PREV NEXT

[Page 247]

Chapter 6. Repetition

(This item omitted from WebBook edition)

PREV PREV NEXT NEXT

[Page 248]

6.1. Do Loops

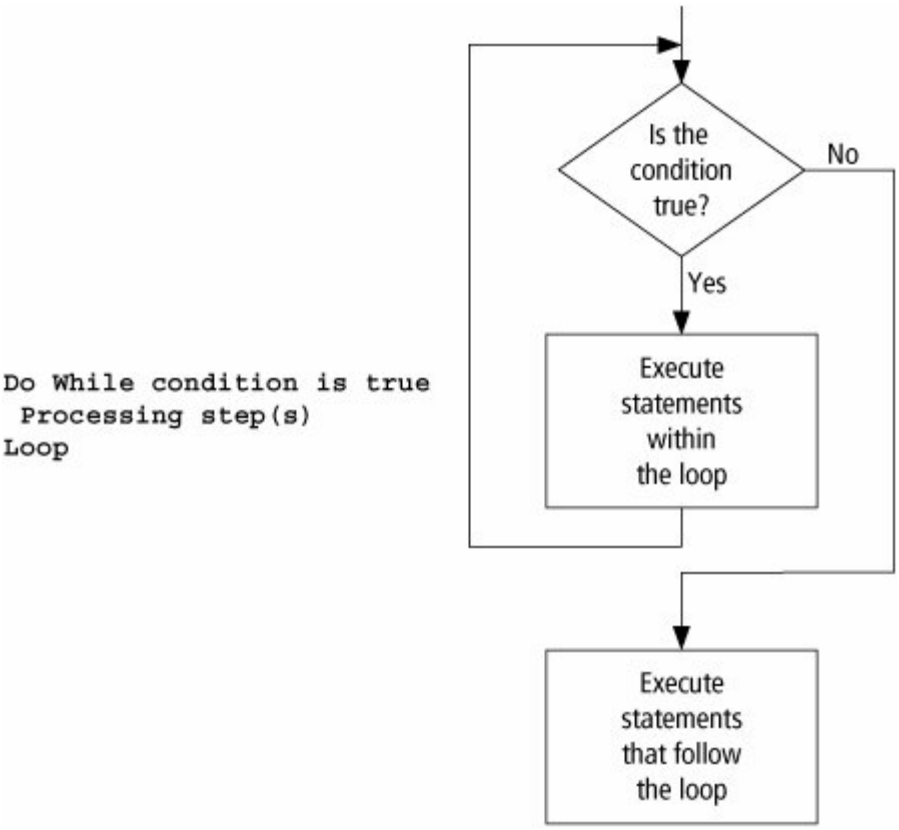
A loop, one of the most important structures in Visual Basic, is used to repeat a sequence of statements a number of times. At each repetition, or pass, the statements act upon variables whose values are changing.

The Do loop repeats a sequence of statements either as long as or until a certain condition is true. A Do statement precedes the sequence of statements, and a Loop statement follows the sequence of statements. The condition, preceded by either the word "While" or the word "Until", follows the word "Do" or the word "Loop". When Visual Basic executes a Do loop of the form

```
Do While condition
  statement(s)
Loop
```

it first checks the truth value of condition. If condition is false, then the statements inside the loop are not executed, and the program continues with the line after the Loop statement. If condition is true, then the statements inside the loop are executed. When the statement Loop is encountered, the entire process is repeated, beginning with the testing of condition in the Do While statement. In other words, the statements inside the loop are repeatedly executed only as long as (that is, while) the condition is true. Figure 6.1 contains the pseudocode and flowchart for this loop.

Figure 6.1. Pseudocode and flowchart for a Do While loop.



[Page 249]

Example 1.

The following program, in which the condition in the Do loop is "num <=7", displays the

numbers from 1 through 7. (After the Do loop executes, the value of num will be 8.)

```
Private Sub btnDisplay_Click(...) Handles btnDisplay.Click
  'Display the numbers from 1 to 7
  Dim num AsInteger = 1
  DoWhile num <= 7
    lstNumbers.Items.Add(num)
    num += 1     'Add 1 to the value of num
  Loop
End Sub
```

[Run, and click the button. The following is displayed in the list box.]

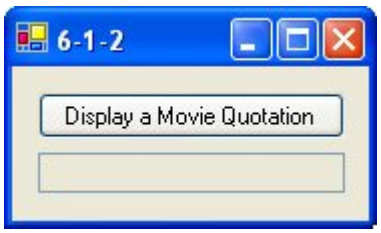
```
1
2
3
4
5
6
7
```

Do loops are commonly used to ensure that a proper response is received from the InputBox function.

Example 2.

(This item is displayed on pages 249 - 250 in the print version)

The following program requires the user to enter a number from 1 through 3. The Do loop repeats the request until the user gives a proper response.



Object	Property	Setting
frmMovie	Text	6-1-2
btnDisplay	Text	Display a Movie Quotation
txtQuotation	ReadOnly	True

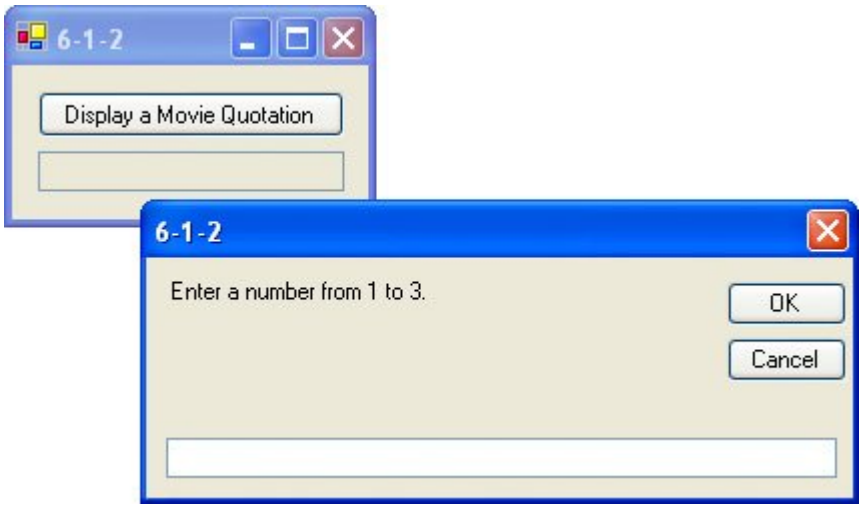
```
Private Sub btnDisplay_Click(...) Handles btnDisplay.Click
  Dim response As Integer, quotation As String = ""
  response = CInt(InputBox("Enter a number from 1 to 3."))
  Do While (response < 1) Or (response > 3)
    response = CInt(InputBox("Enter a number from 1 to 3."))
```

```
  Loop
  Select Case response
    Case 1
      quotation = "Plastics."
    Case 2
      quotation = "Rosebud."
```

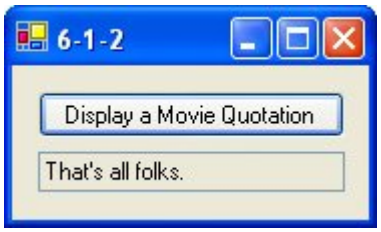


```
    Case 3
      quotation = "That's all folks."
  End Select
  txtQuotation.Text = quotation
End Sub
```

[Run, and click the button.]



[Type 3 into the box and press the OK button.]



In Examples 1 and 2, the condition was checked at the top of the loopthat is, before the statements were executed. Alternatively, the condition can be checked at the bottom of the loop when the statement Loop is reached. When Visual Basic encounters a Do loop of the form

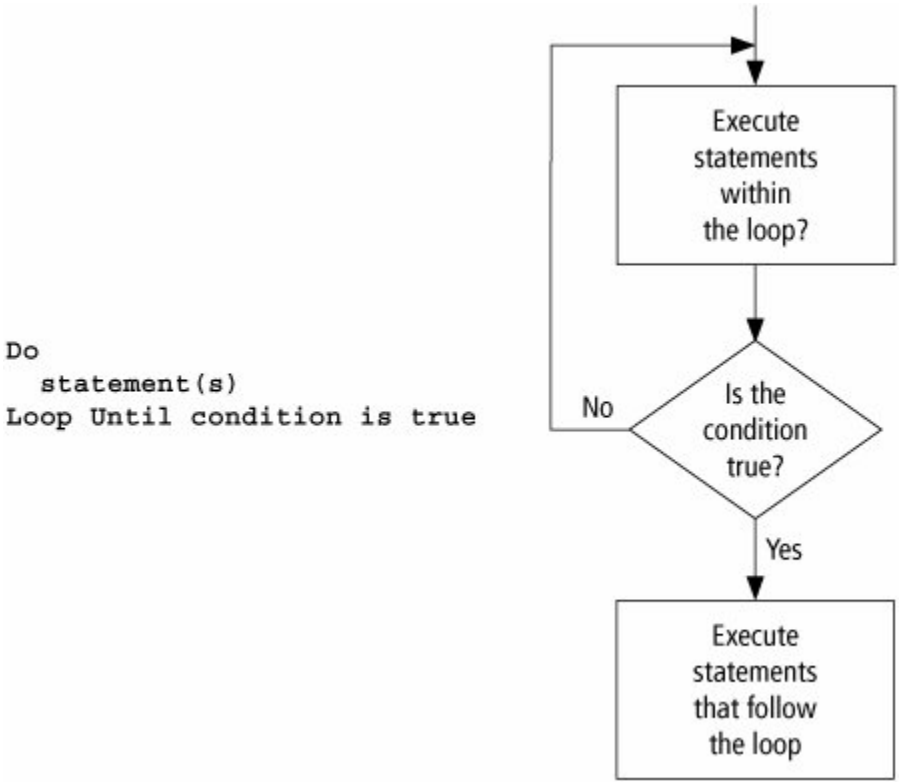
```
Do
  statement(s)
Loop Until condition
```

it executes the statements inside the loop and then checks the truth value of condition. If condition is true, then the program continues with the line after the Loop statement. If condition is false, then the

entire process is repeated beginning with the Do statement. In other words, the statements inside the loop are executed once and then are repeatedly executed until the condition is true. Figure 6.2 shows the pseudocode and flowchart for this type of Do loop.

[Page 251]

Figure 6.2. Pseudocode and flowchart for a Do loop with condition tested at the bottom.



Example 3.

The following program is equivalent to Example 2, except that the condition is tested at the bottom of the loop:

```
Private Sub btnDisplay_Click(...) Handles btnDisplay.Click
  Dim response As Integer, quotation As String = ""
  Do
    response = CInt(InputBox("Enter a number from 1 to 3."))
  Loop Until (response >= 1) And (response <= 3)
  Select Case response
    Case 1
      quotation = "Plastics."
    Case 2
      quotation = "Rosebud."
    Case 3
      quotation = "That's all folks."
  End Select
  txtQuotation.Text = quotation
End Sub
```

Do loops allow us to calculate useful quantities for which we might not know a simple formula.

