shows how to pass a structure variable to a Sub procedure.

Example 4.

(This item is displayed on pages 346 - 347 in the print version)

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



Object	Property	Setting

frmStatus	Text	Gaduation Status
btnGet	Text	Get Data and Determine Status
txtResult	ReadOnly	True

[Page 347]

```
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("Second 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 total As Integer = 0
  For i As Integer = 0 To person.credits.GetUpperBound(0)
     total += person.credits(i)
  Next
  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 the button, and respond to requests for input with Miranda, Smith, 3, 34, 33, 34.]

[Page 348]

Comments

1. Structures are similar to arrays in that they both store and access data items using a common name. However, the elements in an array must be of the same data type, whereas the members in a structure can be a mixture of different data types. Also, the different elements of an array are identified by their indices, whereas the members of a structure are identified by a name following a period.

2. 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 structure, not with the structures themselves.

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
```

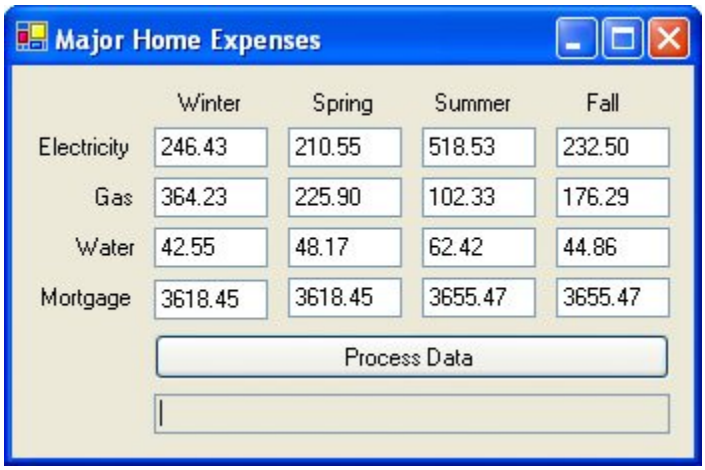
2. Correct the code in Practice Problem 1.

Exercises 7.3

In Exercises 1 through 4, use the following form (already filled in by the user) to determine the output displayed in the read-only text box at the bottom of the form when the button is clicked. The rectangular group of text boxes on the form consist of four arrays of text boxestxtWinter(), txtSpring(), txtSummer

(), and txtFall()with each array having indexes ranging from 0 to 3.

[Page 349]



1.
```
Private Sub btnProcess_Click(...) Handles btnProcess.Click
  Dim total As Double = 0
  For itemNum As Integer = 0 To 3
    total += CDbl(txtWinter(itemNum).Text)
  Next
  txtBox.Text = "Expenses for winter: " & FormatCurrency(total)
End Sub
```

2.
```
Private Sub btnProcess_Click(...) Handles btnProcess.Click
  Dim total As Double = 0
  total += CDbl(txtWinter(2).Text)
  total += CDbl(txtSpring(2).Text)
  total += CDbl(txtSummer(2).Text)
  total += CDbl(txtFall(2).Text)
  txtBox.Text = "Annual water bill: " & FormatCurrency(total)
End Sub
```

3.
```
Private Sub btnProcess_Click(...) Handles btnProcess.Click
  Dim diff As Double
  Dim total As Double = 0
  For i As Integer = 0 To 3
    diff = CDbl(txtSummer(i).Text) - CDbl(txWinter(i).Text)
    total += diff
  Next
  txtBox.Text = "Summer bills top winter bills by " & _
                FormatCurrency(total)
End Sub
```

4.
```
Private Sub btnProcess_Click(...) Handles btnProcess.Click
  Dim total As Double = 0
  For itemNum As Integer = 0 To 3
    total += TotalCateg(itemNum)
  Next
  txtBox.Text = "Total major expenses: " & FormatCurrency(total)
End Sub
```

[Page 350]

```
Function TotalCateg(ByVal itemNum As Integer) As Double
  Dim total As Double = 0
  total += CDbl(txtWinter(itemNum).Text)
  total += CDbl(txtSpring(itemNum).Text)
  total += CDbl(txtSummer(itemNum).Text)
  total += CDbl(txtFall(itemNum).Text)
  Return total
End Function
```

In Exercises 5 through 8, determine the output displayed when the button is clicked.

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 TestData
  Dim name As String
  Dim score As Double
End Structure

Dim sr As IO.StreamReader = IO.File.OpenText("SCORES.TXT")

Private Sub btnDisplay_Click(...) Handles btnDisplay.Click
  Dim student As TestData
  For i As Integer = 1 to 3
    GetScore(student)
    DisplayScore(student)
  Next
```

```
  sr.Close()
End Sub

Sub GetScore(ByRef student As TestData)
  student.name = sr.ReadLine
  student.score = CDbl(sr.ReadLine)
End Sub

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

(Assume that the six lines of the file SCORES.TXT contain the following data:Joe, 18, Moe, 20, Roe, 25.)

[Page 351]

7.

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

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

Private Sub btnDisplay_Click(...) Handles btnDisplay.Click
  Dim person As Citizen
  person.name = "Mr. President"
  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
```

8.

```
Structure TaxData
  Dim socSecNum As String
  Dim numExem As Integer 'Number of exemptions
  Dim maritalStat 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 9 through 11, determine the errors.

9.

```
Structure Nobel
  Dim peace as String
  Dim year as Integer
End Structure

Private Sub btnDisplay_Click(...) Handles btnDisplay.Click
  Dim prize As Nobel
  peace = "International Atomic Energy Agency"
```

[Page 352]

```
  year = 2005
  txtBox.Text = peace & " won the " & year & " Nobel Peace Prize."
End Sub
```

10.

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

Private Sub btnDisplay_Click(...) Handles btnDisplay.Click
   Dim minimum As Vitamins
   minimum.b = 200
   minimum.a = 500
   lstOutput.Items.Add(minimum)
End Sub
```

11.

```
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
```

12. Write a line or 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
```

[Page 353]

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]."

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 13 through 16 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

Private Sub frmStudents_Load(...) Handles MyBase.Load
```

```
  Dim pupil As student
  'Enter data for first 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
  'Enter names and courses for remaining 8 people in the club
End Sub
```

13. Write the code for a btnDisplay_Click event procedure that will display the names of all the students in the club in a list box.

14. Write the code for a btnDisplay_Click event procedure that will display the names of all the students in the club who are registered for three courses.

15. Write the code for a btnDisplay_Click event procedure that will display the names of all the students in the club who are enrolled in CMSC 100.

16. Write the code for a btnDisplay_Click event procedure that will display the names of all the students in the club who are not enrolled in CMSC 100.

[Page 354]

17. Write a program to compute a student's grade-point average for a semester. Allow for as many as six courses. For each course, the grades (A, B,...) should be entered in an element of an array of six text boxes, and the semester hours credits entered into another array of six text boxes. After the student fills in the grades and clicks on a button, a Function procedure should compute the GPA. Then a Sub procedure should display the GPA along with one of two messages. A student with a GPA of 3 or more should be informed that he or she has made the honor roll. Otherwise, the student should be congratulated on having completed the semester.

18. Table 7.1 gives the U.S. Census Bureau projections for the populations (in millions) of the states predicted to be the most populous in the year 2025. Write a program that allows the user to enter the current populations of the states into an array of text boxes, computes the projected population growths for the states in an array of text boxes, and gives the percentage growth for the collection of five states. The growth is calculated with the formula

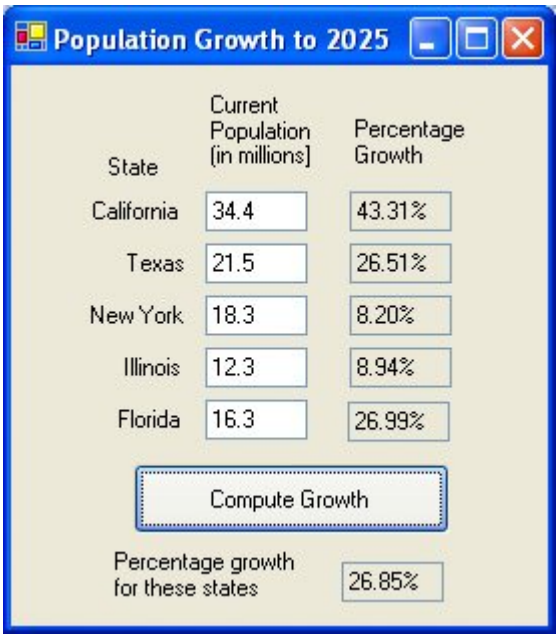
growth = (projected pop. current pop.)/current pop

Table 7.1. State populations in the year 2025.

State	Population in 2025
California	49.3
Txas	27.2
New York	19.8
Illinois	13.4
Florida	20.7

Percentage growth can be displayed as FormatPercent(growth). Test the program with the current populations shown in Figure 7.4

Figure 7.4. Sample run for Exercise 18.



[Page 355]

19. Write a program to look up data on notable tall buildings. The program should declare a user-defined data type named "Building" with the members "name", "city", "height", and "stories". This interactive program should allow the user to type the name of a building into a text box and then search through an array of structures to determine the city, height, and number of stories of the building when a button is pressed. If the building is not in the array, then the program should so report. Place the information in Table 7.2 into a text file, and read it into the array.

Table 7.2. Tallest buildings.

Building	City	Height (ft)	Stories
Empire State	New York	1250	102
sars Tower	Chicago	1454	110
Texas Commerce Tower	Houston	1002	75
Transamerica Pyramid	San Francisco	853	48

20. Given the following flight schedule, write a program to load this information into an array of structures, and ask the user to specify a flight number. Have the computer find the flight number and display the information corresponding to that flight. Account for the case where the user requests a nonexistent flight.

Flight #	Origin	Destination	Departure Time
117	Tucson	Dallas	8:45 a.m.
239	LA	Boston	10:15 a.m.
298	Albany	Reno	1:35 p.m.
326	Houston	New York	2:40 p.m.
445	New York	Tampa	4:20 p.m.

21. Table 7.3 contains the names and number of stores of the top 10 pizza chains in 2003. Write a program to place these data into an array of structures, compute the total number of stores for these 10 chains, and display a table giving the name and percentage of total stores for each of the companies.

Table 7.3. Top 10 pizza chains as of July 1,2003 (and numbers of stores).

Name	Stores	Name	Stores
1. Pizza Hut	7,523	6. Papa Murphy's	750
2. Domino's	4,904	7. Godfather's	566
3. Little Caesar's	2,850	8. Round Table	485
4. Papa John's	2,574	9. CiCi's Pizza	465
5. Sbarro	790	10. Chuck E. Cheese's	460
Source: Pizza Marketing Quarterly, Summer 2003			

22. A retail store has five bins, numbered 1 to 5, each containing a different commodity. At the beginning of a particular day, each bin contains 45 items. Table 7.4 shows the cost per item for each of the bins and the quantity sold during that day.

Table 7.4. Costs of items and quantities sold for Exercise 22.
(This item is displayed on page 356 in the print version)

Bin	Cost per Item	Quantity Sold
1	3.00	10
2	12.25	30
3	37.45	9
4	7.49	42
5	24.95	17

Write a program to

a. place the cost per item and the quantity sold from each bin into an array of structures;

[Page 356]

b. display a table giving the inventory at the end of the day and the amount of revenue obtained from each bin;

c. compute the total revenue for the day;

d. list the number of each bin that contains fewer than 20 items at the end of the day.

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 window. Second, the statements `Team.school = "Rice" and Team.mascot = "Owls"` are not valid. Team.school and Team.mascot 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
```

PREV PREV NEXT NEXT

[Page 356 (continued)]

7.4. Sorting and Searching

A sort is an algorithm for ordering an array. Of the many different techniques for sorting an array we discuss two, the bubble sort and the Shell sort. Both require the interchange of values stored in a pair of variables. If varl, var2, and temp are all variables of the same data type (such as all String), then the statements

```
temp = varl
varl = var2
var2 = temp
```

assign var1's value to var2 and var2's value to var1.

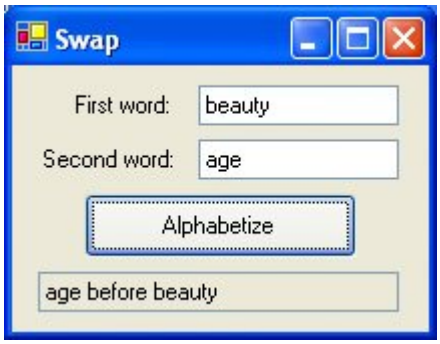
[Page 357]

Example 1.

The following program alphabetizes two words supplied in text boxes:

```
Private Sub btnAlphabetize_Click(...) Handles btnAlphabetize.Click
  'Alphabetize two words
  Dim firstWord, secondWord, temp As String
  firstWord = txtFirstWord.Text
  secondWord = txtSecondWord.Text
  If (firstWord > secondWord) Then   'Swap the two words
    temp = firstWord
    firstWord = secondWord
    secondWord = temp
  End If
  txtResult.Text = firstWord & " before " & secondWord
End Sub
```

[Run, type "beauty" and "age" into the text boxes, and click the button.]



Bubble Sort

The bubble sort is an algorithm that compares adjacent items and swaps those that are out of order. If this process is repeated enough times, the list will be ordered. Let's carry out this process on the list Pebbles, Barney, Wilma, Fred, and Dino. The steps for each pass through the list are as follows:

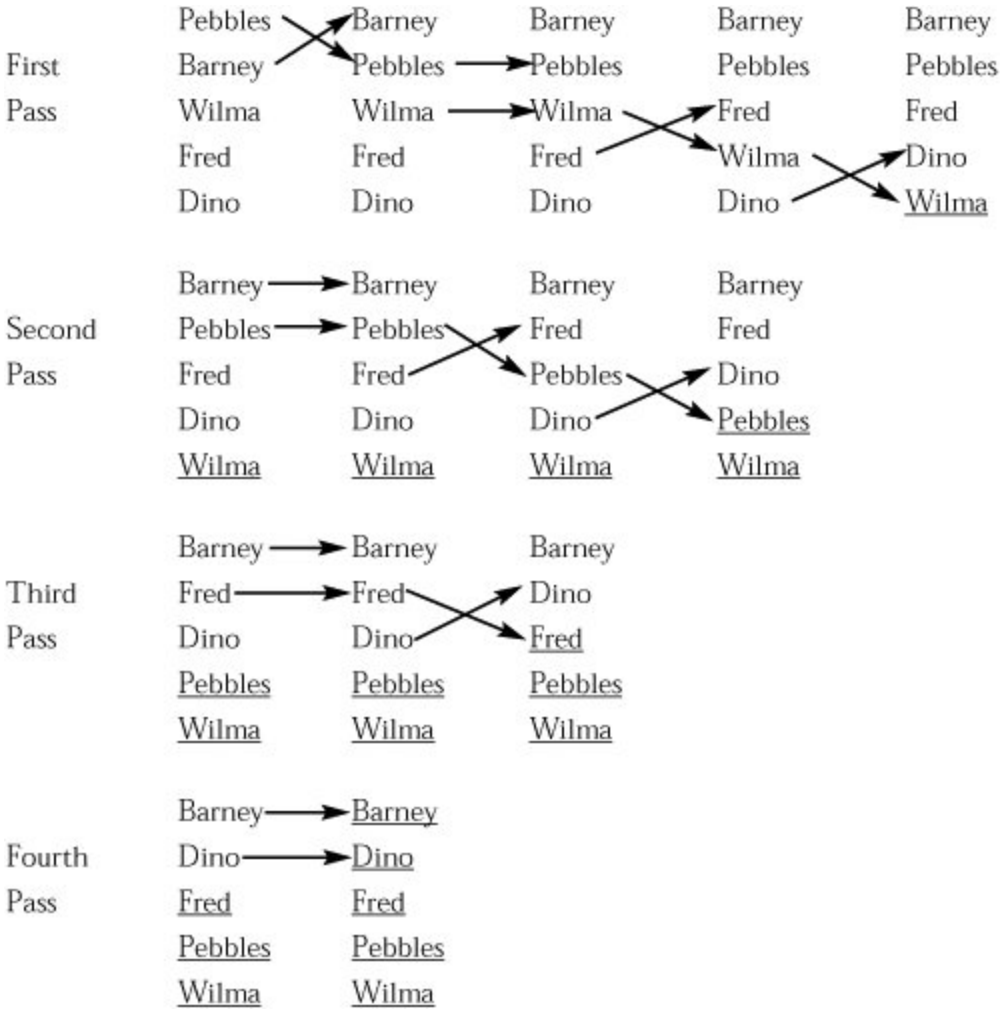
a. Compare the first and second items. If they are out of order, swap them.

b. Compare the second and third items. If they are out of order, swap them.

c. Repeat this pattern for all remaining pairs. The final comparison and possible swap are between the next-to-last and last items.