Example 4.

(This item is displayed on pages 251 - 252 in the print version)

Suppose you deposit money into a savings account and let it accumulate at 6 percent interest compounded annually. The following program determines when you will be a millionaire:

[Page 252]

Object	Property	Setting
frmMillionaire	Text	6% Interest
lblAmount	Text	Amount Deposited:
txtAmount		
btnCalculate	Text	Calculate Years to Become a Millionaire
txtWhen	ReadOnly	True

```
Private Sub btnYears_Click(...) Handles btnYears.Click
  'Compute years required to become a millionaire
  Dim balance AsDouble, numYears As Integer
  balance = CDbl(txtAmount.Text)
  DoWhile balance < 1000000
    balance += 0.06 * balance
    numYears += 1
  Loop
  txtWhen.Text = "In " & numYears & _
                 " years you will have a million dollars."
End Sub
```

[Run, type 100000 into the text box, and press the button.]

Comments

1. Be careful to avoid infinite loopsthat is, loops that are never exited. The following loop is infinite, because the condition "balance < 1000" will always be true.

   Note: Click on the form's Close button at the upper right corner of the title bar to end the program.

```
Private Sub btnButton_Click(...) Handles btnlick
  'An infinite loop
  Dim balance As Double = 100, intRate As Double
  Do While balance < 1000
    balance = (1 + intRate) * balance
  Loop
  txtBalance.Text = FormatCurrency(balance)
End Sub
```

   Notice that this slip-up can be avoided by adding something like `= 0.04` to the end of the Dim statement.

[Page 253]

2. Visual Basic allows the use of the words "While" and "Until" at either the top or bottom of a Do loop. In this text, the usage of these words is restricted for the following reasons:

   a. Because any While statement can be easily converted to an Until statement and vice versa, the restriction produces no loss of capabilities and the programmer has one less matter to think about.

   b. Restricting the use simplifies reading the program. The word "While" proclaims testing at the top, and the word "Until" proclaims testing at the bottom.

   c. Certain other major structured languages only allow "While" at the top and "Until" at the bottom of a loop. Therefore, following this convention will make life easier for people already familiar with or planning to learn one of these languages.

   d. Standard pseudocode uses the word "While" to denote testing a loop at the top and the word "Until" to denote testing at the bottom.

Practice Problems 6.1

1. How do you decide whether a condition should be checked at the top of a loop or at the

bottom?

2. Change the following code segment so it will be executed at least once:

```
Do While continue = "Yes"
  answer = InputBox("Do you want to continue? (Y or N)")
  If answer.ToUpper = "Y" Then
    continue = "Yes"
  Else
    continue = "No"
  End If
Loop
```

Exercises 6.1

In Exercises 1 through 6, determine the output displayed when the button is clicked.

1.
```
Private Sub btnDisplay_Click(...) Handles btnDisplay.Click
  Dim q As Double
  q = 3
  Do While q < 15
    q = 2 * q - 1
  Loop
  txtOutput.Text = CStr(q)
End Sub
```

[Page 254]

2.
```
Private Sub btnDisplay_Click(...) Handles btnDisplay.Click
  Dim balance As Double = 1000
  Dim interest As Double = 0.1
  Dim n As Integer = 0         'Number of years
  Do
    lstOutput.Items.Add(n & " " & balance)
    balance = (1 + interest) * balance
    n += 1
  Loop Until (balance > 1200)
  lstOutput.Items.Add(n)
End Sub
```

3.
```
Private Sub btnDisplay_Click(...) Handles btnDisplay.Click
  'Display a message
  Dim num As Double = 4
  Dim message As String = ""
  Do
    Select Case num
```

```
      Case 1
        message = "grammer!"
        num = -1
      Case 2
        message = "re a su"
        num = (5 - num) * (3 - num)
      Case 3
        message = "per pro"
        num = 4 - num
      Case 4
        message = "You a"
        num = 2 * num - 6
    End Select
    txtOutput.Text &= message
  Loop Until (num = -1)
End Sub
```

4.

```
Private Sub btnQuiz_Click(...) Handles btnQuiz.Click
  Dim prompt As String, firstYear As Integer
  'Computer-assisted instruction
  prompt = "In what year was the IBM PC first produced?"
  lstQuiz.Items.Clear()
  Do
    firstYear = CInt(InputBox(prompt))
    Select Case firstYear
      Case 1981
        lstQuiz.Items.Add("Correct. The computer was an instant")
        lstQuiz.Items.Add("success. By the end of 1981, there")
        lstQuiz.Items.Add("was such a backlog of orders that buyers")
        lstQuiz.Items.Add("had a three-month waiting period.")
```

[Page 255]

```
      Case Is < 1981
        lstQuiz.Items.Add("Later. The Apple II, which preceded")
        lstQuiz.Items.Add("the IBM PC, appeared in 1977.")
      Case Is > 1981
        lstQuiz.Items.Add("Earlier. The first successful IBM PC")
        lstQuiz.Items.Add("clone, the Compaq Portable, ")
        lstQuiz.Items.Add("appeared in 1983.")
    End Select
    lstQuiz.Items.Add("")
  Loop Until firstYear = 1981
End Sub
```

(Assume that the first response is 1980 and the second response is 1981.)

5.

```
Private Sub btnDisplay_Click(...) Handles btnDisplay.Click
  'Calculate the remainder in long division
  txtOutput.Text = CStr(Remainder(3, 17))
End Sub
Function Remainder(ByVal divisor As Double, _
                   ByVal dividend As Double) As Double
  Dim sum As Double = 0
  Do While sum <= dividend
```

```
    sum += divisor
  Loop
  Return (dividend - sum + divisor)
End Function
```

6.
```
Private Sub btnDisplay_Click(...) Handles btnDisplay.Click
  'Simulate IndexOf; search for the letter t
  Dim info As String = "Potato"
  Dim letter As String = "t"
  Dim answer As Integer
  answer = IndexOf(info, letter)
  txtOutput.Text = CStr(answer)
End Sub
Function IndexOf(ByVal word As String, ByVal letter As String) _
             As Integer
  Dim counter As Integer = 0
  Dim current As String = ""
  Do While counter < word.Length
    current = word.Substring(counter, 1)
    If letter = current Then
       Return counter
    End If
    counter += 1 'Add 1 to the value of counter
  Loop
  'If not found then the answer is -1
  Return -1
End Function
```

[Page 256]

In Exercises 7 through 10, identify the errors.

7.
```
Private Sub btnDisplay_Click(...) Handles btnDisplay.Click
  Dim q As Double
  q = 1
  Do While q > 0
    q = 3 * q - 1
    lstOutput.Items.Add(q)
  Loop
End Sub
```

8.
```
Private Sub btnDisplay_Click(...) Handles btnDisplay.Click
  'Display the numbers from 1 to 5
  Dim num As Integer
  Do While num <> 6
    num = 1
```

```
    lstOutput.Items.Add(num)
    num += 1
  Loop
End Sub
```

9.
```
Private Sub btnDisplay_Click(...) Handles btnDisplay.Click
  'Repeat until a yes response is given
  Dim answer As String = "N"
  Loop
    answer = InputBox("Did you chop down the cherry tree (Y/N)?")
  Do Until (answer.ToUpper = "Y")
End Sub
```

10.
```
Private Sub btnDisplay_Click(...) Handles btnDisplay.Click
  'Repeat as long as desired
  Dim n As Integer, answer As String = ""
  Do
    n += 1
    lstOutput.Items.Add(n)
    answer = InputBox("Do you want to continue (Y/N)?")
  Until answer.ToUpper = "N"
End Sub
```

In Exercises 11 through 20, replace each phrase containing "Until" with an equivalent phrase containing "While", and vice versa. For instance, the phrase (Until sum = 100) would be replaced by (While sum <> 100).

11. `Until num < 7`

12. `Until name = "Bob"`

13. `While response = "Y"`

14. `While total = 10`

15. `While name <> ""`

16. `Until balance >= 100`

[Page 257]

17. `While (a > 1) And (a < 3)`

18. `Until (ans = "") Or (n = 0)`

19. `Until Not (n = 0)`

20. `While (ans = "Y") And (n < 7)`

In Exercises 21 and 22, write simpler and clearer code that performs the same task as the given code.

21.
```
Private Sub btnDisplay_Click(...) Handles btnDisplay.Click
  Dim name As String
  name = InputBox("Enter a name:")
  lstOutput.Items.Add(name)
  name = InputBox("Enter a name:")
  lstOutput.Items.Add(name)
  name = InputBox("Enter a name:")
  lstOutput.Items.Add(name)
End Sub
```

22.
```
Private Sub btnDisplay_Click(...) Handles btnDisplay.Click
  Dim loopNum As Integer, answer As String = ""
  Do
    If loopNum >= 1 Then
      answer = InputBox("Do you want to continue (Y/N)?")
      answer = answer.ToUpper
    Else
      answer = "Y"
    End If
    If (answer = "Y") Or (loopNum = 0) Then
      loopNum += 1
      txtOutput.Text = CStr(loopNum)
    End If
  Loop Until (answer <> "Y")
End Sub
```

23. Write a program that displays a Celsius-to-Fahrenheit conversion table. Entries in the table should range from -40 to 40 degrees Celsius in increments of 5 degrees. Note: The formula f = (9/5)*c + 32 converts Celsius to Fahrenheit.

24. The world population doubled from 3 billion in 1959 to 6 billion in 1999. Assuming that the world population has been doubling every 40 years, write a program to determine when in the past the world population was less than 6 million.

25. The world population reached 6.5 billion people in 2006 and was growing at the rate of 1.2 percent each year. Assuming that the population will continue to grow at the same rate, write a program to determine when the population will reach 10 billion.

26. Write a program to display all the numbers between 1 and 100 that are perfect squares. (A perfect square is an integer that is the square of another integer; 1, 4, 9, and 16 are examples of perfect squares.)

[Page 258]

27. Write a program to display all the numbers between 1 and 100 that are part of the Fibonacci sequence. The Fibonacci sequence begins 1, 1, 2, 3, 5, 8,..., where each new number in the sequence is found by adding the previous two numbers in the sequence.

28. Write a program to request positive numbers one at a time from the user in an input dialog box. The user should be instructed to enter -1 after all the positive numbers have been supplied. At that time, the program should display the sum of the numbers.

29. An old grandfather clock reads 6:00 p.m. Sometime not too long after 6:30 p.m., the minute hand will pass directly over the hour hand. Write a program using a loop to make better and better guesses as to what time it is when the hands exactly overlap. Keep track of the positions of both hands using the minutes at which they are pointing. (At 6:00 p.m., the minute hand points at 0 while the hour hand points at 30.) You will need to use the fact that when the minute hand advances m minutes, the hour hand advances m/12 minutes. (For example, when the minute hand advances 60 minutes, the hour hand advances 5 minutes from one hour mark to the next.) To make an approximation, record how far the minute hand is behind the hour hand, then advance the minute hand by this much and the hour hand by 1/12 this much. The loop should terminate when the resulting positions of the minute and hour hands differ by less than .0001 minute. (The exact answer is 32 and 8/11 minutes after 6.)

30. Write a program that requests a word containing the two letters r and n as input and determines which of the two letters appears first. If the word does not contain both of the letters, the program should so advise the user. (Test the program with the words "colonel" and "merriment.")

31. The coefficient of restitution of a ball, a number between 0 and 1, specifies how much energy is conserved when a ball hits a rigid surface. A coefficient of .9, for instance, means a bouncing ball will rise to 90 percent of its previous height after each bounce. Write a program to input a coefficient of restitution and an initial height in meters, and report how many times a ball bounces when dropped from its initial height before it rises to a height of less than 10 centimeters. Also report the total distance traveled by the ball before this point. The coefficients of restitution of a tennis ball, basketball, super ball, and softball are .7, .75, .9, and .3, respectively.

In Exercises 32 through 35, write a program to solve the stated problem.

32. Savings Account. $15,000 is deposited into a savings account paying 5 percent interest and $1000 is withdrawn from the account at the end of each year. Approximately how many years are required for the savings account to be depleted? Note: If at the end of a certain year the balance is $1000 or less, then the final withdrawal will consist of that balance and the account will be depleted.

33. Rework Exercise 32 so that the amount of money deposited initially is input by the user and the program computes the number of years required to deplete the account.

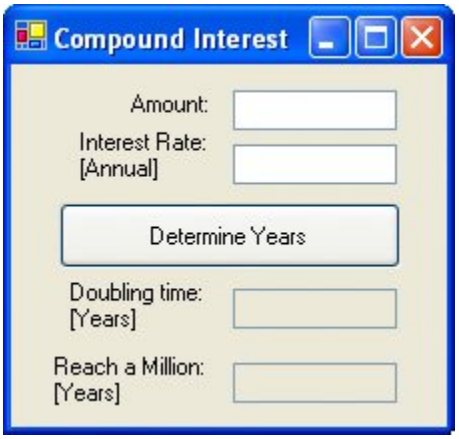
Note: Be careful to avoid an infinite loop.

[Page 259]

34. Consider an account in which $1000 is deposited upon opening the account and an additional $1000 is deposited at the end of each year. If the money earns interest at the rate of 5 percent, how long will it take before the account contains at least $1 million?

35. A person born in 1980 can claim, "I will be x years old in the year x squared." Write a program to determine the value of x.

36. Illustrate the growth of money in a savings account. When the user presses the button, values for Amount and Interest Rate are obtained from text boxes and used to calculate the number of years until the money doubles and the number of years until the money reaches a million dollars. Use the form design shown below. Note: The balance at the end of each year is (1 + r) times the previous balance, where r is the annual rate of interest in decimal form.



Object	Property	Setting
frmInterest	Text	Compound Interest
lblAmount	Text	Amount:
txtAmount		
lblRate	AutoSize	False
	Text	Interest Rate: [Annual]
txtRate		
btnDetermine	Text	Determine Years
lblDouble	AutoSize	False
	Text	Doubling Time: [Years]

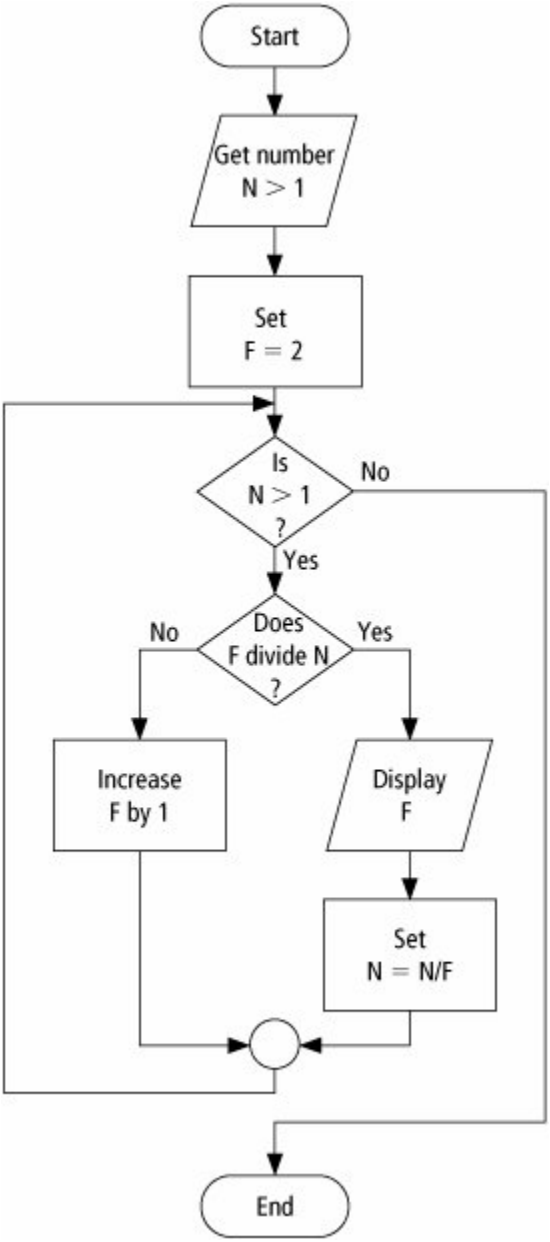
txtDouble	ReadOnly	True
lblMillion	AutoSize	False
	Text	Reach a Million: [Years]
txtMillion	ReadOnly	True

37. Allow the user to enter a sentence. Then, depending on which button the user clicks, display the sentence entirely in capital letters or with just the first letter of each word capitalized.

In Exercises 38 and 39, write a program corresponding to the flowchart.

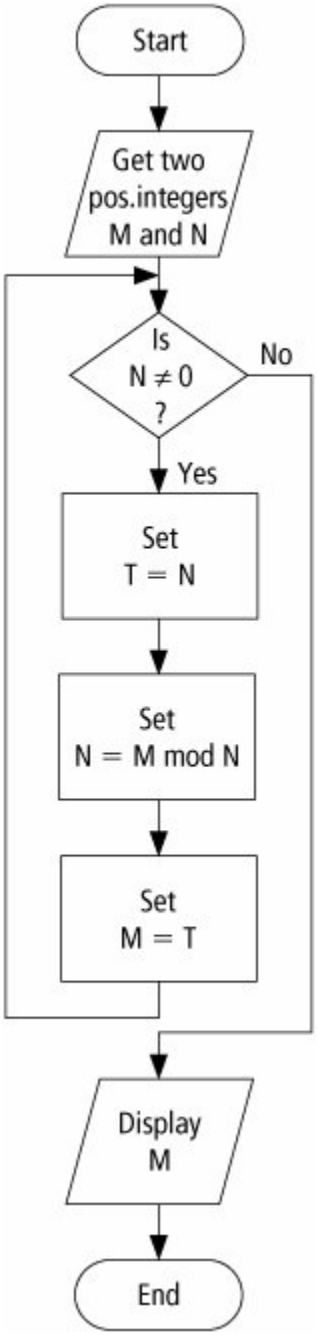
38. The flowchart in Figure 6.3 (on the next page) requests a whole number greater than 1 as input and factors it into a product of prime numbers. Note: A number is prime if its only factors are 1 and itself. Test the program with the numbers 660 and 139.

Figure 6.3. Prime factors.
(This item is displayed on page 260 in the print version)



39. The flowchart in Figure 6.4 (on page 261) finds the greatest common divisor (the largest integer that divides both) of two positive integers input by the user. Write a program that corresponds to the flowchart. Use the Visual Basic operator Mod. The value of m Mod n is the remainder when m is divided by n.

Figure 6.4. Greatest common divisor.
(This item is displayed on page 261 in the print version)



[Page 260]

Solutions to Practice Problems 6.1

1. As a rule of thumb, the condition is checked at the bottom if the loop should be executed at least once.

2. Either precede the loop with the statement `continue = "Yes"`, or change the first line to `Do` and replace the Loop statement with `Loop Until continue <> "Yes"`.

PREV PREV NEXT NEXT

[Page 261]

6.2. Processing Lists of Data with Do Loops

One of the main applications of programming is the processing of lists of data. Do loops are used to display all or selected items from lists, search lists for specific items, and perform calculations on the numerical entries of a list. This section introduces several devices that facilitate working with lists. Counters calculate the number of elements in lists, accumulators sum numerical values in lists, flags record whether certain events have occurred, and the Peek method can be used to determine when the end of a text file has been reached. Nested loops add yet another dimension to repetition.

Peek Method

Data to be processed are often retrieved from a file by a Do loop. Visual Basic has a useful method that will tell us if we have reached the end of the file from which we are reading. Suppose a file has been opened with a StreamReader object named sr. At any time, the value of

---

[Page 262]

```
sr.Peek
```

is the ANSI value of the first character of the line about to be read with ReadLine. If the end of the file has been reached, the value of `sr.Peek` is 1.

One of the programs I wrote when I got my first personal computer in 1982 stored a list of names and phone numbers and displayed a phone directory. I stored the names in a file so I could easily add, change, or delete entries.

Example 1.

(This item is displayed on pages 262 - 263 in the print version)

The following program displays the contents of a telephone directory. The names and phone numbers are contained in the file PHONE.TXT. The loop will repeat as long as there is still data in the file.

PHONE.TXT contains the following eight lines:

Bert

123-4567

Ernie

987-6543

Grover

246-8321

Oscar

135-7900

Object	Property	Setting
frmPhone	Text	Directory
btnDisplay	Text	Display Phone Numbers
lstNumbers		

```
Private Sub btnDisplay_Click(...) Handles btnDisplay.Click
  Dim name, phoneNum AsString
  Dim sr As IO.StreamReader = IO.File.OpenText("PHONE.TXT")
  lstNumbers.Items.Clear()
  DoWhile sr.Peek <> -1
    name = sr.ReadLine
    phoneNum = sr.ReadLine
    lstNumbers.Items.Add(name & "    " & phoneNum)
  Loop
  sr.Close()
End Sub
```

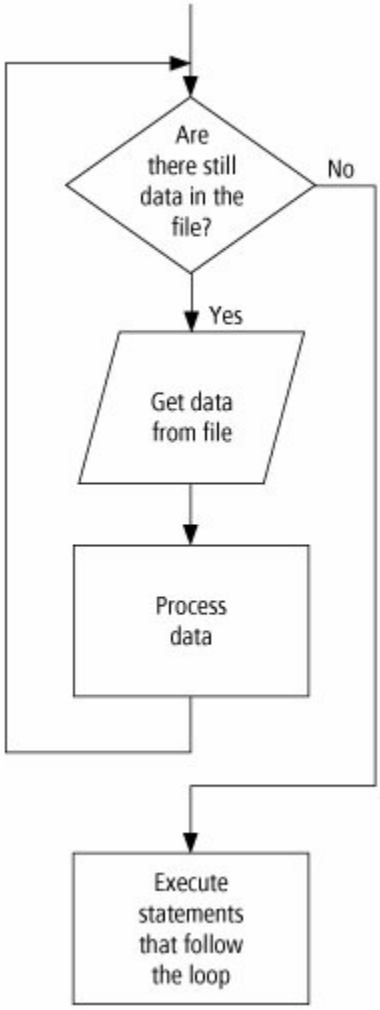
[Page 263]

[Run, and press the button.]