The first time through the list, this process is repeated to the end of the list. This is called the first pass. After the first pass, the last item (Wilma) will be in its proper position. Therefore, the second pass does not have to consider it and so requires one less comparison. At the end of the second pass, the last two items will be in their proper position. (The items that must have reached their proper position have been underlined.) Each successive pass requires one less comparison. After four passes, the last four items will be in their proper positions, and hence, the first will be also.

[Page 358]



Example 2.

(This item is displayed on pages 358 - 359 in the print version)

The following program alphabetizes the names Pebbles, Barney, Wilma, Fred, and Dino. Sorting the list requires a pair of nested loops. The inner loop performs a single pass, and the outer loop controls the number of passes.

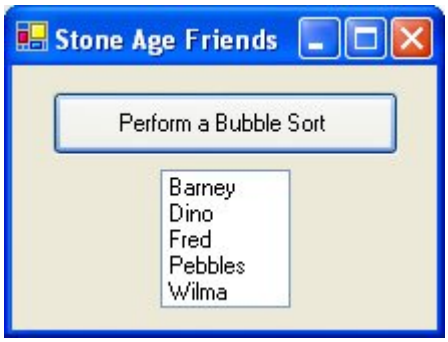
```
Dim person() As String = {"Pebbles", "Barney", "Wilma", "Fred", "Dino"}

Private Sub frmFlintstones_Load(...) Handles MyBase.Load
  'Display unsorted list
  For i As Integer = 0 To 4
    lstPeople.Items.Add(person(i))
  Next
End Sub

Private Sub btnSort_Click(...) Handles btnSort.Click
  'Bubble sort names
  Dim temp As String
```

[Page 359]

```
  For passNum As Integer = 1 To 4 'Number of passes is 1 less than
  'number of items
```

```
    For i As Integer = 1 To 5 - passNum 'Each pass needs 1 less comparison
      If (person(i - 1) > person(i)) Then
        temp = person(i - 1)
        person(i - 1) = person(i)
        person(i) = temp
      End If
    Next
  Next
  'Display alphabetized list
  lstPeople.Items.Clear()
  For i As Integer = 0 To 4
    lstPeople.Items.Add(person(i))
  Next
End Sub
```

[Run, and click the button.]



Example 3.

(This item is displayed on pages 359 - 361 in the print version)

Table 7.5 contains facts about the 10 most populous states with listings in ascending order by state abbreviation. The following program sorts the table in descending order by population. Data are read from a file into an array of structures by the form's Load event procedure. When btnDisplayStats is clicked, the array is sorted based on the population field.

Table 7.5. The 10 most populous states.

State	Per Capita Income	% College Graduates	Population in Millions
CA	$32,678	27.5	33.9
FL	$28,493	22.8	16.0
GA	$28,438	23.1	8.2
IL	$32,755	27.1	12.4
MI	$29,538	23.0	9.9
NJ	$38,153	30.1	8.4

NY	$35,884	28.7	19.0
OH	$28,619	24.6	11.4
PA	$30,617	24.3	12.3
TX	$28,486	23.9	20.9

Note: Column 3 gives the percentage of residents age 25 or older with a college degree.

[Page 360]

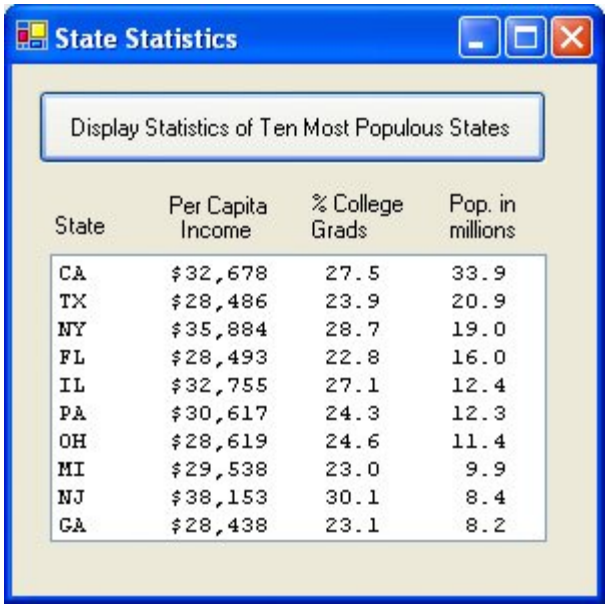
```
Structure District
  Dim name As String
  Dim income As Double
  Dim collegeGrad As Double
  Dim population As Double
End Structure

Dim state(9) As District

Private Sub frmStates_Load(...) Handles MyBase.Load
  'Assume the data for state name, income, % college grads,
  'and population in millions are in the file "STATES.TXT"
  'First four lines of file contain the data CA, 32678, 27.5, 33.9
  Dim sr As IO.StreamReader = IO.File.OpenText("STATES.TXT")
  For i As Integer = 0 To 9
     state(i).name = sr.ReadLine
     state(i).income = CDbl(sr.ReadLine)
     state(i).collegeGrad = CDbl(sr.ReadLine)
     state(i).population = CDbl(sr.ReadLine)
  Next
  sr.Close()
End Sub

Private Sub btnDisplayStats_Click(...) Handles btnDisplayStats.Click
  SortData()
  ShowData()
End Sub

Sub ShowData()
  'Display ordered table
  Dim fmtStr As String = "{0,-4}{1,11:C0}{2,8:N1}{3,9:N1}"
  lstStats.Items.Clear()
  For i As Integer = 0 To 9
    lstStats.Items.Add(String.Format(fmtStr, state(i).name, _
      state(i).income, state(i).collegeGrad, state(i).population))
  Next
End Sub

Sub SortData()
  'Bubble sort table in descending order by population
  For passNum As Integer = 1 To 9
   For index As Integer = 1 To 10 - passNum
    If (state(index - 1).population < state(index).population) Then
     SwapData(index)
```

```
    End If
   Next
  Next
End Sub
```

[Page 361]

```
Sub SwapData(ByVal index As Integer)
  'Swap entries
  Dim temp As District
  temp = state(index - 1)
  state(index - 1) = state(index)
  state(index) = temp
End Sub
```

[Run, and click the button.]



Shell Sort

The bubble sort is easy to understand and program. However, it is too slow for really long lists. The Shell sort, named for its inventor, Donald L. Shell, is much more efficient in such cases. It compares distant items first and works its way down to nearby items. The interval separating the compared items is called the gap. The gap begins at one-half the length of the list and is successively halved until eventually each item is compared with its neighbor as in the bubble sort. The algorithm for a list of n + 1 items referred to as item 0 through item I is as follows.

a. Begin with a gap of g = Int([n + 14]/2).

b. Compare items 0 and g, items 1 and 2 +g, 2 and 2+g,..., n-g and n. Swap any pairs that are out of order.

c. Repeat Step 2 until no swaps are made for gap g.

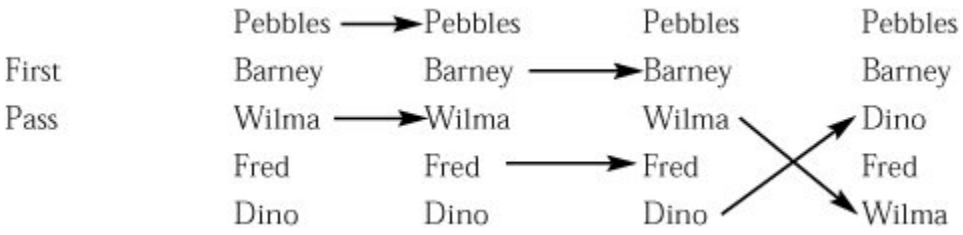
d. Halve the value of g.

e. Repeat Steps 2, 3, and 4 until the value of g is 0.

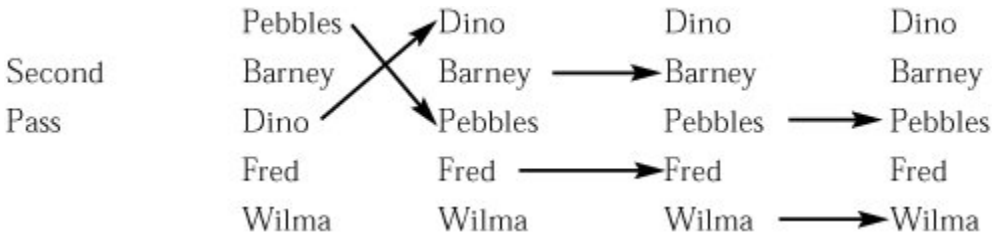
The Shell sort is illustrated in what follows, in which crossing arrows indicate that a swap occurred:

Initial Gap = Int([Number of Items]/2) = Int(5/2) = 2

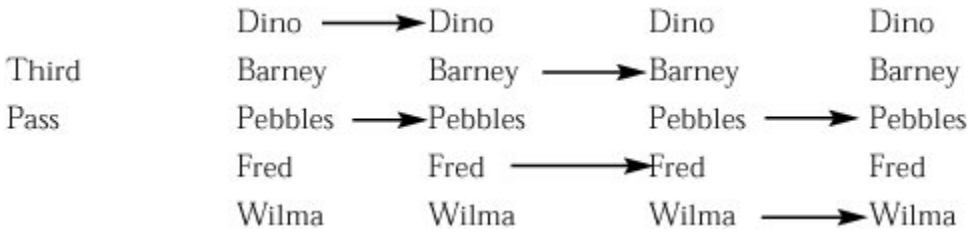
[Page 362]



Because there was a swap, use the same gap for the second pass.

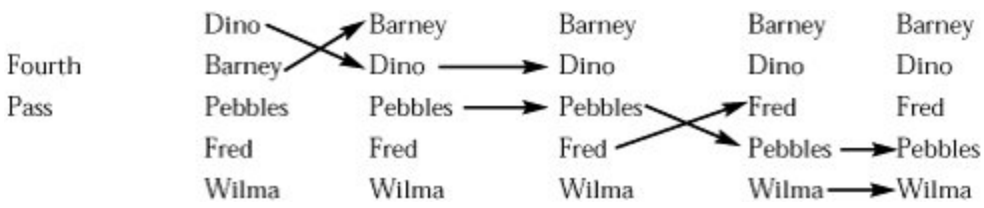


Again, there was a swap, so keep the current gap.

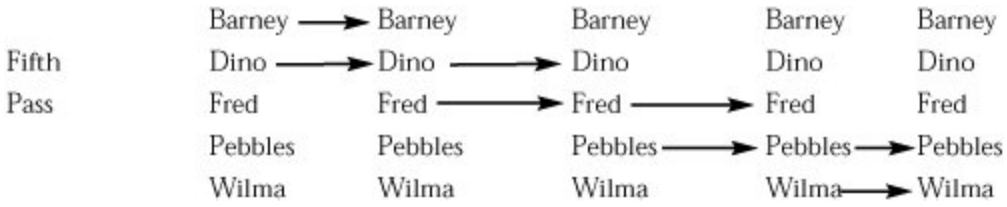


There were no swaps for the current gap of 2, so

Next Gap = Int([Previous Gap]/2) = Int(2/2) = 1.



Because there was a swap (actually two swaps), keep the same gap.



Because there were no swaps for the current gap, then

Next Gap = Int([Previous Gap]/2) = Int(1/2) = 0,

and the Shell sort is complete.

[Page 363]

Notice that the Shell sort required 17 comparisons to sort the list, whereas the bubble sort required only 10 comparisons for the same list. This illustrates the fact that for very short lists, the bubble sort is preferable; however, for lists of 30 items or more, the Shell sort will consistently outperform the bubble sort. Table 7.6 shows the average number of comparisons required to sort arrays of varying sizes.

Table 7.6. Efficiency of bubble and Shell sorts.

Array Elements	Bubble Sort Comparisons	Shell Sort Comparisons
5	10	15
10	45	57
15	105	115
20	190	192
25	300	302
30	435	364
50	1225	926
100	4950	2638
500	124,750	22,517
1000	499,500	58,460

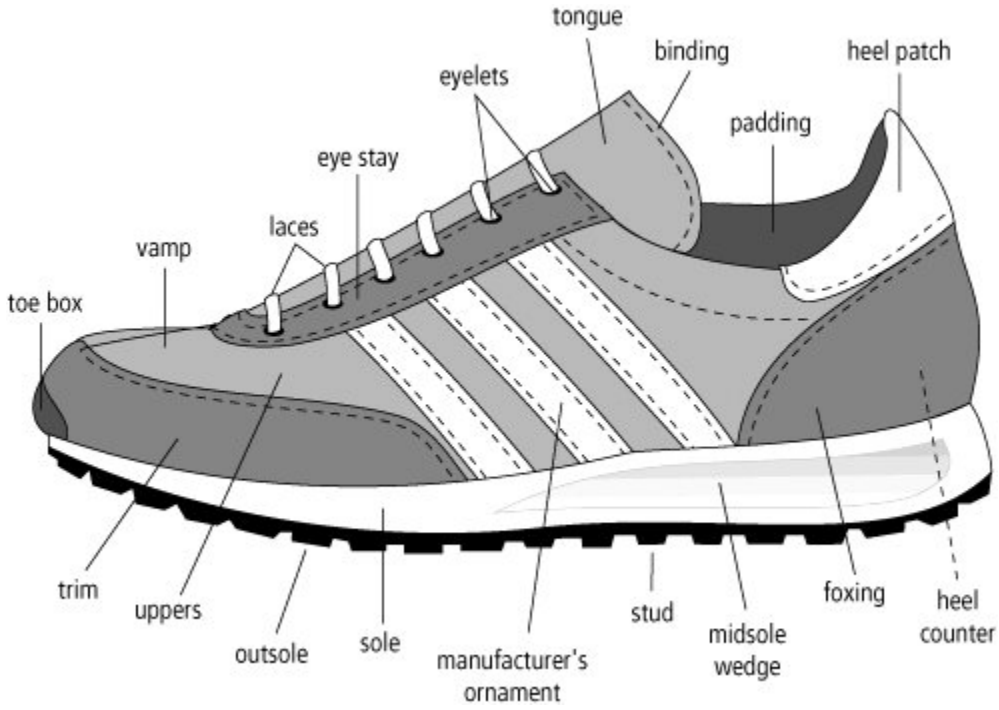
Example 4.

(This item is displayed on pages 363 - 365 in the print version)

The following program uses the Shell sort to alphabetize the parts of a running shoe. (See Figure 7.5.) The data are read into an array whose size is large enough to guarantee sufficient space. In the form's Load event procedure, the variable numParts provides the subscripts for the array and serves as a counter. The final value of numParts is available to all procedures because the variable was created in the Declarations section of the Code

window. The Sub procedure SortData uses a flag to indicate if a swap has been made during a pass.

Figure 7.5. Running shoe.