The program in Example 1 illustrates the proper way to process a list of data contained in a file. The Do loop should be tested at the top with an end-of-file condition. (If the file is empty, no attempt is made to input data from the file.) The first set of data should be input after the Do statement, and then the data should be processed. Figure 6.5 contains the pseudocode and flowchart for this technique.

Figure 6.5. Pseudocode and flowchart for processing data from a file.

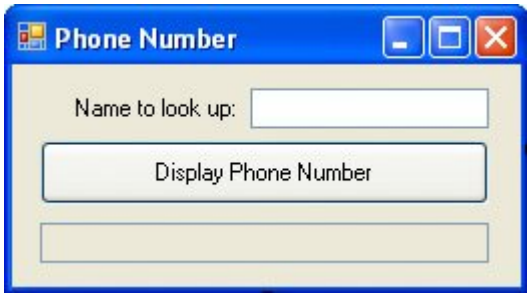
```
Do while there are still data in the file
  Get an item of data
  Process the item
Loop
```



Text files can be quite large. Rather than list the entire contents, we typically search the file for a specific piece of information.

Example 2.

The following program modifies the program in Example 1 to search the telephone directory for a name specified by the user. If the name does not appear in the directory, the user is so notified. We want to keep searching as long as there is no match and we have not reached the end of the list. Therefore, the condition for the Do While statement is a compound logical expression with the operator And. After the last pass through the loop, we will know whether the name was found and will be able to display the requested information.



Object	Property	Setting
frmPhone	Text	Phone Number
lblName	Text	Name to look up:
txtName		
btnDisplay	Text	Display Phone Number
txtNumber	ReadOnly	True

```
Private Sub btnDisplay_Click(...) Handles btnDisplay.Click
  Dim name As String = "",  phoneNum AsString = ""
  Dim sr As IO.StreamReader = IO.File.OpenText("PHONE.TXT")
  DoWhile (name <> txtName.Text) And (sr.Peek <> -1)
    name = sr.ReadLine
    phoneNum = sr.ReadLine
  Loop
  If (name = txtName.Text) Then
    txtNumber.Text = name & "    " & phoneNum
  Else
    txtNumber.Text = "Name not found."
  EndIf
  sr.Close()
End Sub
```

[Run, type "Grover" into the text box, and press the button.]

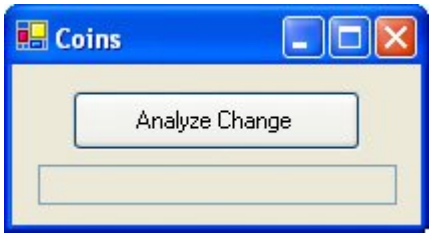
[Page 265]

Counters and Accumulators

A counter is a numeric variable that keeps track of the number of items that have been processed. An accumulator is a numeric variable that totals numbers.

Example 3.

The following program counts and finds the value of coins listed in a file. The file COINS.TXT contains the following entries, one per line: 1, 1, 5, 10, 10, 25. The fifth and sixth lines of the event procedure are not needed, since the counter numCoins and the accumulator sum have the initial value 0 by default. However, since these default values show the starting points for the two variables, the two assignment statements add clarity to the program.

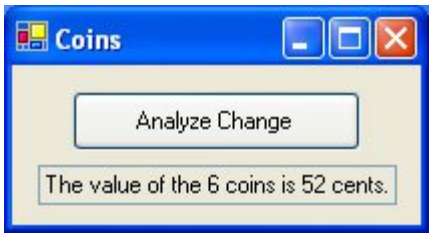


Object	Property	Setting
frmCoins	Text	Coins
btnAnalyze	Text	Analyze Change
txtValue	ReadOnly	True

```
Private Sub btnAnalyze_Click(...) Handles btnAnalyze.Click
  Dim numCoins AsInteger, coin As String
  Dim sum As Double
  Dim sr As IO.StreamReader = IO.File.OpenText("COINS.TXT")
  numCoins = 0
  sum = 0
  DoWhile (sr.Peek <> -1)
    coin = sr.ReadLine
    numCoins += 1       'Add 1 to the value of numCoins
```

```
    sum += CDbl(coin)   'Add the value of the current coin to the sum
  Loop
  sr.Close()
  txtValue.Text = "The value of the " & numCoins & _
                  " coins is " & sum & " cents."
End Sub
```

[Run, and press the button.]



[Page 266]

The value of the counter, numCoins, was initially 0 and changed on each execution of the loop to 1, 2, 3, 4, 5, and finally 6. The accumulator, sum, initially had the value 0 and increased with each execution of the loop to 1, 2, 7, 17, 27, and finally 52.

Flags

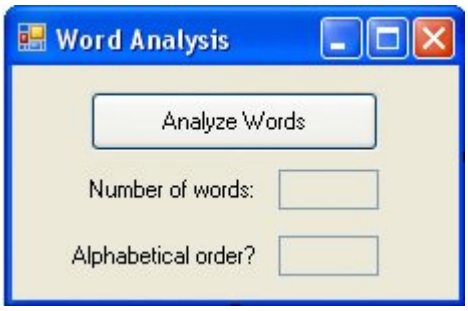
A flag is a variable that keeps track of whether a certain situation has occurred. The data type most suited to flags is the Boolean data type. Variables of type Boolean can assume just two values: True and False. (As a default, Boolean variables are initialized to False.) Flags are used within loops to provide information that will be utilized after the loop terminates. Flags also provide an alternative method of terminating a loop.

Example 4.

(This item is displayed on pages 266 - 267 in the print version)

The following program counts the number of words in the file WORDS.TXT and then reports whether the words are in alphabetical order. In each execution of the loop, a word is compared to the next word in the list. The flag variable, called orderFlag, is initially assigned the value True and is set to False if a pair of adjacent words is out of order. The technique used in this program will be used in Chapter 7 when we study sorting. Note: The first time through the loop, the value of word1 is the empty string. Each word must first be read into the variable word2.

WORDS.TXT contains the following winning words from the U.S. National Spelling Bee, one word per line: cambist, croissant, deification, hydrophyte, incisor, maculature, macerate, narcolepsy, shallon.



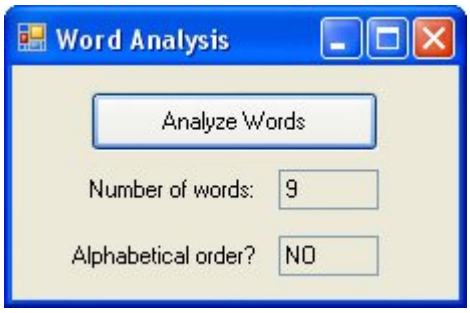
Object	Property	Setting
frmWords	Text	Word Analysis
btnAnalyze	Text	Analyze Words
lblNumber	Text	Number of words:
txtNumber	ReadOnly	True
lblAlpha	Text	Alphabetical order?
txtAlpha	ReadOnly	True

```
Private Sub btnAnalyze_Click(...) Handles btnAnalyze.Click
  'Count words.  Are they in alphabetical order?
  Dim orderFlag As Boolean, wordCounter As Integer
  Dim word1 As String = "", word2 As String
  Dim sr As IO.StreamReader = IO.File.OpenText("WORDS.TXT")
  orderFlag = True
  DoWhile (sr.Peek <> -1)
    word2 = sr.ReadLine
    wordCounter += 1          'Increment the word count by 1
    If word1 > word2 Then     'Two words are out of order
      orderFlag = False
    EndIf
    word1 = word2
  Loop
  sr.Close()
  txtNumber.Text = CStr(wordCounter)
```

[Page 267]

```
  If orderFlag = True Then
    txtAlpha.Text = "YES"
  Else
    txtAlpha.Text = "NO"
  End If
End Sub
```

[Run, and press the button.]



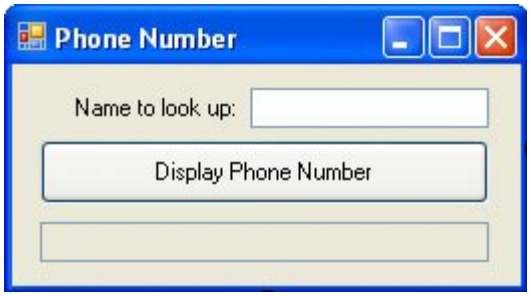
Nested Loops

The statements within a Do loop can contain another Do loop. Such a configuration is referred to as nested loops and is useful in repeating a single data-processing routine several times.

Example 5.

(This item is displayed on pages 267 - 268 in the print version)

The following program allows the user to look through several lists of names. Suppose we have several different phone directories, the names of which are listed in the file LISTS.TXT. (For instance, the file LISTS.TXT might contain the entries CLIENTS.TXT, FRIENDS.TXT, and KINFOLK.TXT.) A sought-after name might be in any one of the files. The statements in the inner Do loop will be used to look up names as before. At least one pass through the outer Do loop is guaranteed, and passes will continue as long as the name is not found and phone lists remain to be examined.



OBJECT	PROPERTY	SETTING
frmPhone	Text	Phone Number
lblName	Text	Name to look up:
txtName		
btnDisplay	Text	Display Phone Number
txtNumber	ReadOnly	True

```
Private Sub btnDisplay_Click(...) Handles btnDisplay.Click
  Dim foundFlag As Boolean
```

```
  Dim fileName As String
  Dim name As String = ""
  Dim phoneNum As String = ""
  Dim sr1 As IO.StreamReader = IO.File.OpenText ("LISTS.TXT")
```

[Page 268]

```
  'The next line is not needed. It is added for clarity.
  foundFlag = False
  DoWhile (foundFlag = False) And (sr1.Peek <> -1)
    fileName = sr1.ReadLine
    Dim sr2 As IO.StreamReader = IO.File.OpenText(fileName)
    DoWhile (name <> txtName.Text) And (sr2.Peek <> -1)
      name = sr2.ReadLine
      phoneNum = sr2.ReadLine
    Loop
    sr2.Close()
    If name = txtName.Text Then
      txtNumber.Text = name & "      " & phoneNum
      foundFlag = True
   End If
  Loop
  sr1.Close()
  If foundFlag = False Then
   txtNumber.Text = "Name not found."
  End If
End Sub
```

Comments

1. In Appendix D, the section "Stepping through a Program Containing a Do Loop: Chapter 6" uses the Visual Basic debugging tools to trace the flow through a Do loop.