[Page 364]

```
Dim part(50) AsString
Dim numParts As Integer

Private Sub frmShoe_Load(...) HandlesMyBase.Load
  'Read names of parts
  numParts = 0      'Number of parts
  Dim sr As IO.StreamReader = IO.File.OpenText("SHOEPART.TXT")
  Do While (sr.Peek <> -1) And (numParts < part.GetUpperBound(0))
    part(numParts) = sr.ReadLine
    numParts += 1
  Loop
  sr.Close()
End Sub

Private Sub btnDisplay_Click(...) Handles btnDisplay.Click
  'Sort and display parts of running shoe
  SortData()
  ShowData()
End Sub

Sub SortData()
  'Shell sort shoe parts
  Dim gap As Integer, doneFlag As Boolean
  Dim temp As String
  gap = CInt(Int(numParts / 2))
  Do While gap >= 1
```

```
    Do
      doneFlag = True
      For index As Integer = 0 To numParts - 1 - gap
        If part(index) > part(index + gap) Then
          temp = part(index)
          part(index) = part(index + gap)
          part(index + gap) = temp
          doneFlag = False
        End If
      Next
    Loop Until doneFlag = True    'Also can be Loop UntildoneFlag
    gap = CInt(Int(gap / 2))      'Halve the length of the gap
  Loop
End Sub

Sub ShowData()
  'Display sorted list of parts
  lstParts.Items.Clear()
  For i As Integer = 0 To numParts - 1
    lstParts.Items.Add(part(i))
  Next
End Sub
```

[Page 365]

[Run, and click the button.]

Searching

Suppose we had an array of 1000 names in alphabetical order and wanted to locate a specific person in the list. One approach would be to start with the first name and consider each name until a match was found. This process is called a sequential search. We would find a person whose name begins with "A" rather quickly, but 1000 comparisons might be necessary to find a person whose name begins with "Z." For much longer lists, searching could be a time-consuming matter. However, there is a method, called a binary search, that shortens the task considerably.

Let us refer to the sought item as quarry. The binary search looks for quarry by determining in which half of the list it lies. The other half is then discarded, and the retained half is temporarily regarded as the entire list. The process is repeated until the item is found. A flag can indicate if quarry has been found.

The algorithm for a binary search of an ascending list is as follows (Figure 7.6 shows the flowchart for a binary search):

1. At each stage, denote the subscript of the first item in the retained list by first and the subscript of the last item by last. Initially, the value of first is 0, the value of last is one less than the number of items in the list, and the value of flag is False.

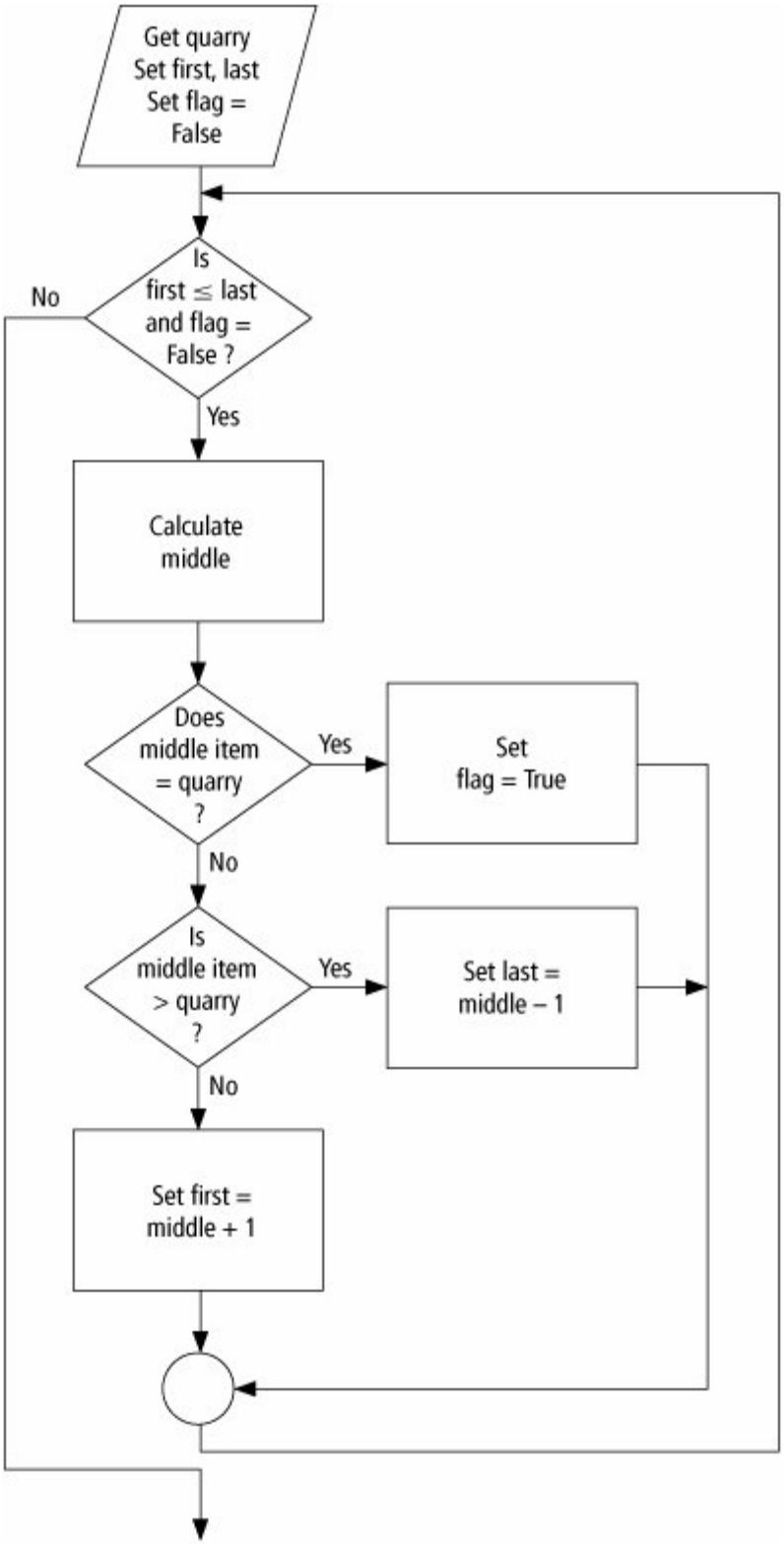
2. Look at the middle item of the current list, the item having the subscript middle = Int((first + last)/2).

3. If the middle item is quarry, then flag is set to True and the search is over.

4. If the middle item is greater than quarry, then quarry should be in the first half of the list. So the subscript of quarry must lie between first and middle -1. That is, the new value of last is middle - 1.

[Page 366]

Figure 7.6. Flowchart for a binary search.



5. If the middle item is less than quarry, then quarry should be in the second half of the list of possible items. So the subscript of quarry must lie between middle +1 and last. That is, the new value of first is middle +1.

[Page 367]

6. Repeat Steps 2 through 5 until quarry is found or until the halving process uses up the entire list. (When the entire list has been used up, first > last.) In the second case, quarry was not in the original list.

Example 5.

(This item is displayed on pages 367 - 368 in the print version)

In the following program the array firm() contains the alphabetized names of up to 100 corporations. The program requests the name of a corporation as input and uses a binary search to determine whether the corporation is in the array.

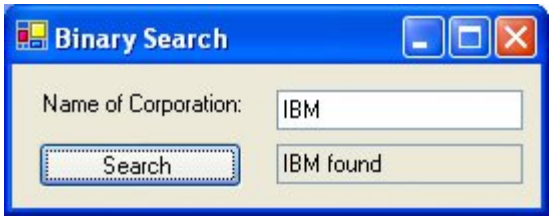
```
Dim firm(99) As String
Dim numFirms As Integer

Private Sub frmCompanies_Load(...) Handles MyBase .Load
  'Fill array with data from FIRMS.TXT, which contains
  'the names of up to 100 companies
  Dim sr As IO.StreamReader = IO.File.OpenText("FIRMS.TXT")
  numFirms = 0
  Do While (sr.Peek <> -1) And (numFirms < firm.GetUpperBound(0))
    firm(numFirms) = sr.ReadLine
    numFirms += 1
  Loop
  sr.Close()
End Sub

Private Sub btnSearch_Click(...) Handles btnSearch.Click
  Dim corp, result As String
  corp = txtCorporation.Text.ToUpper
  result = ""
  BinarySearch(corp, result)
  'Display results of search
  txtResult.Text = corp & " " & result
End Sub

Sub BinarySearch(ByVal corp As String, ByRef result As String)
  'Array firm() assumed already ordered alphabetically
  'Binary search of firm() for corp
  Dim first, middle, last As Integer
  Dim foundFlag As Boolean = False
  first = 0
  last = numFirms - 1
  Do While (first <= last) And (Not foundFlag)
    middle = CInt(Int((first + last) / 2))
    Select Case firm(middle).ToUpper
      Case corp
        foundFlag = True
```

[Page 368]

```
      Case Is > corp
        last = middle - 1
      Case Is < corp
        first = middle + 1
    End Select
  Loop
  If foundFlag Then
    result = "found"
```

```
  Else
    result = "not found"
  End If
End Sub
```

[Run, type "IBM" into the text box, and click the button.]



Suppose the array contains 100 corporations and the corporation input in Example 5 is in the second half of the array. On the first pass, middle would be assigned Int((0 + 99)/2) = Int(49.5) = 49 and then first would be altered to 49 + 1 = 50. On the second pass, middle would be assigned Int((50 + 99)/2) = Int (74.5) = 74. If the corporation is not the array element with subscript 74, then either last would be assigned 73 or first would be assigned 75, depending on whether the corporation appears before or after the 74th element. Each pass through the loop halves the range of subscripts containing the corporation until the corporation is located.

In Example 5, the binary search merely reported whether an array contained a certain item. After finding the item, its array subscript was not needed. However, if the search had been carried out on an array of structures (as in Table 7.5), the subscript of the found item could be used to retrieve the related information for the other members. This process, called a table lookup, is used in the following example.

Example 6.

(This item is displayed on pages 368 - 370 in the print version)

The following program uses a binary search procedure to locate the data for a state from Example 3 requested by the user. The program does not include a sort of the text file STATES.TXT, because the file is already ordered alphabetically by city name.

```
Structure District
  Dim name As String
  Dim income As Double
  Dim collegeGrad As Double
  Dim population As Double
End Structure
```

[Page 369]

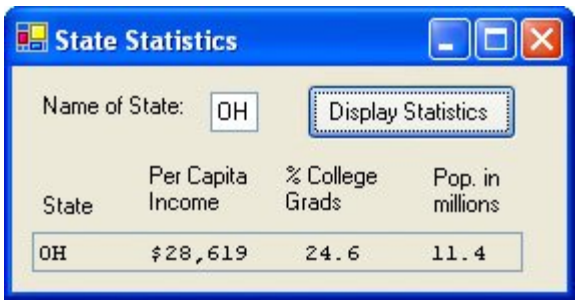
```
Dim state(9) As District
Private Sub frmStates_Load(...) Handles MyBase.Load
  'Assume that the data for state name, per capita income,
  '% college grads, and population in millions
  'have been placed in the file "STATES.TXT"
  'First four lines of file contain the data CA 32678,27.5,33.9
  Dim sr As IO.StreamReader = IO.File.OpenText("STATES.TXT")
  For i As Integer = 0 To 9
    state(i).name = sr.ReadLine
    state(i).income = CDbl(sr.ReadLine)
```

```
    state(i).collegeGrad = CDbl(sr.ReadLine)
    state(i).population = CDbl(sr.ReadLine)
  Next
  sr.Close()
End Sub
Private Sub btnDisplayStats_Click(...) Handles btnDisplayStats.Click
  'Search for state in the populous states table
  Dim searchState As String, result As Integer
  searchState = mtxtState.Text.ToUpper
  result = FindState(searchState)
  If result >= 0 Then
    ShowData(result)
  Else
    MsgBox(searchState & " not in file", 0, "Not Found")
  End If
End Sub
Function FindState(ByVal searchState As String) As Integer
  'Binary search table for state name
  Dim first, middle, last As Integer
  first = 0
  last = 9
  Do While (first <= last)
    middle = CInt(Int((first + last) / 2))
    Select Case city(middle).name.ToUpper
      Case searchState
        Return middle
      Case Is > searchState
        last = middle - 1
      Case Is < searchState
        first = middle + 1
    End Select
  Loop
  Return -1
End Function
```

[Page 370]

```
Sub ShowData(ByVal result As Integer)
  'Display the data for the specified state
  Dim fmtStr As String = "{0,-4}{1,11:C0}{2,8:N1}{3,9:N1}"
  txtStats.text = String.Format(fmtStr, state(result).name, _
                  state(result).income, state(result).collegeGrad, _
                  state(result).population.)
End Sub
```

[Run, type "OH" into the masked text box, and click the button.]



Comments

1. Suppose that our bubble sort algorithm is applied to an ordered list. The algorithm will still make n - 1 passes through the list. The process could be shortened for some lists by flagging the presence of out-of-order items as in the Shell sort.

2. In Example 3, an array of structures already ordered by one member was sorted by another member. Usually, an array of structures is sorted by the member to be searched when accessing the member. This member is called the key member.

3. Suppose that an array of 2000 items is searched sequentiallythat is, one item after anotherin order to locate a specific item. The number of comparisons would vary from 1 to 2000, with an average of 1000. With a binary search, the number of comparisons would be at most 11, because 2^{11} >2000.

4. The method ToUpper converts all the characters in a string to uppercase. ToUpper is useful in sorting and searching arrays of strings when the alphabetic case (upper or lower) is unimportant. For instance, Example 5 includes ToUpper in the Select Case comparisons, and so the binary search will locate "Mobil" in the array even if the user entered "MOBIL".

5. Visual Basic has a built-in sort routine that can be used to sort arrays. The statement

```
Array.Sort(arrayName)
```

sorts arrayName into ascending order. For instance, in Example 2 the event procedure can be replaced with the following code.

```
Private Sub btnSort_Click(...) Handles btnSort.Click
  Dim name() As String = {"Pebbles", "Barney", "Wilma", _
                          "Fred", "Dino"}
  Array.Sort(name)
  'Display alphabetized list
  lstNames.Items.Clear()
```

```
  For i As Integer = 0 To 4
    lstNames.Items.Add(name(i))
  Next
End Sub
```

Array.Sort cannot be used with arrays of structures and therefore is not adequate for our purposes.

Practice Problems 7.4

1. The pseudocode for a bubble sort of an array of n items follows. Why is the terminating value of the outer loop n-1 and the terminating value of the inner loop n-j?

```
For j As Integer = 1 To n - 1
  For k As Integer = 1 To n - j
    If [(k-1)st and kth items are out of order] Then [swap them]
```

```
  Next
Next
```

2. Complete the table that follows by filling in the values of each variable after successive passes of a binary search of an array of 20 items, where the sought item is in the position 13.

First	Last	Middle
0	19	9
10	19	

Exercises 7.4

In Exercises 1 through 4, determine the output displayed in the list box when the button is clicked.

1.
```
Private Sub btnDisplay_Click(...) Handles btnDisplay.Click
  Dim p As Integer = 100
  Dim q As Integer = 200
  Dim temp As Integer = p
  p = q
  q = temp
  lstOutput.Items.Add(p)
  lstOutput.Items.Add(q)
End Sub
```

2.
```
Dim gag() As String = {"Stan"," Oliver"}

Private Sub btnDisplay_Click(...) Handles btnDisplay.Click
 Dim temp As String
  If gag(1) < gag(0) Then
   temp = gag(1)
```

[Page 372]

```
   gag(1) = gag(0)
   gag(0) = temp
  End If
 lstOutput.Items.Add(gag(0))
 lstOutput.Items.Add(gag(1))
End Sub
```

3.
```
Private Sub btnDisplay_Click(...) Handles btnDisplay.Click
 Dim x, y, temp As Double
```

```
 Dim swappedFlag As Boolean = False
 x = 7
 y = 11
 If y > x Then
   temp = x
   x = y
   y = temp
   swappedFlag = True
 End If
 lstOutput.Items.Add(x)
 lstOutput.Items.Add(y)
 If swappedFlag Then
   lstOutput.Items.Add("Numbers interchanged.")
 End If
End Sub
```

4.

```
Dim a(2) As Integer

Private Sub btnDisplay_Click(...) Handles btnDisplay.Click
 Dim temp As Integer
 For j As Integer = 0 To 1
  For k As Integer = 0 To 2 - j
   If a(k) > a(k + 1) Then
    temp = a(k)
    a(k) = a(k + 1)
    a(k + 1) = temp
   End If
  Next
 Next
 For j As Integer = 0 To 2
   lstOutput.Items.Add(a(j))
 Next
End Sub
Private Sub frmNumbers_Load(...) Handles MyBase.Load
   Dim sr As IO.StreamReader = IO.File.OpenText("DATA.TXT")
   For j As Integer = 0 To 2
     a(j) = CInt(sr.ReadLine)
   Next
   sr.Close()
End Sub
```

(Assume that the three lines of the file DATA.TXT contain the following entries: 7, 4, 3.)

[Page 373]

In Exercises 5 and 6, identify the errors.

5.

```
Dim c(3), d(3) As Integer
```

```
Private Sub btnDisplay_Click(...) Handles btnDisplay.Click
  'Swap two items
  c(3) = d(3)
  d(3) = c(3)
  lstOutput.Items.Add(c(3))
  lstOutput.Items.Add(d(3))
End Sub

Private Sub frmNumbers_Load(...) Handles MyBase.Load
  Dim sr As IO.StreamReader = IO.File.OpenText("DATA.TXT")
  For i As Integer = 0 To 3
    c(i) = CInt(sr.ReadLine)
    d(i) = CInt(sr.ReadLine)
  Next
  sr.Close()
End Sub
```

(Assume that the eight lines of the file DATA.TXT contain the following entries:1, 2, 3, 4, 5, 6, 7, 8.)

6.

```
Dim a(2), b(2) As Integer

Private Sub btnDisplay_Click(...) Handles btnDisplay.Click
  'Swap the two arrays
  Dim temp(2) As Integer
  a = b
  b = a
End Sub

Private Sub frmNumbers_Load(...) Handles MyBase.Load
  Dim sr As IO.StreamReader = IO.File.OpenText("DATA.TXT")
  For i As Integer = 0 To 2
    a(i) = CInt(sr.ReadLine)
    b(i) = CInt(sr.ReadLine)
  Next
  sr.Close()
End Sub
```

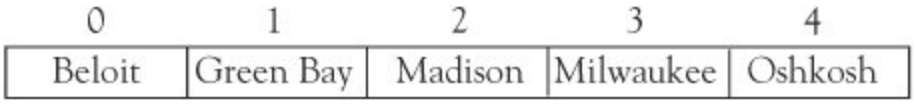
(Assume that the six lines of the file DATA.TXT contain the following entries:1, 3, 5, 7, 9, 11.)

7. Which type of search would be best for the following array?

0	1	2	3	4
Paul	Ringo	John	George	Pete

[Page 374]

8. Which type of search would be best for the following array?



9. Consider the items Tin Man, Dorothy, Scarecrow, and Lion in that order. After how many swaps in a bubble sort will the list be in alphabetical order?

10. How many comparisons will be made in a bubble sort of six items?

11. How many comparisons will be made in a bubble sort of n items?

12. Modify the program in Example 2 so that it will keep track of the number of swaps and comparisons and display these numbers before ending.

13. Rework Exercise 9 using the Shell sort.

14. How many comparisons would be made in a Shell sort of six items if the items were originally in descending order and were sorted in ascending order?

15. If a list of six items is already in the proper order, how many comparisons will be made by a Shell sort?

16. Suppose a list of 5000 numbers is to be sorted, but the numbers consist of only 1, 2, 3, and 4. Describe a method of sorting the list that would be much faster than either the bubble or Shell sort.

17. What is the maximum number of comparisons required to find an item in a sequential search of 16 items? What is the average number of comparisons? What is the maximum number of comparisons required to find an item in a binary search of 16 items?

18. Redo Exercise 17 with 2^n items, where is any positive integer.

In Exercises 19 through 26, write a program (or procedure) to complete the stated task.

19. Exchange the values of the variables x, y, and z so that x has y's value, y has z's value, and z has x's value.

20. Display the names of the seven dwarfs in alphabetical order. For the contents of a text file use Doc, Grumpy, Sleepy, Happy, Bashful, Sneezy, and Dopey.

21. The nation's capital has long been a popular staging area for political, religious, and other large public rallies, protest marches, and demonstrations. The events in Table 7.7 have drawn the largest crowds, according to estimates from D.C., U.S. Park, or Capitol police. Read the data into an array of structures and display a similar table with the event names in alphabetical order. Note: The data is contained in the file EVENTS.TXT.

22. Table 7.8 presents statistics on the five leading sneaker brands. Read the data into an array of structures, and display a similar table with wholesale revenue in descending order.

23. Accept 10 words to be input in alphabetical order, and store them in an array. Then accept an 11th word as input, and store it in the array in its correct alphabetical position.

[Page 375]

Table 7.7. Large Public Displays of Emotion in Washington, D.C.

Event	Crowd Estimate (in thousands)
Lyndon Johnson inauguration (1/23/65)	1,200
Bicentennial fireworks (7/4/76)	1,000
Desert Storm rally (6/8/91)	800
Bill Clinton inauguration (l/20/93)	800
Beach Boys concert (7/4/85)	625
Washington Redskins victory parade (2/3/88)	600
Vietnam moratorium rally (11/15/69)	600
Ronald Reagan inauguration (1/20/81)	500
U.S. Iran hostage motorcade (l/28/81)	500

Table 7.8. 2004 U.S.market share in sneakers.

Brand	Wholesale Revenue (in millions)	Percentage Share of U.S. Market
Adidas USA	792	8.9
Fila	392	4.4
New Balance	1024	11.5
Nike	3231	36.3
Reebok	1086	12.1
Source: Morgan Stanley Dean Witter Research.		

24. An airline has a list of 200 flight numbers (between 1 and 1000) in ascending order in the file FLIGHTS.TXT. Accept a number as input and do a binary search of the list to determine if the flight number is valid.

25. Allow a number n to be input by the user. Then accept as input a list of numbers. Place the numbers into an array and apply a bubble sort.

26. Rework Exercise 25, but use a flag to detect the presence of out-of-order items as in the Shell sort.

27. Write a program that accepts a word as input and converts it into Morse code. The dots and dashes corresponding to each letter of the alphabet are as follows:

A .	H	O	V . . .
B . . .	I . .	P . .	W .
C ..	J .	Q .	X . .
D . .	K .	R . .	Y .
E .	L . . .	S . . .	Z . .
F . . .	M	T	
G .	N .	U . .	

[Page 376]

28. Write a program that accepts an American word as input and performs a binary search to translate it into its British equivalent. Use the following list of words for data, and account for the case when the word requested is not in the list:

American	British	American	British
attic	loft	ice cream	ice
business suit	lounge suit	megaphone	loud hailer
elevator	lift	radio	wireless
flashlight	torch	sneakers	plimsolls
french fries	chips	truck	lorry
gasoline	petrol	zero	nought

29. Write a program that accepts a student's name and seven test scores as input and calculates the average score after dropping the two lowest scores.

30. Suppose letter grades are assigned as follows:

97 and above	A+	7476	C
9496	A	7073	C-

9093	A	6769	D+
8789	B+	6466	D
8486	B	6063	D
8083	B	059	F
7779	C+		

Write a program that accepts a grade as input and displays the corresponding letter.

Hint: This problem shows that when you search an array, you don't always look for equality. Set up an array of structures with the member range containing the values 97, 94, 90, 87, 84, ..., 0 and the member letter containing A+, A, A , B+,...,F. gNext, perform a sequential search to find the first structure such that the value of the range member is less than or equal to the input grade.

31. The median of a set of n measurements is a number such that half the n measurements fall below the median, and half fall above. If the number of measurements n is odd, the median is the middle number when the measurements are arranged in ascending or descending order. If the number of measurements n is even, the median is the average of the two middle measurements when the measurements are arranged in ascending or descending order. Write a program that requests a number n and a set of n measurements as input and then displays the median.

32. Write a program with two buttons labeled Ascending Order and Descending Order that displays the eight vegetables in V8® in either ascending or descending alphabetic order. The vegetables (tomato, carrot, celery, beet, parsley, lettuce, watercress, and spinach) should be stored in a class-level array.

Solutions to Practice Problems 7.4

1. The outer loop controls the number of passes, one less than the number of items in the list. The inner loop performs a single pass, and the jth pass consists of n j comparisons.

[Page 377]

2.

First	Last	Middle
0	19	9
10	19	14
10	13	11
12	13	12

13 13 13

PREV PREV NEXT NEXT

[Page 377 (continued)]

7.5. Two-Dimensional Arrays

Each array discussed so far held a single list of items. Such array variables are called one-dimensional arrays or single-subscripted variables. An array can also hold the contents of a table with several rows and columns. Such arrays are called two-dimensional arrays or double-subscripted variables. Two tables follow. Table 7.9 gives the road mileage between certain cities. It has four rows and four columns. Table 7.10 shows the leading universities in three disciplines. It has three rows and five columns.

Two-dimensional array variables store the contents of tables. They have the same types of names as other array variables. The only difference is that they have two subscripts, each with its own upper bound. The first subscript is determined by the number of rows in the table, and the second subscript is determined by the number of columns. The statement

```
Dim arrayName(m, n) As varType
```

declares an array of type varType corresponding to a table with rows labeled from 0 to m and columns labeled from 0 to n. The entry in the jth row, kth column is arrayName(j, k). For instance, the data in Table 7.9 can be stored in an array named rm(). The statement

```
Dim rm(3, 3) As Double
```

[Page 378]

will declare the array. Each element of the array has the form rm(row, column). The values of the elements of the array we use will be

rm(0,0)= 0	rm(0,1)= 2054	rm(0,2)= 802	rm(0,3)= 738
rm(1,0)= 2054	rm(1,1)= 0	rm(1,2)= 2786	rm(1,3)= 2706
rm(2,0)= 802	rm(2,1)= 2786	rm(2,2)= 0	rm(2,3)= 100
rm(3,0)= 738	rm(3,1)= 2706	rm(3,2)= 100	rm(3,3)= 0

Table 7.9. Road mileage between selected U.S. cities.
(This item is displayed on page 377 in the print version)

	Chicago	Los Angeles	New York	Philadelphia
Chicago	0	2054	802	738
Los Angeles	2054	0	2786	2706
New York	802	2786	0	100
Philadelphia	738	2706	100	0

Table 7.10. University rankings.
(This item is displayed on page 377 in the print version)

	1	2	3	4	5
Business	U of PA	U of IN	U of MI	UC Berk	U of VA
Comp Sci.	MIT Cng-Mellon	UC Berk	Cornell	U of IL	
Engr/Gen.	U of IL	U of OK	U of MD	Cng-Mellon	CO Sch. of Mines
Source: A Rating of Undergraduate Programs in American and International Universities, Dr. Jack Gourman, 1998.					

The data in Table 7.10 can be stored in a two-dimensional string array named univ(). The appropriate array is declared with the statement

```
Dim univ(2, 4) As String
```

Some of the entries of the array are

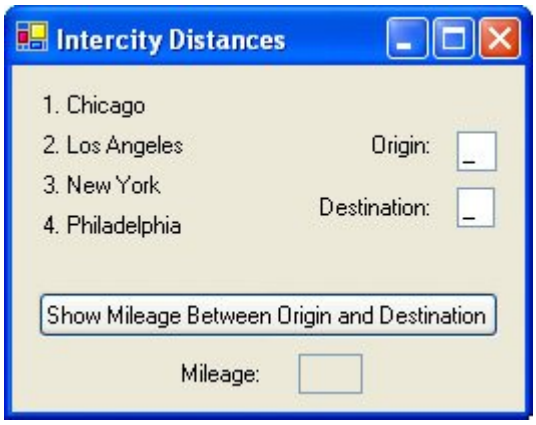
univ(0, 0) = "U of PA"

univ(1, 2) = "UC Berk"

univ(2, 4) = "CO Sch. of Mines"

Example 1.
(This item is displayed on pages 378 - 379 in the print version)

The following program stores and accesses the data from Table 7.9. Data are read from the text file DISTANCE.TXT into a two-dimensional class-level array using a pair of nested loops. The outer loop controls the rows, and the inner loop controls the columns.



Object	Property	Setting
frmDistances	Text	Intercity Distances
lblCh	Text	1.Chicago
lblLA	Text	2.Los Angeles
lblNY	Text	3.New York
lblPh	Text	4.Philadelphia
lblOrig	Text	Origin:
mtxtOrig	Mask	0
lblDest	Text	Destination:
mtxtDest	Mask	0
btnShow	Text	Show Mileage Between Origin and Destination
lblMiles	Text	Mileage:
txtMiles	ReadOnly	True

```
Dim rm(3, 3) As Double

Private Sub frmDistances_Load(...) Handles MyBase.Load
  'Fill two-dimensional array with intercity mileages
  'Assume the data has been placed in the file "DISTANCE.TXT"
  '(First four lines of the file contain the numbers 0, 2054, 802, 738)
  Dim sr As IO.StreamReader = IO.File.OpenText("DISTANCE.TXT")
```

[Page 379]

```
  For row As Integer = 0 To 3
    For col As Integer = 0 To 3
      rm(row, col) = CDbl(sr.ReadLine)
    Next
  Next
  sr.Close()
```

```
End Sub
Private Sub btnShow_Click(...) Handles btnShow.Click
  'Determine road mileage between cities
  Dim row, col As Integer
  row = CInt(mtxtOrig.Text)
  col = CInt(mtxtDest.Text)
  If (row >= 1 And row <= 4) And (col >= 1 And col <= 4) Then
    txtMiles.Text = CStr(rm(row - 1, col - 1))
  Else
    MsgBox("Origin and Dest. must be numbers from 1 to 4", 0, "Error")
  End If
End Sub
```

[Run, type 3 into the Origin box, type 1 into the Destination box, and click the button.]

An unsized two-dimensional array can be declared with a statement of the form

```
Dim arrayName(,) As varType
```

and a two-dimensional array can be declared and initialized at the same time with a statement of the form

```
Dim arrayName(,) As varType = {{ROW0}, {ROW1},... {ROWm}}
```

where ROW0 consists of the entries in the top row of the corresponding table separated by commas, ROW1 consists of the entries in the next row of the corresponding table separated by commas, and so on. For instance, in Example 1, the declaration statement and frmDistances_Load event procedure can be replaced by the single statement

```
Dim rm(,) As Double = {{0, 2054, 802, 738}, _
            {2054, 0, 2786, 2706}, {802, 2786, 0, 100}, _
            {738, 2706, 100, 0}}
```

An already-created array can be resized with

```
ReDim arrayName(r, s)
```

which loses the current contents, or with

```
ReDim Preserve arrayName(r, s)
```

which keeps the current values. However, when the keyword Preserve is used, only the last coordinate can be resized. The upper bound of the first coordinate of the array is given by `arrayName.GetUpperBound(0)`, and the upper bound of the second coordinate is given by `arrayName.GetUpperBound(1)`.

So far, two-dimensional arrays have been used only to store data for convenient lookup. In the next example, an array is used to make a valuable computation.

Example 2.