2. When flagVar is a variable of Boolean type, the statements

   `If flagVar = True Then` and `If flagVar = False Then`

   can be replaced by

   `If flagVar Then` and `If Not flagVar Then`

   Similarly, the statements

   `Do While flagVar = True` and `Do While flagVar = False`

   can be replaced by

   `Do While flagVar` and `Do While Not flagVar`

Practice Problems 6.2

1. Determine the output of the following program, where the three lines of the file SCORES.TXT contain the scores 150, 200, and 300:

```
Private Sub btnCompute_Click(...) Handles btnCompute.Click
  'Find the sum of a collection of bowling scores
  Dim sum, score As Double
```

[Page 269]

```
  Dim sr As IO.StreamReader = IO.File.OpenText("SCORES.TXT")
  sum = 0
  score = CDbl(sr.ReadLine)
  Do While (sr.Peek <> -1)
    sum += score   'Add value of score to value of sum
    score = CDbl(sr.ReadLine)
  Loop
  sr.Close()
  txtTotal.Text = CStr(sum)
End Sub
```

2. Why didn't the preceding program produce the intended output?

3. Correct the preceding program so it has the intended output.

Exercises 6.2

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

1.
```
Private Sub btnDisplay_Click(...) Handles btnDisplay.Click
  Dim flag As Boolean, word As String
  Dim sr As IO.StreamReader = IO.File.OpenText("DATA.TXT")
  Do While (sr.Peek <> -1)
    word = sr.ReadLine
    If word.IndexOf("A") <> -1 Then
      flag = True
    End If
  Loop
  sr.Close()
  If flag Then
    txtOutput.Text = "At least one word contains the letter A."
  Else
    txtOutput.Text = "No word contains the letter A."
  End If
End Sub
```

(Assume that the four lines of the file DATA.TXT have the following entries: AL, GORE, VIDAL, SASSOON.)

2.
```
Private Sub btnDisplay_Click(...) Handles btnDisplay.Click
  Dim name As String
```

```
  Dim sr As IO.StreamReader = IO.File.OpenText("DATA.TXT")
  Do While (sr.Peek <> -1)
    name = sr.ReadLine
    txtOutput.Text &= name
  Loop
  sr.Close()
End Sub
```

(Assume that the two lines of the file DATA.TXT contain the following entries: Ben, son.)

[Page 270]

3.

```
Private Sub btnDisplay_Click(...) Handles btnDisplay.Click
  'Display list of desserts
  Dim dessert As String
  Dim sr As IO.StreamReader = IO.File.OpenText("DESSERTS.TXT")
  Do While (sr.Peek <> -1)
    dessert = sr.ReadLine
    lstOutput.Items.Add(dessert)
  Loop
  sr.Close()
End Sub
```

(Assume that the three lines of the file DESSERTS.TXT contain the following entries: pie, cake, melon.)

4.

```
Private Sub btnDisplay_Click(...) Handles btnDisplay.Click
  Dim city As String, pop As Double
  Dim sr As IO.StreamReader = IO.File.OpenText("CITYPOPS.TXT")
  Do While (sr.Peek <> -1)
    city = sr.ReadLine
    pop = CDbl(sr.ReadLine)
    If pop >= 3 Then
      lstOutput.Items.Add(city & "  "& pop)
    End If
  Loop
  sr.Close()
End Sub
```

(Assume that the eight lines of the file CITYPOPS.TXT contain the following data, where each pair of lines gives a city and its population in millions: New York, 8.1, Los Angeles, 3.8, Chicago, 2.9, Houston, 2.0.)

5.

```
Private Sub btnDisplay_Click(...) Handles btnDisplay.Click
  Dim firstLetter As String = "", fruit As String = ""
  Dim sr As IO.StreamReader = IO.File.OpenText("DATA.TXT")
  Do While (sr.Peek <> -1)
    fruit = sr.ReadLine
    If fruit.Substring(0, 1) <> firstLetter Then
      If firstLetter <> "" Then
```

```
        lstOutput.Items.Add("")
      End If
      firstLetter = fruit.Substring(0, 1)
      lstOutput.Items.Add("   "& firstLetter)
    End If
    lstOutput.Items.Add(fruit)
  Loop
  sr.Close()
End Sub
```

(Assume that the eight lines of the file FRUITS.TXT contain the following entries: Apple, Apricot, Avocado, Banana, Blueberry, Grape, Lemon, Lime.)

[Page 271]

6.

```
Private Sub btnDisplay_Click(...) Handles btnDisplay.Click
  'Display list of numbers
  Dim num As Double
  Dim sr As IO.StreamReader = IO.File.OpenText("DATA.TXT")
  num = CDbl(sr.ReadLine)
  Do While (sr.Peek <> -1)
    lstOutput.Items.Add(num)
    num = CDbl(sr.ReadLine)
  Loop
  sr.Close()
End Sub
```

(Assume the four lines of the file DATA.TXT contain the entries 2, 3, 8, 5.)

7.

```
Private Sub btnDisplay_Click(...) Handles btnDisplay.Click
  Dim animal As String = "", groupName As String = ""
  InputAnimal(animal)
  If animal <> "" Then
    SearchList(animal, groupName)
    DisplayResult(animal, groupName)
  End If
End Sub
Sub DisplayResult(ByVal anml As String, ByVal gName As String)
  If gName = "" Then
    txtOutput.Text = "Animal not found."
  Else
    txtOutput.Text = "A "& gName & " of "& anml & "s."
  End If
End Sub
Sub InputAnimal(ByRef anml As String)
  'Request the name of an animal as input
  anml = InputBox("Please enter the name of an animal.")
End Sub
Sub SearchList(ByVal anml As String, ByRef gName As String)
  Dim creature, groupName As String
  creature = ""
  Dim sr As IO.StreamReader = IO.File.OpenText("ANIMALS.TXT")
  Do While (creature <> anml) And (sr.Peek <> -1)
```

```
    creature = sr.ReadLine
    groupName = sr.ReadLine
  Loop
  If (sr.Peek = -1) Then
    gName = ""
  Else
    gName = groupName
  End If
  sr.Close()
End Sub
```

[Page 272]

(The six lines of the file ANIMALS.TXT contain the following entries: lion, pride, duck, brace, bee, swarm. Each pair of lines give an animal and the name of a group of those animals. Assume that the response is "duck".)

8.

```
Private Sub btnDisplay_Click(...) Handles btnDisplay.Click
  Dim excerpt As String
  excerpt = "I think I can. "
  Duplicate30(excerpt)
  excerpt = "We're off to see the wizard, " & _
            "the wonderful wizard of Oz."
  Duplicate30(excerpt)
End Sub
Sub Duplicate30(ByVal sentence As String)
  Dim flag As Boolean
  flag = False   'Flag tells whether loop has been executed
  Do While sentence.Length < 30
    flag = True
    sentence &= sentence
  Loop
  If flag = True Then
    lstOutput.Items.Add(sentence)
  Else
    lstOutput.Items.Add("Loop not executed.")
  End If
End Sub
```

9.

```
Private Sub btnDisplay_Click(...) Handles btnDisplay.Click
  Dim word As String = "", cWord As String = ""
  Dim sr As IO.StreamReader = IO.File.OpenText("WORDS.TXT")
  Do While (sr.Peek <> -1)
    word = sr.ReadLine
    If word.Substring(0, 1) = "c" Then
      cWord = word
    End If
  Loop
  sr.Close()
  txtOutput.Text = cWord
End Sub
```

(Assume that the 11 lines of the file WORDS.TXT contain the following items: time, is, a, child, idly, moving, counters, in, a, game, Heraclitus.)

10.
```
Private Sub btnDisplay_Click(...) Handles btnDisplay.Click
  Dim max, value, rowMax As Double
  Dim sr As IO.StreamReader = IO.File.OpenText("DATA.TXT")
  Do While (sr.Peek <> -1)
    value = CDbl(sr.ReadLine)
    rowMax = 0
```

```
    Do While value <> -2
      If value > rowMax Then
        rowMax = value
      End If
      value = CDbl(sr.ReadLine)
    Loop
    lstOutput.Items.Add(rowMax)
    If rowMax > max Then
      max = rowMax
    End If
  Loop
  sr.Close()
  lstOutput.Items.Add(max)
End Sub
```

(Assume that the 12 lines of the file DATA.TXT contain the following entries: 5, 7, 3, 2, 10, 12, 6, 4, 2, 1, 9, 2.)

In Exercises 11 through 14, identify the errors.

11.
```
Private Sub btnDisplay_Click(...) Handles btnDisplay.Click
  Dim num As Double
  Dim sr As IO.StreamReader = IO.File.OpenText("DATA.TXT")
  Do While (sr.Peek <> -1) And (num > 0)
    num = CDbl(sr.ReadLine)
    lstOutput.Items.Add(num)
    sr.Close()
End Sub
```

(Assume that the five lines of the file DATA.TXT contain the following entries: 7, 6, 0, 2.)

12.
```
Private Sub btnDisplay_Click(...) Handles btnDisplay.Click
  Dim flag As Boolean, num, sum As Double
  Do While flag = False
    num = CDbl(InputBox("Enter a number"))
    sum += num
    If num * num < 0 Then
      flag = True
```

```
    End If
  Loop
  txtOutput.Text = CStr(num)
End Sub
```

13.

```
Private Sub btnDisplay_Click(...) Handles btnDisplay.Click
  'Display names of four U.S. Presidents
  Dim president As String
  Dim sr As IO.StreamReader = IO.File.OpenText("PRES.TXT")
  president = sr.ReadLine
```

[Page 274]

```
  Do
    lstOutput.Items.Add(president)
    president = sr.ReadLine
  Loop Until (sr.Peek = -1)
  sr.Close()
End Sub
```

(Assume that the lines of the file PRES.TXT contain the following entries: Lincoln, Washington, Kennedy, Jefferson.)

14.

```
Private Sub btnDisplay_Click(...) Handles btnDisplay.Click
  Dim num As Double
  Dim sr As IO.StreamReader = IO.File.OpenText("DATA.TXT")
  If (sr.Peek = -1) Then
    num = 0
  Else
    num = CDbl(sr.ReadLine)
  End If
  Do While 1 < num < 5
    lstOutput.Items.Add(num)
    If (sr.Peek = -1) Then
      num = 0
    Else
      num = CDbl(sr.ReadLine)
    End If
  Loop
  sr.Close()
End Sub
```

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

15. Write a program to find and display the largest of a collection of positive numbers contained in a text file. (Test the program with the collection of numbers 89, 77, 95, and 86.)

16. Write a program to find and display those names that are repeated in a text file. Assume that the file has already been sorted into alphabetical order. When a name is found to be repeated, display it only once.

17. Suppose that the file FINAL.TXT contains student grades on a final exam. Write a program that displays the average grade on the exam and the percentage of grades that are above average.