(This item is displayed on pages 380 - 383 in the print version)

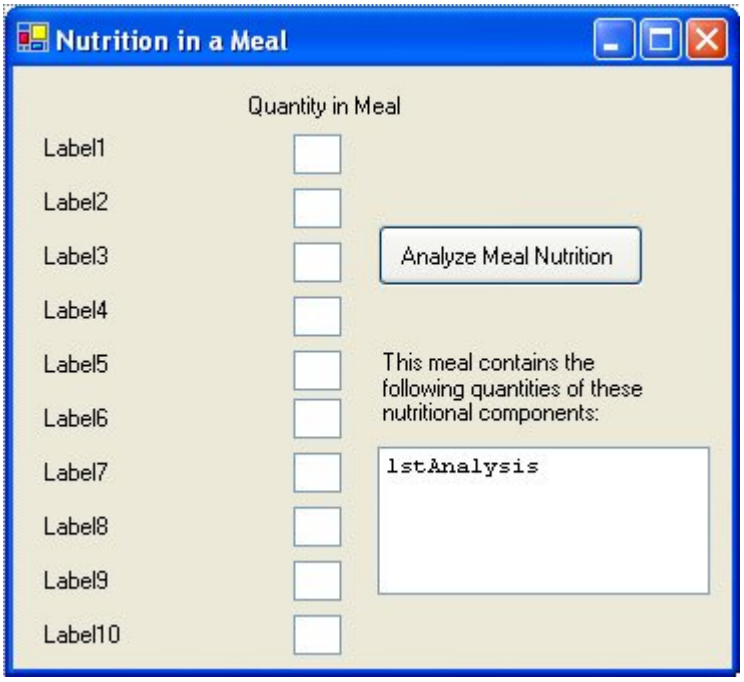
The Center for Science in the Public Interest publishes The Nutrition Scorebook, a highly respected rating of foods. The top two foods in each of five categories are shown in Table 7.11 along with some information on their composition. The following program computes the nutritional content of a meal. The table is read into an array, and then the program requests the quantities of each food item that is part of the meal. The program then computes the amounts of each nutritional component consumed by summing each column with each entry weighted by the quantity of the food item. Coding is simplified by using an array of labels to hold the food names and an array of text boxes to hold the amounts input by the user. The five nutrients of interest and the actual names and nutrient values of the foods to be used in building a meal are read from the text file NUTTABLE.TXT.

Table 7.11. Composition of 10 top-rated foods.

	Calories	Protein (grams)	Fat (grams)	Vit A (IU)	Calcium (mg)
Spinach (1 cup)	23	3	0.3	8100	93
Sweet potato (1 med.)	160	2	1	9230	46
Yogurt (8 oz.)	230	10	3	120	343
Skim milk (1 cup)	85	8	0	500	302
Whole wheat bread (1 slice)	65	3	1	0	24
Brown rice (1 cup)	178	3.8	0.9	0	18
Watermelon (1 wedge)	110	2	1	2510	30
Papaya (1 lg.)	156	2.4	0.4	7000	80

Tuna in water (1 lb)	575	126.8	3.6	0	73
Lobster (1 med.)	405	28.8	26.6	984	190

[Page 381]



Object	Property	Setting
frmNutrition	Text	Nutrition in a Meal
Label1Label10 lblQnty	Text	Quantity in Meal
TextBox1TextBox10 btnAnalyze	Text	Analyze Meal Nutrition
lblStatement	AutoSize	False
	Text	This meal contains the following quantities of these nutritional components:
	Visible	False
lstAnalysis	Visible	False

```
Dim lblarrayFood(9) As Label
Dim txtarrayQnty(9) As TextBox
Dim nutName(4) As String        'Nutrient names
Dim nutTable(9, 4) As Double    'Nutrient values for each food

Private Sub frmNutrition_Load(...) Handles MyBase.Load
  'Fill arrays, set text for labels
  lblarrayFood(0) = Label1
  lblarrayFood(1) = Label2
  lblarrayFood(2) = Label3
  lblarrayFood(3) = Label4
  lblarrayFood(4) = Label5
  lblarrayFood(5) = Label6
  lblarrayFood(6) = Label7
  lblarrayFood(7) = Label8
  lblarrayFood(8) = Label9
  lblarrayFood(9) = Label10
  txtarrayQnty(0) = TextBox1
  txtarrayQnty(1) = TextBox2
  txtarrayQnty(2) = TextBox3
  txtarrayQnty(3) = TextBox4
  txtarrayQnty(4) = TextBox5
  txtarrayQnty(5) = TextBox6
  txtarrayQnty(6) = TextBox7
  txtarrayQnty(7) = TextBox8
  txtarrayQnty(8) = TextBox9
  txtarrayQnty(9) = TextBox10
```

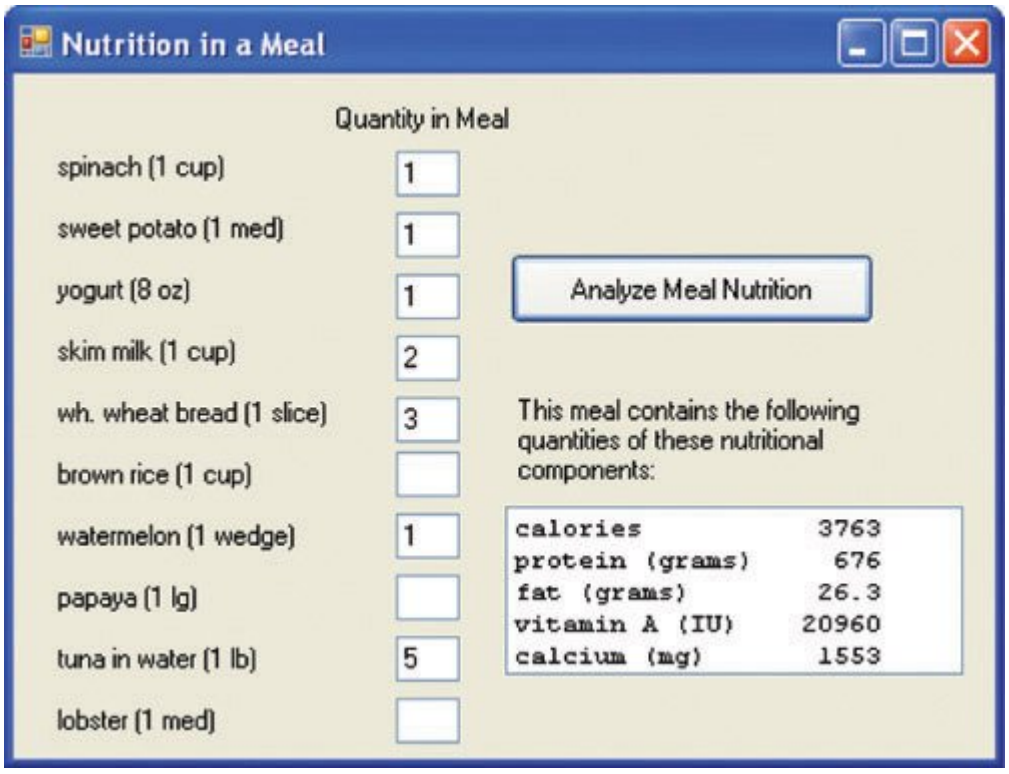
[Page 382]

```
  Dim foodName As String
  'Fill arrays; assign label captions
  Dim sr As IO.StreamReader = IO.File.OpenText("NUTTABLE.TXT")
  For i As Integer = 0 To 4
    nutName(i) = sr.ReadLine
  Next
  For i As Integer = 0 To 9
    foodName = sr.ReadLine
    lblarrayFood(i).Text = foodName
    For j As Integer = 0 To 4
      nutTable(i, j) = CDbl(sr.ReadLine)
    Next
  Next
  sr.Close()
End Sub
Private Sub btnAnalyze_Click(...) Handles btnAnalyze.Click
  'Determine the nutritional content of a meal
  Dim quantity(9) As Double   'amount of food in meal
  GetAmounts(quantity)
  lblStatement.Visible = True
  lstAnalysis.Visible = True
  ShowData(quantity)
End Sub
Sub GetAmounts(ByRef quantity() As Double)
  'Obtain quantities of foods consumed
  For i As Integer = 0 To 9
    If txtarrayQnty(i).Text <> "" Then
      quantity(i) = CDbl(txtarrayQnty(i).Text)
    Else
```

```
      quantity(i) = 0
    End If
  Next
End Sub
Sub ShowData(ByVal quantity() As Double)
  'Display amount of each component
  Dim amount As Double
  Dim fmtStr As String = "{0,-15}{1,8}"
  lstAnalysis.Items.Clear()
  For col As Integer = 0 To 4
    amount = 0
    For row As Integer = 0 To 9
      amount = amount + quantity(row) * nutTable(row, col)
    Next
    lstAnalysis.Items.Add(String.Format(fmtStr, nutName(col), amount))
  Next
End Sub
```

[Page 383]

[Run, type quantities into the text boxes, and press the button.]



Comments

1. We can define three- (or higher-) dimensional arrays much as we do two-dimensional arrays. A three-dimensional array uses three subscripts, and the assignment of values requires a triple-nested loop. As an example, a meteorologist might use a three-dimensional array to record temperatures for various dates, times, and elevations. The array might be created by the statement

```
Dim temps(31, 24, 14) As Double
```

2. A ReDim statement cannot change the number of dimensions of an array. For instance, it cannot change a one-dimensional array into a two-dimensional array.

Practice Problems 7.5

1. Consider the road-mileage program in Example 1. How can the program be modified so the actual names of the cities can be supplied by the user?
2. In what types of problems are two-dimensional arrays superior to arrays of structures?

[Page 384]

Exercises 7.5

In Exercises 1 through 8, determine the output displayed when the button is clicked. All Dim statements for arrays are in the Declarations section of the Code window.

1.
```
Dim a(20, 30) As Double

Private Sub btnDisplay_Click(...) Handles btnDisplay.Click
  a(3, 5) = 6
  a(5, 3) = 2 * a(3, 5)
  txtOutput.Text = CStr(a(5, 3))
End Sub
```

2.
```
Dim years(100, 50) As Double

Private Sub btnDisplay_Click(...) Handles btnDisplay.Click
  Dim x As Integer = 7, y As Integer = 8
  years(x, y) = 1937
  txtOutput.Text = CStr(years(7, 8) + 60)
End Sub
```

3.
```
Dim w(10, 15) As String

Private Sub btnDisplay_Click(...) Handles btnDisplay.Click
  Dim d As String = "Dorothy", n As Integer = 1
  w(1, 1) = d
  txtOutput.Text = w(n, n)
End Sub
```

4.
```
Dim actor(5, 5) As String
```

```
Private Sub btnDisplay_Click(...) Handles btnDisplay.Click
  Dim a As Integer = 2, b As Integer = 3, temp As Integer
  actor(a, b) = "Bogart"
  temp = a
  a = b
  b = temp
  lstOutput.Items.Add("1. " & actor(a, b))
  lstOutput.Items.Add("2. " & actor(b, a))
End Sub
```

5.

```
Dim a(,) As Integer

Private Sub btnDisplay_Click(...) Handles btnDisplay.Click
  Dim p, q, sum As Integer
  Dim sr As IO.StreamReader = IO.File.OpenText("DATA.TXT")
  p = CInt(sr.ReadLine)
  q = CInt(sr.ReadLine)
  ReDim a(p, q)
  For j As Integer = 0 To p
    sum = 0
```

[Page 385]

```
  For k As Integer = 0 To q
    a(j, k) = CInt(sr.ReadLine)
    sum += a(j, k)
  Next
  lstOutput.Items.Add(sum)
  Next
  sr.Close()
End Sub
```

(Assume that the eight lines of the file DATA.TXT contain the following data: 1, 2, 4, 1, 6, 5, 8, 2.)

6.

```
Dim a(3, 4) As Integer

Private Sub btnDisplay_Click(...) Handles btnDisplay.Click
  Dim row As String
  For j As Integer = 0 To 3
    row = ""
    For k As Integer = 0 To 4
      a(j, k) = j + k
      row &= a(j, k) & " "
   Next
   lstOutput.Items.Add(row)
  Next
End Sub
```

7.

```
Dim s(2, 2) As Double
```

```
Private Sub btnDisplay_Click(...) Handles btnDisplay.Click
  Dim sr As IO.StreamReader = IO.File.OpenText("DATA.TXT")
  For j As Integer = 0 To 2
    For k As Integer = 0 To 2
      s(j, k) = CDbl(sr.ReadLine)
    Next
  Next
  sr.Close()
  For j As Integer = 0 To 2
    lstOutput.Items.Add(s(j, j))
  Next
End Sub
```

(Assume that the nine lines of the file DATA.TXT contain the following entries: 1, 2, 3, 4, 3, 2, 3, 4, 5.)

8.

```
Dim m(,) As Integer

Private Sub btnDisplay_Click(...) Handles btnDisplay.Click
  Dim x, y As Integer
  Dim sr As IO.StreamReader = IO.File.OpenText("DATA.TXT")
  x = CInt(sr.ReadLine)
  y = CInt(sr.ReadLine)
  ReDim m(x, y)
```

[Page 386]

```
    For j As Integer = 0 To x
    For k As Integer = 0 To y - j
      m(j, k) = CInt(sr.ReadLine)
      lstOutput.Items.Add(m(j, k) - k)
    Next
  Next
  sr.Close()
End Sub
```

(Assume that the 10 lines of the file DATA.TXT contain the following entries: 1, 2, 6, 3, 2, 1, 3, 4, 9, 8.)

In Exercises 9 and 10, identify the errors.

9.

```
Dim a(2, 3) As Integer

Private Sub btnDisplay_Click(...) Handles btnDisplay.Click
  'Fill an array
  Dim sr As IO.StreamReader = IO.File.OpenText("DATA.TXT")
  For j As Integer = 0 To 3
    For k As Integer = 0 To 2
      a(j, k) = CInt(sr.ReadLine)
    Next
  Next
  sr.Close()
End Sub
```

(Assume that the 12 lines of the file DATA.TXT contain the following entries: 1, 2, 3, 4, 5, 6, 7, 8, 9, 0, 1, 2.)

10.
```
Dim score(2, 2) As Integer

Private Sub frmScores_Load(...) Handles MyBase.Load
  'Fill array from text file
  Dim student As Integer
  Dim sr As IO.StreamReader = IO.File.OpenText("SCORES.TXT")
  For j As Integer = 0 To 2
    student = CInt(sr.ReadLine)
    For k As Integer = 0 To 2
      score(k, j) = CInt(sr.ReadLine)
    Next
  Next
End Sub
Private Sub btnDisplay_Click(...) Handles btnDisplay.Click
  'Report individual scores
  Dim student, exam As Integer
  student = CInt(txtStudent.Text)
  exam = CInt(txtExam.Text)
  If (student >=0 And student<=2) And (exam>=0 And exam<=2) Then
    txtOutput.Text = CStr(score(student, exam))
  End If
End Sub
```

[Page 387]

(Assume that the 12 lines of the file SCORES.TXT contain the following data: 0, 80, 85, 90, 1, 72, 80, 88, 2, 87, 93, 90.)

In Exercises 11 through 14, write a procedure to perform the stated task.

11. Given an array declared with the statement `Dim a(10, 10) As Double`, set the entries in the jth column to j (for j=0,...,10).

12. Given an array declared with the statement `Dim a(10, 10) As Double`, and values assigned to each entry, compute the sum of the values in row 10.

13. Given an array declared with the statement `Dim a(10, 10) As Double`, and values assigned to each entry, interchange the values in rows 2 and 3.

14. Given an array declared with the statement `Dim a(3, 45) As Double`, and values assigned to each entry, find the greatest value and the places (possibly more than one) at which git occurs.