18. Suppose that the file BIDS.TXT contains a list of bids on a construction project. Write a program to analyze the list and report the two highest bids.

19. Suppose that the file USPRES.TXT contains the names of the United States presidents in order from George Washington to George W. Bush. Write a program that asks the user to type a number from 1 to 43 into a text box and then, when the user clicks a button, displays the name of the president corresponding to that number.

[Page 275]

20. Table 6.1 shows the different grades of eggs and the minimum weight required for each classification. Write a program that processes the text file EGGS.TXT containing a list of the weights of a sample of eggs. The program should report the number of eggs in each grade and the weight of the lightest and heaviest egg in the sample. Figure 6.6 shows a possible output of the program. Note: Eggs weighing less than 1.5 ounces cannot be sold in supermarkets.

Table 6.1. Grades of eggs.

Grade	Weight (in ounces)
Jumbo	2.5
Extra Large	2.25
Large	2
Medium	1.75
Small	1.5

Figure 6.6. Possible output for Exercise 20.

```
57 Jumbo eggs
95 ExTRa Large eggs
76 Large eggs
96 Medium eggs
77 Small eggs
Lightest egg: 1 ounces
Heaviest egg: 2.69 ounces
```

21. Write a program to request a positive integer as input and carry out the following algorithm. If the number is even, divide it by 2. Otherwise, multiply the number by 3 and add 1. Repeat this process with the resulting number and continue repeating the process until the number 1 is reached. After the number 1 is reached, the program should display how many iterations were required. Note: A number is even if int(num/2) = num/2. (Test the program with the numbers 9, 21, and 27.)

22. Suppose that the file USPRES.TXT contains the names of all United States Presidents, and the file USSENATE.TXT contains the names of all former and present U.S. Senators. Write a program with nested loops that uses these files to display the names of all Presidents who served in the Senate.

23. Suppose that the file SONNET.TXT contains Shakespeare's Sonnet #18. Each entry in the file consists of a line of the sonnet. Write a program using nested loops to analyze this file line by line and report the average number of words in a line and the total number of words in the sonnet.

24. Suppose that the file SALES.TXT contains information on the sales during the past week at a new car dealership. The file contains the following information for each salesperson at the dealership: the name of the salesperson, pairs of numbers giving the final sales price and the dealer cost for each sale made by that salesperson, and a pair of zeros to indicate the end of data for that salesperson. The first fourteen lines of the file contain the following data: Tom Jones, 18100, 17655, 22395, 21885, 15520, 14895, 0, 0, Bill Smith, 16725, 16080, 0, 0. Write a program to display the name of each salesperson and the commission earned for the week. Assume that the commission is 15% of the profit on each sale.

[Page 276]

25. Write a program that uses a flag and does the following:

a. Ask the user to input a sentence containing parentheses. Note: The closing parenthesis should not directly precede the period.

b. Display the sentence with the parentheses and their contents removed. Test the program with the following sentence as input: BASIC (Beginners All-purpose Symbolic Instruction Code) was the world's most widely known computer language.

26. The salespeople at a health club keep track of the members who have joined in the last month. Their names and types of membership (Bronze, Silver, or Gold) are stored in the text file NEWMEMBERS.TXT. Write a program that displays all the Bronze members, then the Silver members, and finally the Gold members.

27. Table 6.2 gives the prices of various liquids. Write a program that requests an amount of money as input and displays the names of all liquids for which a gallon could be purchased with that amount of money. The information from the table should be read from a file. As an example, if the user has $2.35, then the following should be displayed in the list box:

```
You can purchase one gallon of any of the following liquids:
Bleach
Gasoline
Milk
```

Table 6.2. Some comparative prices per gallon of various liquids.

Liquid	Price	Liquid	Price
Apple Cider	2.60	Milk	2.30
Beer	6.00	Gatorade	4.20
Bleach	1.40	Perrier	6.85
Coca-Cola	2.55	Pancake Syrup	15.50
Gasoline	2.25	Spring Water	4.10

Solutions to Practice Problems 6.2

1. 350

2. When the third score was read from the file, sr.Peek assumed the value -1 With sr.peek = -1,the loop terminated and the third score was never added to sum.

3.
```
Private Sub btnCompute_Click(...) Handles btnCompute.Click
  'Find the sum of a collection of bowling scores
  Dim sum, score As Double
  Dim sr As IO.StreamReader = IO.File.OpenText("SCORE.TXT")
  Do While (sr.Peek <> -1)
    score = CDbl(sr.ReadLine)
    sum += score
```

[Page 277]

```
  Loop
  sr.Close()
  txtTotal.Text = CStr(sum)
End Sub
```

PREV PREV NEXT NEXT

[Page 277 (continued)]

6.3. For... Next Loops

When we know exactly how many times a loop should be executed, a special type of loop, called a For... Next loop, can be used. For... Next loops are easy to read and write, and have features that make them ideal for certain common tasks. The following code uses a For... Next loop to display a table:

```
Private Sub btnDisplayTable_Click(...) Handles btnDisplayTable.Click
 'Display a table of the first 5 numbers and their squares
  'Assume the font for lstTable is Courier New
  Dim i As Integer
  For i = 1 To 5
    lstTable.Items.Add(i & "  " & i ^ 2)
  Next
End Sub
```

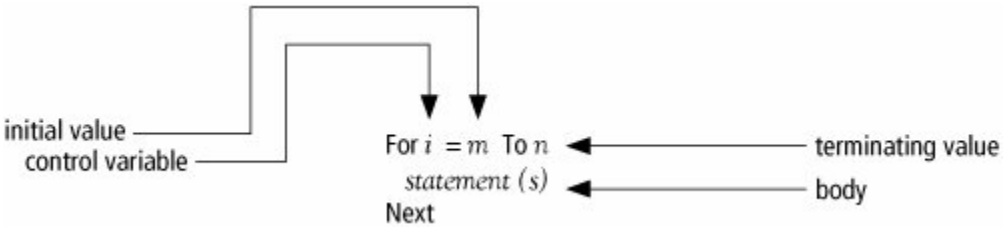
[Run, and click on btnDisplayTable. The following is displayed in the list box.]

```
1  1
2  4
3  9
4  16
5  25
```

The equivalent program written with a Do loop is as follows.

```
Private Sub btnDisplayTable_Click(...) Handles btnDisplayTable.Click
  'Display a table of the first 5 numbers and their squares
  Dim i As Integer
  i = 1
  Do While i <= 5
    lstTable.Items.Add(i & "  " & i ^ 2)
    i += 1    'Add 1 to i
  Loop
End Sub
```

In general, a portion of a program of the form



[Page 278]

constitutes a For... Next loop. The pair of statements For and Next cause the statements between them to be repeated a specified number of times. The For statement designates a numeric variable, called the control variable, that is initialized and then automatically changes after each execution of the loop. Also,

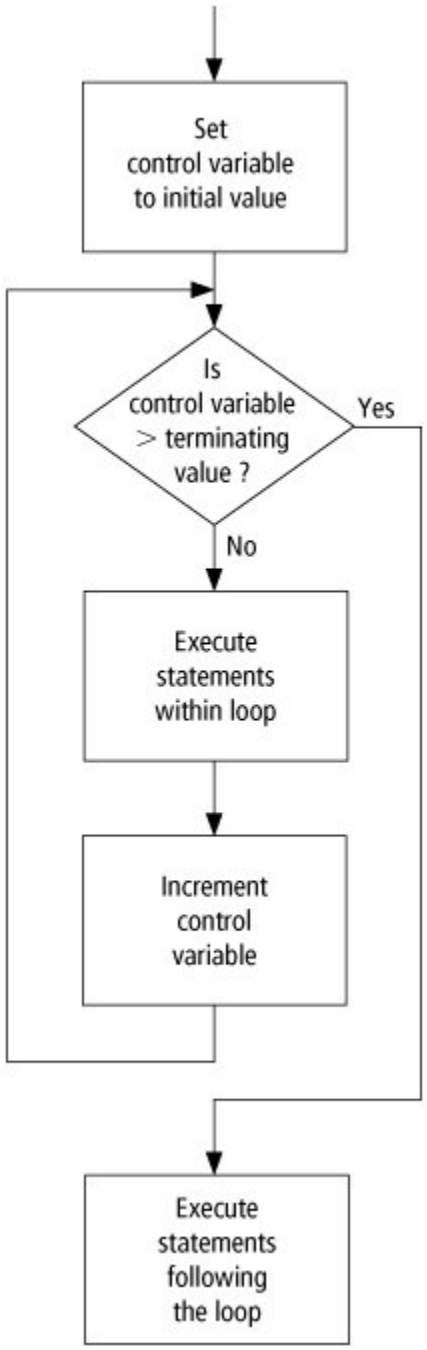
the For statement gives the range of values this variable will assume. The Next statement increments the control variable. If m  n, theni is assigned the values m, m + 1, ..., n in order, and the body is executed once for each of these values. If m > n, then the body is skipped and execution continues with the statement after the For... Next loop.

[Page 279]

When program execution reaches a For... Next loop, such as the one shown previously, the For statement assigns to the control variable i the initial value m and checks to see whether i is greater than the terminating value n. If so, then execution jumps to the line following the Next statement. If i<=n, the statements inside the loop are executed. Then, the Next statement increases the value of i by 1 and checks this new value to see if it exceeds n. If not, the entire process is repeated until the value of i exceeds n. When this happens, the program moves to the line following the loop. Figure 6.7 contains the pseudocode and flowchart of a For ... Next loop.

Figure 6.7. Pseudocode and flowchart of a For Next loop.
(This item is displayed on page 278 in the print version)

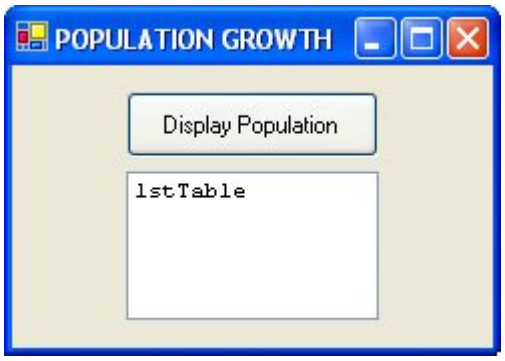
```
For i = m to n
  Processing steps(s)
Next
```



The control variable can be any numeric variable. The most common single-letter names are i, j, and k; however, if appropriate, the name should suggest the purpose of the control variable.

Example 1.

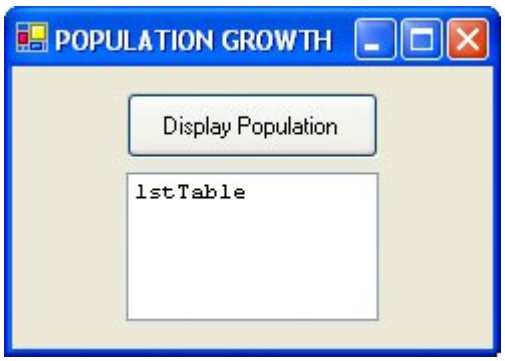
Suppose the population of a city is 300,000 in the year 2006 and is growing at the rate of 3 percent per year. The following program displays a table showing the population each year until 2010.



Object	Property	Setting
frmPopulation	Text	POPULATION GROWTH
btnDisplay	Text	Display Population
lstTable		

```
Private Sub btnDisplay_Click(...) Handles btnDisplay.Click
  'Display population from 2006 to 2010
  Dim pop As Double = 300000, yr As Integer
  Dim fmtStr As String = "{0,4}{1,12:N0}"
  For yr = 2006 To 2010
     lstTable.Items.Add(String.Format(fmtStr, yr, pop))
     pop += 0.03 * pop
  Next
End Sub
```

[Run, and click the button.]



[Page 280]

The initial and terminating values can be literals, variables, or expressions. For instance, the For statement in the preceding program can be replaced by

```
Dim firstYr As Integer = 2006
Dim lastYr As Integer = 2010
```

```
For yr = firstYr To lastYr
```

In Example 1, the control variable was increased by 1 after each pass through the loop. A variation of the For statement allows any number to be used as the increment. The statement

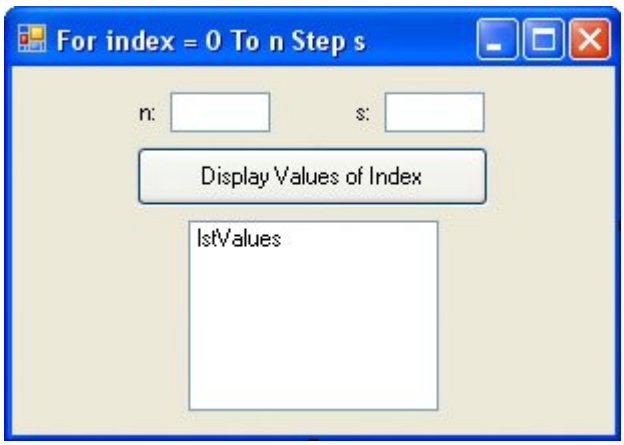
```
For i = m To n Step s
```

instructs the Next statement to add s to the control variable instead of 1. The numbers m, n, and s do not have to be whole numbers. The number s is called the step value of the loop. Note: If the control variable will assume values that are not whole numbers, then the variable must be of type Double.

Example 2.

(This item is displayed on pages 280 - 281 in the print version)

The following program displays the values of the index of a For... Next loop for terminating and step values input by the user:



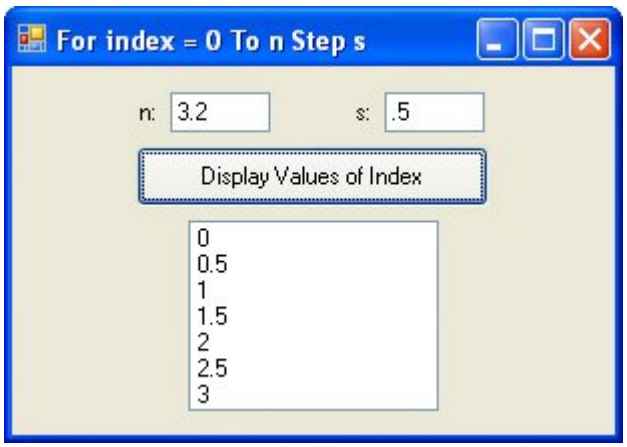
Object	Property	Setting
frmIndex	Text	For index = 0 To n Step s
lblN txtEnd	Text	n:
lblS txtStep	Text	s:
btnDisplay	Text	Display Values of Index
lstValues		

```
Private Sub btnDisplay_Click(...) Handles btnDisplay.Click
  'Display values of index ranging from 0 to n Step s
  Dim n, s, As Double
  Dim index As Double
  n = CDbl(txtEnd.Text)
  s = CDbl(txtStep.Text)
  lstValues.Items.Clear()
```

```
  For index = 0 To n Step s
    lstValues.Items.Add(index)
  Next
End Sub
```

[Page 281]

[Run, type 3.2 and .5 into the text boxes, and click the button.]

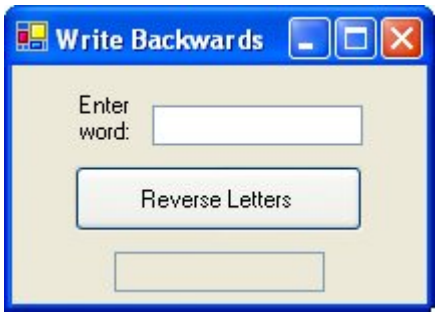


In the examples considered so far, the control variable was successively increased until it reached the terminating value. However, if a negative step value is used and the initial value is greater than the terminating value, then the control value is decreased until reaching the terminating value. In other words, the loop counts backward or downward.

Example 3.

(This item is displayed on pages 281 - 282 in the print version)

The following program accepts a word as input and displays it backwards:



Object	Property	Setting
FrmBackwards	Text	Write Backwards
lblWord	AutoSize Text	False Enter word:
txtWord	Text	Reverse Letters

btnReverse

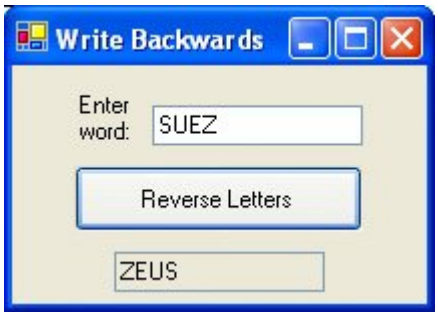
txtBackwards ReadOnly True

```
Private Sub btnReverse_Click(...) Handles btnReverse.Click
  txtBackwards.Text = Reverse(txtWord.Text)
End Sub

Function Reverse(ByVal info As String) As String
  Dim m, j As Integer, temp As String =""
  m = info.Length
  For j = m - 1 To 0 Step -1
    temp &= info.Substring(j, 1)
  Next
  Return temp
End Function
```

[Page 282]

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



Note: The initial and terminating values of a For ...Next loop can be expressions. For instance, the third and fourth lines of the function in Example 3 can be consolidated to

```
For j = info.Length - 1 To 0 Step -1
```

Declaration of Control Variables

The control variable of a For ... Next loop can be declared directly in the For statement. A statement of the form

```
For i As DataType = m to n
```

(possibly with a Step clause) both declares the control variable i and specifies its initial and terminating values. In this case, however, the scope of the control variable i is limited to the body of the For ... Next loop. For instance, in Example 1, the two statements

```
Dim yr As Integer
For yr = 2006 to 2010
```

can be replaced by the single statement

```
For yr As Integer = 2006 to 2010
```

In Example 2, the two statements

```
Dim index as Double
For index = 0 to n Step s
```

can be replaced by the single statement

```
For index As Double = 0 to n Step s
```

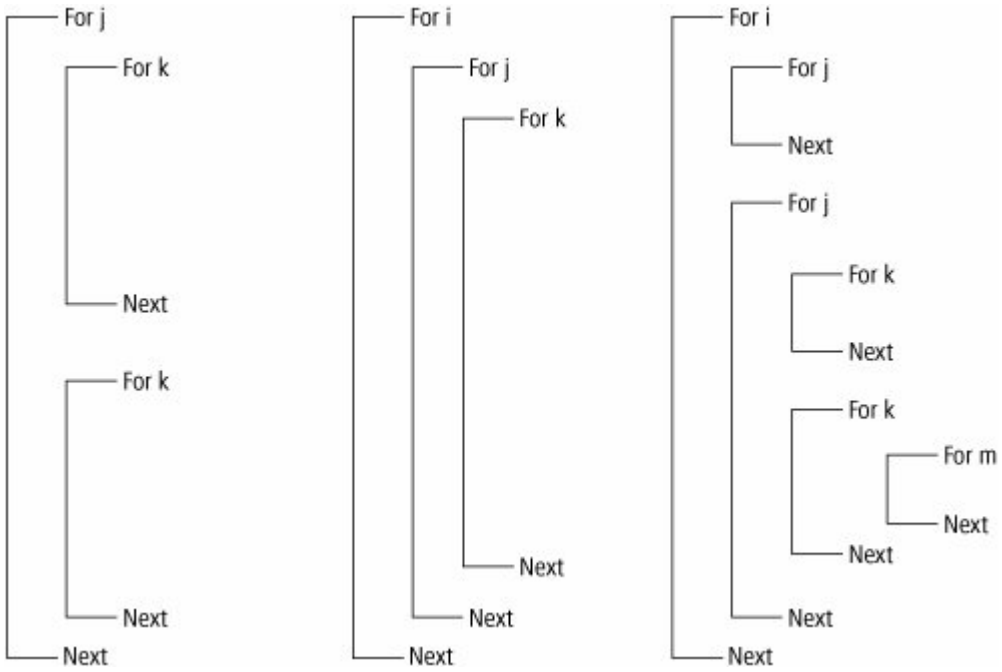
This feature is new to Visual Basic 2005 and is the preferred way to declare control variables of For... Next loops.

Nested For... Next Loops

The body of a For... Next loop can contain any sequence of Visual Basic statements. In particular, it can contain another For... Next loop. However, the second loop must be completely contained inside the first loop and must have a different control variable. Such a configuration is called nested For...Next loops. Figure 6.8 shows several examples of valid nested loops.

[Page 283]

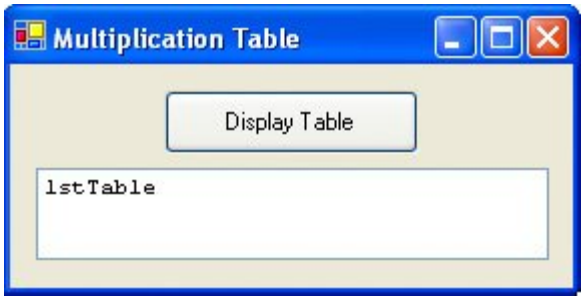
Figure 6.8. Nested For Next loops.



Example 4.

(This item is displayed on pages 283 - 284 in the print version)

The following program displays a multiplication table for the integers from 1 to 3. Here j denotes the left factors of the products, and k denotes the right factors. Each factor takes on a value from 1 to 3. The values are assigned to j in the outer loop and to k in the inner loop. Initially, j is assigned the value 1, and then the inner loop is traversed three times to produce the first row of products. At the end of these three passes, the value of j will still be 1, and the value of k will have been incremented to 4. The first execution of the outer loop is then complete. Following this, the statement Next increments the value of j to 2. The statement beginning "For k" is then executed. It resets the value of k to 1. The second row of products is displayed during the next three executions of the inner loop and so on.