In Exercises 15 through 25, write a program to perform the stated task.

15. A company has two stores (1 and 2), and each store sells three items (1, 2, and 3). The following tables give the inventory at the beginning of the day and the amount of each item sold during that day.

Beginning Inventory

		ITEM		
		1	2	3
Store	1	25	64	23
	2	12	82	19

Sales for Day

		ITEM		
		1	2	3
Store	1	7	45	11
	2	4	24	8

a. Record the values of each table in an array.

b. Adjust the values in the first array to hold the inventories at the end of the day and display these new inventories.

c. Calculate and display the number of items in each store at the end of the day.

16. Table 7.12 gives the results of a survey on the uses of computers in the workplace. Each entry shows the percentage of respondents from the age category that use the computer for the indicated purpose.

Table 7.12. Workers Using Computers on the job.

Age	Word Processing	Spreadsheets	Internet/ e-mail	Calendar/ schedule	Programming
1824	55.1	53.7	58.2	47.5	12.7
2529	67.5	65.9	73.7	54.9	17.2
3039	69.5	66.5	75.0	56.6	17.4
4049	68.8	63.6	73.5	54.9	15.9
5059	69.1	61.2	74.0	51.3	12.5
60 and older	61.0	50.1	65.7	40.2	12.3

Source: U.S. Center of Educational Statistics, Digest of Educational Statistics, 2001

a. Place the data from the table in an array. (Use 7-5-E16.TXT.)

b. Determine the average of the percentages in the Spreadsheets column.

[Page 388]

17. A university offers 10 courses at each of three campuses. The number of students enrolled in each is presented in Table 7.13.

a. Find the total number of course enrollments on each campus.

b. Find the total number of students taking each course.

Table 7.13. Number of students enrolled in courses.

		Course									
		1	2	3	4	5	6	7	8	9	10
	1	5	15	22	21	12	25	16	11	17	23
Campus	2	11	23	51	25	32	35	32	52	25	21
	3	2	12	32	32	25	26	29	12	15	11

18. Table 7.14 gives the 2002 and 2003 U.S. sales for the five top restaurant chains.

a. Place the data into an array.

b. Calculate the total change in sales for these five restaurant chains.

Table 7.14. Top restaurant chains.

	2002 Sales $MM	2003 Sales $MM
1. McDonald's	20.3	22.1
2. Burger King	8.7	7.9
3. Wendy's	7.1	7.5
4. Subway	5.2	5.7
5. Taco Bell	5.2	5.3
Source: QSR Magazine, Dec. 2004		

19. The scores for the top three golfers at the 2005 Buick Invitational are shown in Table 7.15

1. Place the data into an array.

2. Compute the total score for each player.

3. Compute the average score for each round.

Table 7.15. 2005 Buick Invitational.

	Round			
	1	2	3	4
Tiger Woods	69	63	72	68
Luke Donald	68	67	67	73
Bernhard Langer	69	69	67	72

[Page 389]

20. Table 7.16 contains part of the pay schedule for federal employees in Washington, D.C.Table 7.17 gives the number of employees of each classification in a certain division. Place the data from each table into an array and compute the amount of money this division pays for salaries during the year.

Table 7.16. 2005 pay schedule for federal white-collar workers.

	Step			
	1	2	3	4
GS-1	16,016	16,550	17,083	17,613
GS-2	18,007	18,435	19,031	19,537
GS-3	19,647	20,302	20,957	21,612
GS-4	22,056	22,791	23,526	24,261
GS-5	24,677	25,500	26,323	27,146
GS-6	27,507	28,424	29,341	30,258
GS-7	30,567	31,586	32,605	33,624

Table 7.17. Number of employees in each category.

	1	**2**	**3**	**4**
GS-1	0	0	2	1
GS-2	2	3	0	1
GS-3	4	2	5	7
GS-4	12	13	8	3

GS-5	4	5	0	1
GS-6	6	2	4	3
GS-7	8	1	9	2

21. Consider Table 7.10, the rankings of three university departments. Write a program that places the data into an array, allows the name of a college to be input, and gives the categories in which it appears. Of course, a college might appear more than once or not at all.

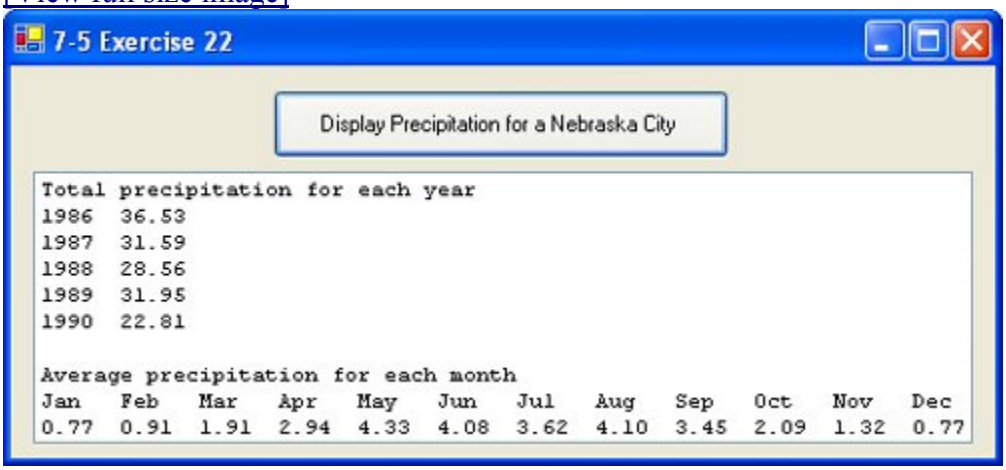
22. Table 7.18 gives the monthly precipitation for a typical Nebraska city during a five-year period. Write a program that reads the table from a text file into an array and then displays in a list box the output on the next page.

Table 7.18. Monthly precipitation (in inches) for a typical Nebraska city.

	Jan.	Feb.	Mar.	Apr.	May	June	July	Aug.	Sept.	Oct.	Nov.	Dec.
1986	0.88	1.11	2.01	3.64	6.44	5.58	4.23	4.34	4.00	2.05	1.48	0.77
1987	0.76	0.94	2.09	3.29	4.68	3.52	3.52	4.82	3.72	2.21	1.24	0.80
1988	0.67	0.80	1.75	2.70	4.01	3.88	3.72	3.78	3.55	1.88	1.21	0.61
1989	0.82	0.80	1.99	3.05	4.19	4.44	3.98	4.57	3.43	2.32	1.61	0.75
1990	0.72	0.90	1.71	2.02	2.33	2.98	2.65	2.99	2.55	1.99	1.05	0.92

[Page 390]

[View full size image]



23. Suppose that a course has 15 students enrolled and that five exams are given during the

semester. Write a program that accepts each student's name and grades as input and places the names in a one-dimensional array and the grades in a two-dimensional array. The program should then display each student's name and semester average. Also, the program should display the median for each exam. (For an odd number of grades, the median is the middle grade. For an even number of grades, it is the average of the two middle grades.)

24. An n-by-n array is called a magic square if the sums of each row, each column, and each diagonal are equal. Write a program to determine if an array is a magic square and use it to determine if either of the following arrays is a magic square. Hint: If, at any time, one of the sums is not equal to the others, the search is complete.

a.

1	15	15	4
12	6	7	9
8	10	11	5
13	3	2	16

b.

11	10	4	23	17
18	12	6	5	24
25	19	13	7	1
2	21	20	14	8
9	3	22	16	15

25. A company has three stores (1, 2, and 3), and each store sells five items (1, 2, 3, 4, and 5). The following tables give the number of items sold by each store and category on a particular day, and the cost of each item.

a. Place the data from the left-hand table in a two-dimensional array and the data from the right-hand table in a one-dimensional array.

b. Compute and display the total dollar amount of sales for each store and for the entire company.

[Page 391]

Number of Items Sold During Day

	Item					Item	Cost per item
	1	2	3	4	5	1	$12.00

	1	25	64	23	45	14	2	$17.95
Store	2	12	82	19	34	63	3	$95.00
	3	54	22	17	43	35	4	$86.50
							5	$78.00

Solutions to Practice Problems 7.5

1. Replace the masked text boxes with ordinary text boxes to hold city names. The function FindCityNum can be used to determine the subscript associated with each city. This function and the modified event procedure btnShow_Click are as follows:

```
Function FindCityNum(ByVal city As String) As Integer
  Select Case city.ToUpper
    Case "CHICAGO"
      Return 1
    Case "LOS ANGELES"
      Return 2
    Case "NEW YORK"
      Return 3
    Case "PHILADELPHIA"
      Return 4
    Case Else
      Return 0
  End Select
End Function

Private Sub btnShow_Click(...) Handles btnShow.Click
  Dim orig, dest As String
  Dim row, col As Integer 'Determine road mileage between cities
  orig = txtOrig.Text
  dest = txtDest.Text
  row = FindCityNum(orig)
  col = FindCityNum(dest)
  If (row < 1) Or (row > 4) Then
    MsgBox("City of origin not available", 0, "Error")
  ElseIf (col < 1) Or (col > 4) Then
    MsgBox("Destination city not available", 0, "Error")
  Else
    txtMiles.Text = CStr(rm(row, col))
 End If
 End Sub
```

2. Both arrays of structures and two-dimensional arrays are used to hold related data. If some of the data are numeric and some are string, then structures must be used, because all entries of a two-dimensional array must be of the same type. Arrays of structures should also be used if the data will be sorted. Two-dimensional arrays are best suited to

tabular data.

PREV PREV NEXT NEXT

[Page 392]

7.6. A Case Study: A Sophisticated Cash Register

One of the most prevalent types of computer equipment today is the cash register. This machine has evolved considerably from the time when it functioned like an old-fashioned mechanical typewriter, recording the amount of each sale on a strip of paper and keeping a running sum. Today, cash registers are located not only in stores, but also within gasoline pumps and airport kiosks, performing many functions above and beyond printing receipts. The data from customers' purchases can be stored on a central computer and analyzed to determine trends in buying behavior. For example, a grocery store may glean from cash register transactions that customers who purchase potato chips are likely to also purchase soft drinks. This might lead the store to stock soft drinks in the same aisle as snacks. In the retail industry, the location of a cash register is called the "point of sale" (abbreviated as "POS").

The Design of the Program

This case study develops a sophisticated cash register that supports three main features: First, it accepts user input for the quantity, price, and department of a particular article. Second, it calculates the totals including sales tax and stores those data along with the payment method into an array. Finally, the program generates a report on the stored data. Each of these features is a task that is executed by clicking on a button:

a. Enter the price, quantity, and department of a new article to purchase. Display the article's data.

b. Display the subtotal, sales tax, and total, and record the purchase and payment method in an array.

c. Generate a report showing (a) the quantity and sales for each department and (b) the number of receipts and amounts for each payment method.

The User Interface

Two text boxes are used to collect input for quantity and price. Buttons that represent each department are used to record and display this data in an output list box. Each payment type is represented by a button that finalizes the purchase and displays the totals. A report button is enabled after a purchase is finalized and, when pressed, generates the report in the same list box. See Figure 7.7 and Table 7.19.

[Page 393]

Figure 7.7. Sample output after pressing the bottom button.

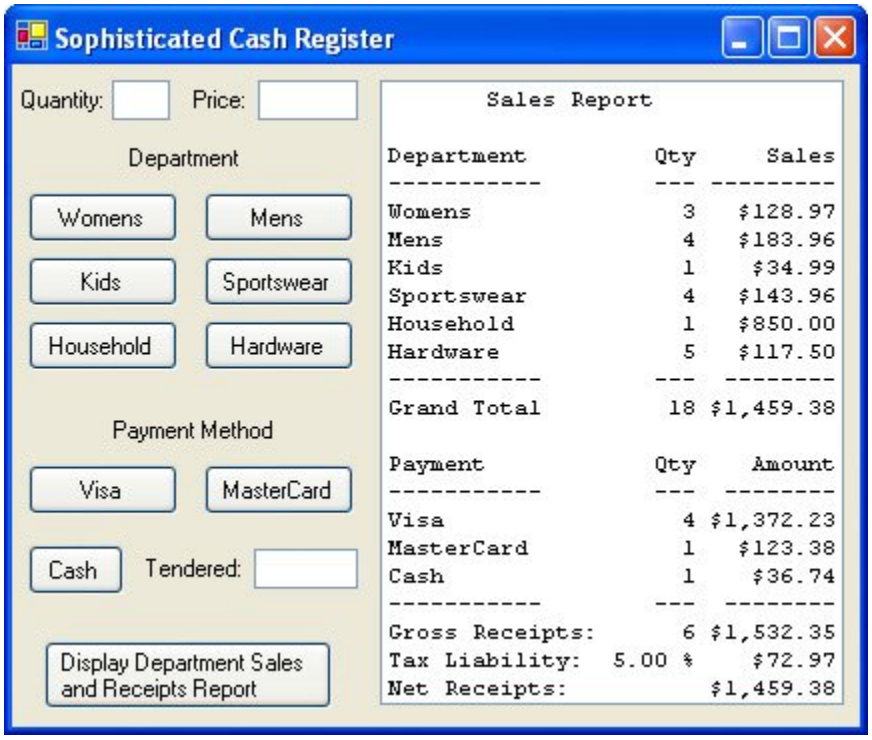


Table 7.19. Controls that comprise the user interface.

Object	Property	Setting
frmCashRegister	Text	Sophisticated Cash Register
lblQuantity	Text	Quantity:
txtQuantity		
lblPrice	Text	Price:
txtPrice		
lblDepartment	Text	Department
btnWomens	Text	Womens
btnMens	Text	Mens
btnKids	Text	Kids
btnSportswear	Text	Sportswear
btnHousehold	Text	Household
btnHardware	Text	Hardware
lblMethod	Text	Payment Method
btnVisa	Text	Visa
btnMasterCard	Text	MasterCard

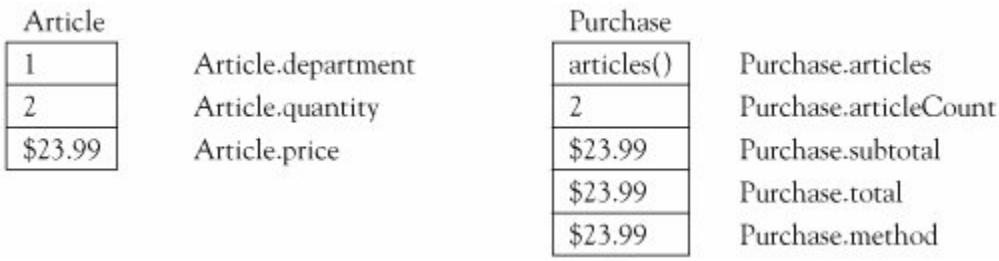
btnCash	Text	Cash
lblCash	Text	Tendered:
txtCash		
btnReport	Enabled	False
Text	Display	Department Sales and Receipts Report
lstOutput		

[Page 394]

The Data Structures

The cash register inputs, stores, and processes data. These data are organized into related fields that lend themselves to structure data types. The program employs an Article structure that stores the data for each article that is purchased and a Purchase structure that stores all of the articles and the totals on one receipt. An array of Purchase structures is declared as a class-level variable and holds all of the data for the store. Figure 7.8 shows a representation of the way the structures are organized.

Figure 7.8. Structures used in the Sophisticated Cash Register program.



Coding the Program

The top row of Figure 7.9 shows the different events to which the program must respond. Table 7.20 identifies the corresponding event procedures and the general procedures they call. Let's examine each event procedure:

1. frmCashRegister_Load initializes the first Purchase structure by dimensioning its articles array to an initial upper bound of 5. It is necessary to do this in the Load procedure because arrays cannot be sized within the structure definition block, yet they must be dimensioned before they are accessed.

2. RingupArticle is called by the event procedures connected to the buttons that represent the department names (Womens, Mens, Kids, etc.). It takes as its parameter the number of the department that was clicked. The procedure stores the department, quantity, and price of the article into a new Article structure. If the quantity and price are valid (that is, they are greater than zero), then the new article is stored into the array that is a member of the current Purchase structure. If that array does not have room to accommodate the new article, the array is enlarged

using a ReDim Preserve statement. Finally, the procedure calls the Display-Article procedure and resets the quantity to 1 for the next article.

[Page 395]

Figure 7.9. Hierarchy chart for Sophisticated Cash Register program.
(This item is displayed on page 394 in the print version)

[View full size image]

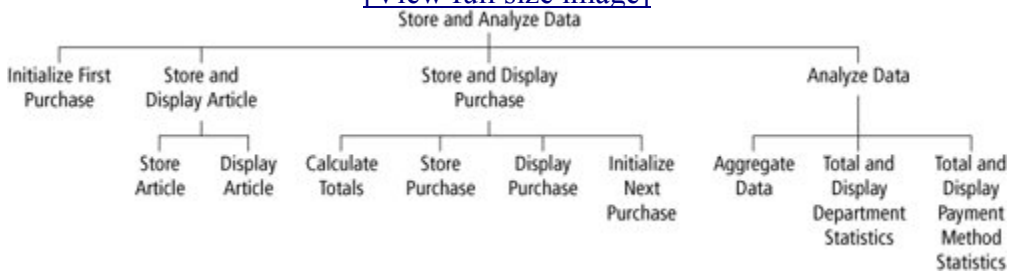


Table 7.20. Tasks and their procedures.

1.	Initialize First Purchase	frmCashRegister_Load
2.	Store and Display Article	RingupArticle
	2.1 Store Article	RingupArticle
	2.2 Display Article	DisplayArticle
3.	Store and Display Purchase	TotalPurchase
	3.1 Calculate Totals	TotalPurchase
	3.2 Store Purchase	TotalPurchase
	3.3 Display Purchase	DisplayTotals
	3.4 Initialize Next Purchase	TotalPurchase
4.	Analyze Data	btnReport_Click
	4.1 Aggregate Data	btnReport_Click
	4.2 Total and Display Department Statistics	btnReport_Click
	4.2 Total and Display Payment Method Statistics	btnReport_Click

The DisplayArticle procedure displays the quantity, department, price, and total (that is, quantity times price) for an article in the output list box. A heading is displayed along with the first purchase on the receipt. The department names are stored in a string array called department() that is declared as a class-level variable. The procedure displays the article's department name by using its department number as the subscript in the department() array.

3. TotalPurchase is called by the event procedures that are connected to the buttons representing the

methods of payment (Visa, MasterCard, Cash). It takes as its parameter the number of the payment method that was clicked. The procedure first checks to see if any articles have been stored. If not, a message box is displayed. Otherwise, the articles are totaled, and sales tax is applied to calculate the purchase's subtotal and total. If the payment method is Cash, the procedure verifies that the amount tendered by the customer is at least the total amount. If it isn't, a message box is displayed. Otherwise, the purchase is stored, the DisplayTotals procedure is called, and the articles array in the next purchase structure is ReDimmed to the default upper bound of 4.

The technique used to store a purchase requires elaboration. The class-level array purchases() is declared in the class with an initial upper bound of 30. The integer counter count keeps track of the number of stored purchases and is initialized to the default value of zero. New articles are stored in the purchases() element with subscript count. When a payment method is clicked, the purchase is finalized. The action that does this is the line count + =1, which increments the counter by one. Therefore, the new purchase is stored in the purchases() array all along, but it is only recognized as "stored" when its index is less than the counter.

[Page 396]

The DisplayTotals procedure displays the subtotal, applicable sales tax, and total for the purchase. It also displays the payment method in the following manner: If the payment method is Cash (method #3), it displays the cash tendered and the change due. Otherwise it displays the type of credit card used.

4. btnReport_Click aggregates the data in the purchases() array and displays the resulting analysis. It first declares the arrays and variables that are used for the results. It then loops through all of the articles for each purchase (using a nested loop) to aggregate each article into a department result and a payment method result. It uses the department and payment method numbers as subscripts for the result arrays, the four arrays declared at the top of the procedure. The department statistics are totaled and displayed, then the payment method statistics are totaled and displayed.P

```
'Define the Purchase and Article structures
Structure Article
Dim department As Integer  'Department number, see department() below
Dim quantity As Integer    'Quantity of articles purchased
Dim price As Double        'Price of each article
End Structure
Structure Purchase
 Dim articles() As Article   'Articles purchased
Dim articleCount As Integer 'Number of articles purchased
Dim subtotal As Double      'Subtotal of articles' prices times quantities
Dim total As Double 'Total of purchase including tax
Dim method As Integer       'Method of payment, see method() below
End Structure
'Declare the class-level variables
Dim  purchases(30) As Purchase 'Array that holds all purchases
Dim count As Integer          'Number of completed purchases
Dim  taxRate As Double = 0.05  '5% sales tax
Dim  department() As String = { " Womens ",   " Mens ",   " Kids ",  _
     " Sportswear ",   " Household ",   " Hardware "}
Dim  method() As String = { " Visa ",   " MasterCard ",   " Cash "}
Private Sub frmCashRegister_Load(...) Handles MyBase.Load
'Initialize the first purchase
ReDim purchases(0).articles(4)
```

```
End Sub
Sub RingupArticle(ByVal dept As Integer)
 'Store the user's input into a new article
Dim newArticle As Article      'New article to hold user input
Dim size As Integer              'Size of the articles array
  With newArticle
    .department = dept
    .quantity = CInt(txtQuantity.Text)
```

[Page 397]

```
 'If blank price then use default of 0
 If txtPrice.Text =  ""  Then
   .price = 0
 Else
   .price = CDbl(txtPrice.Text)
 End If
End With
 'If price and quantity are valid, then store and display article
 If (newArticle.quantity > 0) And (newArticle.price > 0) Then
   'Increment the number of articles in the current purchase
   purchases(count).articleCount += 1
   'Ensure the Purchase's articles array has room for the new article
   size = purchases(count).articles.GetUpperBound(0)
   If purchases(count).articleCount >= size Then
  'If not enough room, increase the array size by 5
  ReDim Preserve purchases(count).articles(size + 5)
 End If
 'Store the new article into the purchase
 purchases(count).articles(purchases(count).articleCount - 1) = _
                        newArticle
 'Display the article in the receipt
 DisplayArticle()
 'Clear the quantity and price for the next article
 txtQuantity.Clear()
 txtPrice.Clear()
End If
 'Disable the report button until the article is paid for
 btnReport.Enabled = False
 txtQuantity.Focus() 'Focus on the quantity for the next article
End Sub
Private Sub btnWomens_Click(...) Handles btnWomens.Click
  RingupArticle(0)
End Sub
Private Sub btnMens_Click(...) Handles btnMens.Click
  RingupArticle(1)
End Sub
Private Sub btnKids_Click(...) Handles btnKids.Click
  RingupArticle(2)
End Sub
Private Sub btnSportswear_Click(...) Handles btnSportswear.Click
  RingupArticle(3)
End Sub
Private Sub btnHousehold_Click(...) Handles btnHousehold.Click
  RingupArticle(4)
End Sub
```

[Page 398]

```
Private Sub btnHardware_Click(...) Handles btnHardware.Click
  RingupArticle(5)
End Sub
```

```
Sub  DisplayArticle()
  Dim cost As Double
  Dim fmtStr As String =  "  {0,3} {1,-12} {2,7:C} {3,7:C} "
  'If article is the first, clear the display and display the header
  If purchases(count).articleCount = 1 Then
    lstOutput.Items.Clear()
    lstOutput.Items.Add("   Customer Receipt ")
    lstOutput.Items.Add("")
    lstOutput.Items.Add(String.Format(fmtStr,  " Qty " ,  " Dept ", _
    "   Price " ,  "   Total "))
   lstOutput.Items.Add(String.Format(fmtStr,  " --- " ,  " ------------ " , _
    " ------- " ,  " ------- "))
  End If
   'Display the article's information
   With purchases(count).articles(purchases(count).articleCount - 1)
     cost = .quantity * .price
     lstOutput.Items.Add(String.Format(fmtStr, .quantity, _
                         department(.department), .price, cost))
   End With
 End Sub
 Sub  TotalPurchase(ByVal meth As Integer)
   Dim cash As Double 'Amount of cash tendered
   Dim flag As Boolean = True 'If purchase is valid
   With purchases(count)
     'If no articles purchased then display message and exit
     If .articleCount < 1 Then
       MsgBox( "  No articles purchased. " , 0,  "  No sale ")
       txtQuantity.Focus()
       flag = False
    End If
   'Calculate and store totals
   .subtotal = 0
   For i As Integer = 0 To .articleCount - 1
     .subtotal += .articles(i).price * .articles(i).quantity
   Next
   'Add sales tax to get total
   .total = (1 + taxRate) * .subtotal
   'Store the method of payment
   .method = meth
   'If cash method, then get the amount tendered by customer
   If .method = 2 Then
     If txtCash.Text =  ""  Then
       cash = 0
```

```
  Else
    cash = CDbl(txtCash.Text)
  End If
  'If tendered amount is not enough, then display message
  If (cash < .total) Then
    MsgBox( "  Not enough cash tendered (need  "  & _
      FormatCurrency(.total) &  " ) " , 0,  "Insufficient Cash")
    txtCash.Focus() 'Focus on cash for reinput
    flag = False
  End If
 End If
End With
'If purchase is valid, then store and display it
If flag Then
   'Increment the counter so the latest purchase is considered stored
```

```
    count += 1
    'Display purchase totals
    DisplayTotals()
    'Make sure there is room for the next purchase and initialize it
    If purchases.GetUpperBound(0) <= count Then
      ReDim purchases(count + 5)
    End If
    ReDim purchases(count).articles(4)
  End If
  'Enable the report button
    btnReport.Enabled = True
  End Sub
  Sub  DisplayTotals()
     'Display subtotal, tax, and total
     Dim prompt As String =  "Tax: ( "  & FormatPercent(taxRate) &  " ) "
     Dim cash As Double
     Dim fmtStr As String =  "{0,3} {1,-12} {2,7} {3,7:C} "
     With purchases(count - 1)
       lstOutput.Items.Add(String.Format(fmtStr,  "",   "",   "",   " ------- "))
       lstOutput.Items.Add(String.Format(fmtStr,  "",   "  Sub Total: ",   "", _
   .subtotal))
       lstOutput.Items.Add(String.Format(fmtStr,  "" , prompt,  "" , _
                                         .total - .subtotal))
       lstOutput.Items.Add(String.Format(fmtStr,  "",   "",   "",   " ------- "))
       lstOutput.Items.Add(String.Format(fmtStr,  "",   " Total: ",   "",  .total))
       'If cash purchase, then display amount tendered and change
       If .method = 2 Then
         cash = CDbl(txtCash.Text)
         lstOutput.Items.Add( "")
         lstOutput.Items.Add(String.Format(fmtStr,  "",  "Cash:",  "" , cash))
         lstOutput.Items.Add(String.Format(fmtStr, "",   " Change: ",   "" , _
                                         cash - .total))
```



```
    Else
     'If not cash, mention how tendered
     lstOutput.Items.Add( "   Charged to  "  & method(.method))
  End If
  End With
  'Clear the cash, and set focus to quantity for the next purchase
  txtCash.Clear()
  txtQuantity.Focus()
End Sub
Private Sub btnVisa_Click(...) Handles btnVisa.Click
   TotalPurchase(0)
End Sub
Private Sub btnMasterCard_Click(...) Handles btnMasterCard.Click
  TotalPurchase(1)
End Sub
Private Sub btnCash_Click(...) Handles btnCash.Click
  TotalPurchase(2)
End Sub
Private Sub btnReport_Click(...) Handles btnReport.Click
  Dim deptQuant(5), deptQuantT As Integer  'Quantity per department
  Dim deptSales(5), deptSalesT As Double   'Sales per department
  Dim methQuant(2), methQuantT As Integer  'Quantity per method
  Dim methAmnt(2), methAmntT As Double     'Sales per method
  Dim liability As Double                  'Amount of tax liability
  Dim fmtStr1 As String =  "{0,-15} {1,6} {2,9:C}"
  Dim fmtStr2 As String =  "{0,-15} {1,6:P} {2,9:C}"
```

```
  'Sum quantity and total amounts per department, without tax
  For i As Integer = 0 To count - 1
    'Sum quantity and total amounts per payment method
   methQuant(purchases(i).method) += 1
    methAmnt(purchases(i).method) += purchases(i).total
    For j As Integer = 0 To purchases(i).articleCount - 1
      With purchases(i).articles(j)
        deptQuant(.department) += .quantity
        deptSales(.department) += .price * .quantity
      End With
    Next
  Next
  'Display the quantity and sales per department
  lstOutput.Items.Clear()
  lstOutput.Items.Add("   Sales Report ")
  lstOutput.Items.Add("")
  lstOutput.Items.Add(String.Format(fmtStr1,  "Department",  "Qty" , _
    "   Sales "))
   lstOutput.Items.Add(String.Format(fmtStr1,  " ----------- ",  " --- " , _
    " --------- "))
```

```
For i As Integer = 0 To 5
  lstOutput.Items.Add(String.Format(fmtStr1, department(i), _
                        deptQuant(i), deptSales(i)))
  deptQuantT += deptQuant(i)
  deptSalesT += deptSales(i)
 Next
 lstOutput.Items.Add(String.Format(fmtStr1,  " ----------- " ,  " --- " , _
                                          " -------- "))
 lstOutput.Items.Add(String.Format(fmtStr1,  "  Grand Total " , _
                                     deptQuantT, deptSalesT))
 'Display the quantity and sales per payment method
 lstOutput.Items.Add( "")
 lstOutput.Items.Add(String.Format(fmtStr1,  "  Payment  ",   " Qty " , _
    "   Amount "))
   lstOutput.Items.Add(String.Format(fmtStr1,  " ----------- ",   " --- " , _
                                       " -------- "))
 For i As Integer = 0 To 2
   lstOutput.Items.Add(String.Format(fmtStr1, method(i), _
                                     methQuant(i), methAmnt(i)))
   methQuantT += methQuant(i)
   methAmntT += methAmnt(i)
 Next
 'Calculate the total amount of sales tax
 liability = methAmntT - deptSalesT
 lstOutput.Items.Add(String.Format(fmtStr1,  " ----------- ",   " --- ",  _
                                     " -------- "))
 lstOutput.Items.Add(String.Format(fmtStr1,  "Gross Receipts:" , _
                                    methQuantT, methAmntT))
 lstOutput.Items.Add(String.Format(fmtStr2,  "Tax Liability:" , _
                                         taxRate, liability))
 lstOutput.Items.Add(String.Format(fmtStr1,  "Net Receipts:" , "", _
                                          deptSalesT))
 txtQuantity.Focus() 'Focus on quantity for next article
End Sub
```

PREV PREV NEXT NEXT

[Page 401 (continued)]

Chapter 7 Summary

1. For programming purposes, tabular data are most efficiently processed if stored in an array. An array is declared with a Dim statement, which also can specify its size and initial values. The size of an already declared array can be specified or changed with a ReDim or ReDim Preserve statement. The highest subscript of the array is called its upper bound and is returned with the GetUpperBound(0) method. The Clear statement releases all memory allocated to an array.

2. An array of labels, text boxes, or buttons is referred to as a control array.

3. A structure is a composite programmer-designed data type with a fixed number of fields, each of which can be of any data type.

[Page 402]

4. Two of the best-known methods for ordering (or sorting) arrays are the bubble sort and the Shell sort.

5. Any array can be searched sequentially to find the subscript associated with a sought- after value. Ordered arrays can be searched most efficiently by a binary search.

6. A table can be effectively stored in a two-dimensional array.

PREV PREV NEXT NEXT

[Page 402 (continued)]

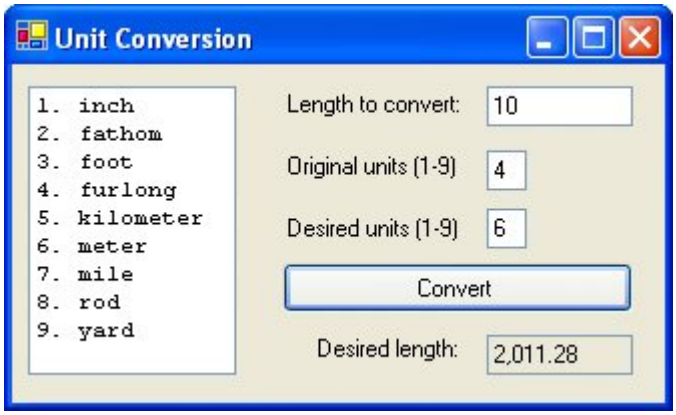
Chapter 7 Programming Projects

1. Table 7.21 contains some lengths in terms of feet. Write a program that displays the nine different units of measure; requests the unit to convert from, the unit to convert to, and the quantity to be converted, and then displays the converted quantity. A typical outcome is shown in Figure 7.10.

Table 7.21. Equivalent lengths.

1 inch = .0833 foot	1 rod = 16.5 feet
1 yard = 3 feet	1 furlong = 660 feet
1 meter = 3.28155 feet	1 kilometer = 3281.5 feet
1 fathom = 6 feet	1 mile = 5280 feet

Figure 7.10. Possible outcome of Programming Project 1.



2. Statisticians use the concepts of mean and standard deviation to describe a collection of data. The mean is the average value of the items, and the standard deviation measures the spread or dispersal of the numbers about the mean. Formally, if x1,x2,x3,...,xn,is a collection of data, then

$$\text{mean} = m = \frac{x_1 + x_2 + x_3 + \cdots + x_n}{n}$$

standard deviation =

$$s = \sqrt{\frac{(x_1 - m)^2 + (x_2 - m)^2 + (x_3 - m)^2 + \cdots + (x_n - m)^2}{n - 1}}$$

[Page 403]

Write a computer program to

a. place the exam scores 59, 60, 65, 75, 56, 90, 66, 62, 98, 72, 95, 71, 63, 77, 65, 77, 65, 50, 85, and 62 into an array;

b. calculate the mean and standard deviation of the exam scores;

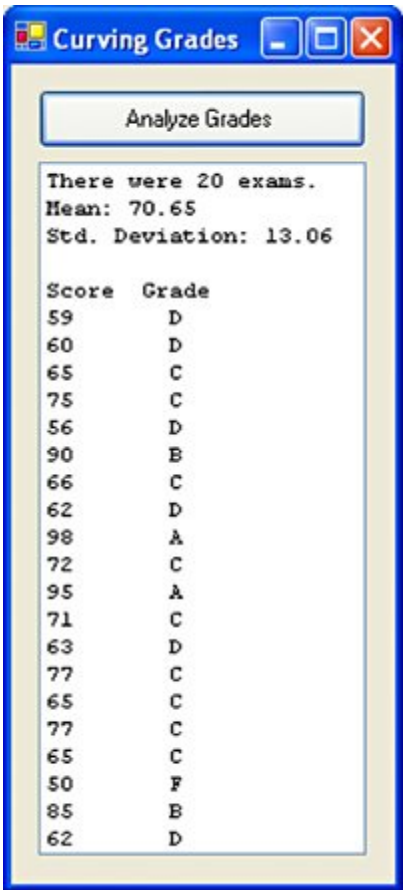
c. assign letter grades to each exam score, ES, as follows:

ES m + 1.5s	A
m + .5s ES < m + 1.5s	B
m .5s ES <m + .5s	C
m 1.5s ES m .5s	D
ES < m 1.5s	F

For instance, if m were 70 and s were 12, then grades of 88 or above would receive A's, grades between 76 and 87 would receive B's, and so on. A process of this type is referred to as curving grades.

d. Display a list of the exam scores along with their corresponding grades as shown in Figure 7.11.

Figure 7.11. Output of Programming Project 2.



[Page 404]

3. Rudimentary Translator. Table 7.22 gives English words and their French and German equivalents. Store these words in a text file and read them into a structure having a member for each language. Write a program that sorts arrays according to the English words. The program should then request an English sentence as input from the keyboard and translate it into French and German. For example, if the English sentence given is MY PENCIL IS ON THE TABLE, then the French translation will be MON CRAYON EST SUR LA TABLE, and the German translation will be MEIN BLEISTIFT IST AUF DEM TISCH.

Note: Assume that the only punctuation in the sentence is a period at the end of the sentence. Use a binary search to locate each English word.

Table 7.22. English words and their French and German equivalents.

English	French	German	English	French	German

YES	OUI	JA	LARGE	GROS	GROSS
TABLE	TABLE	TISCH	NO	NON	NEIN
THE	LA	DEM	HAT	CHAPEAU	HUT
IS	EST	IST	PENCIL	CRAYON	BLEISTIFT
YELLOW	JAUNE	GELB	RED	ROUGE	ROT
FRIEND	AMI	FREUND	ON	SUR	AUF
SICK	MALADE	KRANK	AUTO	AUTO	AUTO
MY	MON	MEIN	OFTEN	SOUVENT	OFT

4. Write a program that allows a list of no more than 50 soft drinks and their percent changes in sales volume for a particular year to be input and displays the information in two lists titled gainers and losers. Each list should be sorted by the amount of the percent change. Try your program on the data for the seven soft drinks in Table 7.23.

Note: You will need to store the data initially in an array to determine the number of gainers and losers.

Table 7.23. Changes in sales volume from 2002 to 2003 of leading soft-drink brands.

Brand	% Change in Volume	Brand	% Change in Volume
Coke Classic	-3.0	Sprite	-5.0
Pepsi-Cola	-4.5	Dr. Pepper	-3.9
Diet Coke	5.0	Diet Pepsi	6.1
Mt. Dew	-1.5		
Source: Beverage Digest, 2004			

5. Each team in a six-team soccer league played each other team once. Table 7.24 shows the winners. Write a program to

a. Place the team names in an array of structures that also holds the number of wins.

b. Place the data from Table 7.24 in a two-dimensional array.

[Page 405]

Table 7.24. Soccer league winners.

	Jazz	Jets	Owls	Rams	Cubs	Zips

Jazz		Jazz	Jazz	Rams	Cubs	Jazz
Jets	Jazz		Jets	Jets	Cubs	Zips
Owls	Jazz	Jets		Rams	Owls	Owls
Rams	Rams	Jets	Rams		Rams	Rams
Cubs	Cubs	Cubs	Owls	Rams		Cubs
Zips	Jazz	Zips	Owls	Rams	Cubs	

c. Place the number of games won by each team in the array of structures.

d. Display a listing of the teams giving each team's name and number of games won. The list should be in decreasing order by the number of wins.

6. A poker hand can be stored in a two-dimensional array. The statement

```
Dim hand(3, 12) As Integer
```

declares an array with 52 elements, where the first subscript ranges over the four suits and the second subscript ranges over the thirteen denominations. A poker hand is specified by placing ones in the elements corresponding to the cards in the hand. See Figure 7.12.

Figure 7.12. Array for the poker hand A A 5 9 Q.

	A	2	3	4	5	6	7	8	9	10	J	Q	K
Club ♣	0	0	0	0	0	0	0	0	1	0	0	0	0
Diamond ♦	1	0	0	0	0	0	0	0	0	0	0	0	0
Heart ♥	1	0	0	0	0	0	0	0	0	0	0	1	0
Spade ♠	0	0	0	0	1	0	0	0	0	0	0	0	0

Write a program that requests the five cards as input from the user, creates the related array, and passes the array to procedures to determine the type of the hand: flush (all cards have the same suit), straight (cards have consecutive denominationsace can come either before 2 or after King), straight flush, four-of-a-kind, full house (3 cards of one denomination, 2 cards of another denomination), three-of-a-kind, two pairs, one pair, or none of the above.

[Page 406]

7. Airline Reservations. Write a reservation system for an airline flight. Assume the airplane has 10 rows with 4 seats in each row. Use a two-dimensional array of strings to maintain a seating chart. In addition, create an array to be used as a waiting list in case the plane is full. The waiting list

should be "first come, first served," that is, people who are added early to the list get priority over those added later. Allow the user the following three options:

a. Add a passenger to the flight or waiting list.

1. Request the passenger's name.

2. Display a chart of the seats in the airplane in tabular form.

3. If seats are available, let the passenger choose a seat. Add the passenger to the seating chart.

4. If no seats are available, place the passenger on the waiting list.

b. Remove a passenger from the flight.

1. Request the passenger's name.

2. Search the seating chart for the passenger's name and delete it.

3. If the waiting list is empty, update the array so the seat is available.

4. If the waiting list is not empty, remove the first person from the list, and give him or her the newly vacated seat.

c. Quit.

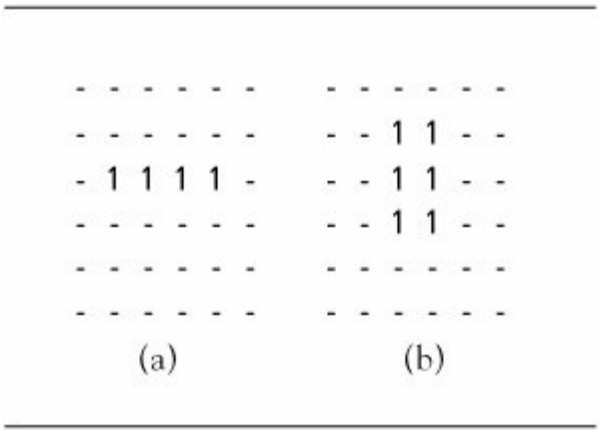
8. The Game of Life was invented by John H. Conway to model some genetic laws for birth, death, and survival. Consider a checkerboard consisting of an n-by-n array of squares. Each square can contain one individual (denoted by 1) or be empty (denoted by). Figure 7.13(a) shows a 6-by-6 board with four of the squares occupied. The future of each individual depends on the number of his neighbors. After each period of time, called a generation, certain individuals will survive, others will die due to either loneliness or overcrowding, and new individuals will be born. Each nonborder square has eight neighboring squares. After each generation, the status of the squares changes as follows:

a. An individual survives if there are two or three individuals in neighboring squares.

b. An individual dies if he has more than three individuals or less than two in neighboring squares.

c. A new individual is born into each empty square with exactly three individuals as neighbors.

Figure 7.13. Two generations.



[Page 407]

Figure 7.13(b) shows the status after one generation. Write a program to do the following:

1. Declare a two-dimensional array of size n, where n is input by the user, to hold the status of each square in the current generation. To specify the initial configuration, have the user input each row as a string of length n, and break the row into 1's or dashes with the Substring method.

2. Declare a two-dimensional array of size n to hold the status of each square in the next generation. Compute the status for each square and produce the display in Figure 7.13(b). Note: The generation changes all at once. Only current cells are used to determine which cells will contain individuals in the next generation.

3. Assign the next-generation values to the current generation and repeat as often as desired.

4. Display the individuals in each generation. Hint: The hardest part of the program is determining the number of neighbors a cell has. In general, you must check a 3-by-3 square around the cell in question. Exceptions must be made when the cell is on the edge of the array. Don't forget that a cell is not a neighbor of itself.

   (Test the program with the initial configuration shown in Figure 7.14. It is known as the figure-eight configuration and repeats after eight generations.)

Figure 7.14. The figure eight.

```
- - - - - - - - - -
- - - - - - - - - -
- - 1 1 1 - - - - -
- - 1 1 1 - - - - -
- - 1 1 1 - - - - -
- - - - - 1 1 1 - -
- - - - - 1 1 1 - -
- - - - - 1 1 1 - -
- - - - - - - - - -
- - - - - - - - - -
```

9. Every book is identified by a ten-character International Standard Book Number (ISBN) that is usually printed on the back cover of the book. The first nine characters are digits and the last character is either a digit or the letter X (which stands for ten). Three examples of ISBN numbers are 0-13-030657-6, 0-32-108599-X, and 0-471-58719-2. The hyphens separate the characters into four blocks. The first block usually consists of a single digit and identifies the language (0 for English, 2 for French, 3 for German, etc.). The second block identifies the publisher (for example, 13 for Prentice Hall, 32 for Addison-Wesley-Longman, and 471 for Wiley); The third block is the number the publisher has chosen for the book. The fourth block, which always consists of a single character called the check digit, is used to test for errors. Let's refer to the ten characters of the ISBN as d1,d2,d3,d4,d5,d6,d7,d8,d9,and d10.The check digit is chosen so that the sum

[Page 408]

$$10 \cdot d_1 + 9 \cdot d_2 + 8 \cdot d_3 + 7 \cdot d_4 + 6 \cdot d_5 + 5 \cdot d_6 + 4 \cdot d_7 + 3 \cdot d_8 + 2 \cdot d_9 + 1 \cdot d_{10} \qquad (*)$$

is a multiple of 11. Note: A number is multiple of 11 if it is exactly divisible by 11.). If the last character of the ISBN is an X, then in the sum(*),d10 is replaced with 10. For example, with the ISBN 0-32-108599-X, the sum would be

$$10 \cdot 0 + 9 \cdot 3 + 8 \cdot 2 + 7 \cdot 1 + 6 \cdot 0 + 5 \cdot 8 + 4 \cdot 5 + 3 \cdot 9 + 2 \cdot 9 + 1 \cdot 10 = 165$$

Since 165/11 is 15, the sum is a multiple of 11. This checking scheme will detect every single-digit and transposition-of-adjacent-digits error. That is, if while coping an IBSN number you miscopy a single character or transpose two adjacent characters, then the sum(*) will no longer be a multiple of 11.

a. Write a program to accept an ISBN type number (including the hyphens) as input, calculate the sum(*), and tell if it is a valid ISBN number. (Hint: The number n is divisible by 11 if n Mod 11 is 0.) Before calculating the sum, the program should check that each of the first nine characters is a digit and that the last character is either a digit or an X.

b. Write a program that begins with a valid ISBN number (such as 0-13-030657-6) and then confirms that the checking scheme described above detects every single-digit and transposition-of-adjacent-digits error by testing every possible error. [Hint: If d is a digit to be replaced, then the nine possibilities for the replacements are (d+1) Mod 10, (d+2) Mod 10, (d+3) Mod 10, Mod 10....(d+9) mod 10.].

10. User-Operated Directory Assistance. Have you ever tried to call a person at their business and been told to type in some of the letters of their name on your telephone's keypad in order to obtain their extension? Write a program to simulate this type of directory assistance. Suppose the names and telephone extensions of all the employees of a company are contained in the text file EMPLOYEES.TXT. Each set of three lines of the file has the three pieces of information: last name, first and middle name(s), and telephone extension. (We have filled the file with the names of the U.S. Presidents so that the names will be familiar.) The user should be asked to press buttons for the first three letters of the person's last name followed by the first letter of the first name. For instance, if the person's name is Gary Land, the user would type in 5264. The number 5264 is referred to as the "push-button encoding" of the name. (Note: People with different names can have the same push-button encoding for instance, Herb James and Gary Land.)

The program should declare an array of structures to hold full names, extensions, and push-button encodings. The arrays should be filled while making a single pass through the file and then sorted by push-button encoding. After the user presses four keys on the keypad, the program should display the names and extensions of all the employees having the specified push-button encoding. See Figure 7.15.

[Page 409]

Figure 7.15. Sample run of Programming Project 10.

[View full size image]