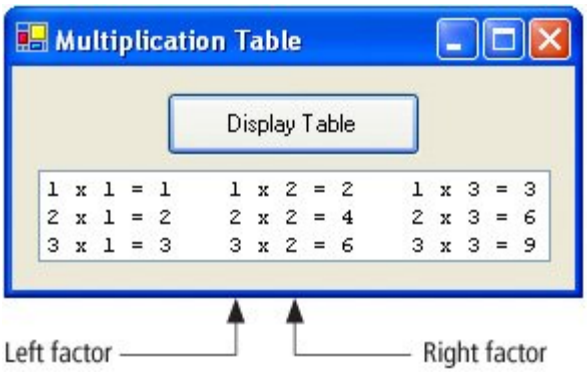
Object	Property	Setting
frmTable	Text	Multiplication Table
btnDisplay	Text	Display Table
lstTable	Font	Courier New

```
Private Sub btnDisplay_Click(...) Handles btnDisplay.Click
  Dim row, entry As String
  lstTable.Items.Clear()
  For j As Integer = 1 To3
    row = ""
```

[Page 284]

```
  For k As Integer = 1 To3
    entry = j & " x " & k & " = " & (j * k)
    row &= entry & "    "
   Next
   lstTable.Items.Add(row)
 Next
End Sub
```

[Run, and press the button.]



Comments

1. For and Next statements must be paired. If one is missing, the automatic syntax checker will complain with a wavy underline and a message such as "A 'For' must be paired with a 'Next'."

2. Consider a loop beginning with For i = m To n Step s. The loop will be executed exactly once if m equals n no matter what value s has. The loop will not be executed at all if m is greater than n and s is positive, or if m is less than n and s is negative.

3. The value of the control variable should not be altered within the body of the loop; doing so might cause the loop to repeat indefinitely or have an unpredictable number of repetitions.

4. Noninteger step values can lead to roundoff errors with the result that the loop is not executed the intended number of times. For instance, a loop beginning with `For i As Double = 1 To 2 Step .1` will be executed only 10 times instead of the intended 11 times. It should be replaced with `For i As Double = 1 To 2.01 Step .1`.

5. Visual Basic provides a way to skip an iteration in a For ... Next loop. When the statement `Continue For` is encountered in the body of the loop, execution immediately jumps to the Next statement. An analogous statement `Continue Do` is available for Do loops. This feature is new in Visual Basic 2005.

6. Visual Basic provides a way to back out of a For ... Next loop. When the statement `Exit For` is encountered in the body of the loop, execution jumps immediately to the statement following the Next statement. An analogous statement `Exit Do` is available for Do loops.

[Page 285]

Practice Problems 6.3

1. Why won't the following lines of code work as intended?

```
For i As Integer = 15 To 1
  lstBox.Items.AddItem(i)
Next
```

2. When is a For... Next loop more appropriate than a Do loop?

Exercises 6.3

In Exercises 1 through 12, determine the output displayed in the list box when the button is clicked.

1.
```
Private Sub btnDisplay_Click(...) Handles btnDisplay.Click
  For i As Integer = 1 To 4
    lstBox.Items.Add("Pass #"& i)
  Next
End Sub
```

2.
```
Private Sub btnDisplay_Click(...) Handles btnDisplay.Click
  For i As Integer = 3 To 6
    lstBox.Items.Add(2 * i)
  Next
End Sub
```

3.
```
Private Sub btnDisplay_Click(...) Handles btnDisplay.Click
  For j As Integer = 2 To 8 Step 2
    lstBox.Items.Add(j)
  Next
  lstBox.Items.Add("Who do we appreciate?")
End Sub
```

4.
```
Private Sub btnDisplay_Click(...) Handles btnDisplay.Click
  For countdown As Integer = 10 To 1 Step -1
    lstBox.Items.Add(countdown)
```

```
  Next
  lstBox.Items.Add("blastoff")
End Sub
```

5.
```
Private Sub btnDisplay_Click(...) Handles btnDisplay.Click
  Dim num As Integer
  num = 5
  For i As Integer = num To (2 * num - 3)
    lstBox.Items.Add(i)
  Next
End Sub
```

[Page 286]

6.
```
Private Sub btnDisplay_Click(...) Handles btnDisplay.Click
  For i As Double = 3 To 5 Step .25
    lstBox.Items.Add(i)
  Next
  lstBox.Items.Add(i)
End Sub
```

7.
```
Private Sub btnDisplay_Click(...) Handles btnDisplay.Click
  'First entry in text file is number of records in file
  Dim recCount As Integer
  Dim name, mileTime As String
  Dim sr As IO.StreamReader = IO.File.OpenText("MILER.TXT")
  recCount = CInt(sr.Readline)
  For miler As Integer = 1 To recCount
    name = sr.Readline
    mileTime = sr.Readline
    lstBox.Items.Add(name & "  "& mileTime)
  Next
  sr.Close()
End Sub
```

(Assume that the seven lines of the file MILER.TXT contain the following entries: 3, Steve Cram, 3:46.31, Steve Scott, 3:51.6, Mary Slaney, 4:20.5.)

8.
```
Private Sub btnDisplay_Click(...) Handles btnDisplay.Click
  'First entry in text file is number of records in file
  Dim recCount, total, score As Integer
  Dim sr As IO.StreamReader = IO.File.OpenText("SCORES.TXT")
  recCount = CInt(sr.Readline)
  For i As Integer = 1 To recCount
    score = CInt(sr.Readline)
    total += score 'Add the value of score to the value of total
  Next
```

```
  sr.Close()
  lstBox.Items.Add("Average = "& total / recCount)
End Sub
```

(Assume the file SCORES.TXT contains the entries 4, 89, 85, 88, 98.)

9.

```
Private Sub btnDisplay_Click(...) Handles btnDisplay.Click
  Dim row, entry As String
  For i As Integer = 0 To 2
    row = ""
    For j As Integer = 0 To 3
      entry = CStr(i + (3 * j) + 1)
      row &= entry & "    "
    Next
    lstBox.Items.Add(row)
  Next
End Sub
```

[Page 287]

10.

```
Private Sub btnDisplay_Click(...) Handles btnDisplay.Click
  Dim row As String
  For i As Integer = 1 To 5
    row = ""
    For j As Integer = 1 To i
      row &= "*"
    Next
    lstBox.Items.Add(row)
  Next
End Sub
```

11.

```
Private Sub btnDisplay_Click(...) Handles btnDisplay.Click
  Dim word As String
  Dim num1, num2 As Integer
  word = InputBox("Please enter a word.")
  num1 = CInt(Int((20 - word.Length) / 2))
  num2 = 20 - num1 - word.Length
  lstBox.Items.Add(Asterisks(num1) & word & Asterisks(num2))
End Sub

Function Asterisks(ByVal num As Integer) As String
  Dim chain As String = ""
  For i As Integer = 1 To num
    chain &= "*"
  Next
  Return chain
End Function
```

(Assume that the response is Hooray.)

12.
```
Private Sub btnDisplay_Click(...) Handles btnDisplay.Click
  'Display an array of letters
  Dim info As String, letter As String
  info = "DATA"
  For i As Integer = 1 To info.Length
    letter = info.Substring(i - 1, 1)
    lstBox.Items.Add(RepeatFive(letter))
  Next
End Sub

Function RepeatFive(ByVal letter As String) As String
  Dim chain As String = ""
  For i As Integer = 1 To 5
    chain &= letter
  Next
  Return chain
End Function
```

[Page 288]

In Exercises 13 through 16, identify the errors.

13.
```
Private Sub btnDisplay_Click(...) Handles btnDisplay.Click
  For j As Double = 1 To 25.5 Step -1
    lstBox.Items.Add(j)
  Next
End Sub
```

14.
```
Private Sub btnDisplay_Click(...) Handles btnDisplay.Click
  For i As Integer = 1 To 3
    lstBox.Items.Add(i & " "& 2 ^ i)
End Sub
```

15.
```
Private Sub btnDisplay_Click(...) Handles btnDisplay.Click
  'Display all numbers from 0 through 20 except for 13
  For i As Double = 20 To 0
    If i = 13 Then
      i = 12
    End If
    lstBox.Items.Add(i)
  Next
End Sub
```

16.
```
Private Sub btnDisplay_Click(...) Handles btnDisplay.Click
  For j As Integer = 1 To 4 Step 0.5
```

```
    lstBox.Items.Add(j)
  Next
End Sub
```

In Exercises 17 and 18, rewrite the program using a For ... Next loop.

17.
```
Private Sub btnDisplay_Click(...) Handles btnDisplay.Click
  Dim num As Integer = 1
  Do While num <= 9
    lstBox.Items.Add(num)
    num += 2    'Add 2 to value of num
  Loop
End Sub
```

18.
```
Private Sub btnDisplay_Click(...) Handles btnDisplay.Click
  lstBox.Items.Add("hello")
  lstBox.Items.Add("hello")
  lstBox.Items.Add("hello")
  lstBox.Items.Add("hello")
End Sub
```

In Exercises 19 through 38, write a program to complete the stated task.

19. Display a row of 10 stars (asterisks).

20. Request a number from 1 to 20 and display a row of that many stars (asterisks).

21. Display a 10-by-10 array of stars.

[Page 289]

22. Request a number and call a Sub procedure to display a square having that number of stars on each side.

23. Find the sum 1 + 1/2 + 1/3 + 1/4 + ... + 1/100

24. Find the sum of the odd numbers from 1 through 99.

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

26. When $1000 is deposited at 5 percent simple interest, the amount grows by $50 each year. When money is invested at 5 percent compound interest, then the amount at the end of each year is 1.05 times the amount at the beginning of that year. Write a program to

display the amounts for 10 years for a $1000 deposit at 5 percent simple and compound interest. The first few lines displayed in the list box should appear as in Figure 6.9.

Figure 6.9. Growth of $1000 at simple and compound interest.

Year	Amount Simple Interest	Amount Compound Interest
1	$1,050.00	$1,050.00
2	$1,100.00	$1,102.50
3	$1,150.00	$1,157.63

27. According to researchers at Stanford Medical School (as cited in Medical Self Care), the ideal weight for a woman is found by multiplying her height in inches by 3.5 and subtracting 108. The ideal weight for a man is found by multiplying his height in inches by 4 and subtracting 128. Request a lower and upper bound for heights and then produce a table giving the ideal weights for women and men in that height range. For example, when a lower bound of 62 and an upper bound of 65 are specified, Figure 6.10 shows the output displayed in the list box.

Figure 6.10. Output for Exercise 27.

Height	Wt-Women	Wt-Men
62	109	120
63	112.5	124
64	116	128
65	119.5	132

28. Table 6.3 (on the next page) gives data (in millions) on personal computer sales and revenues. Read the data from the file PC.TXT, and generate an extended table with two additional columns, Pct Foreign (percent of personal computers sold outside the United States) and Avg Rev (average revenue per computer sold in the United States).

29. Request a sentence, and display the number of sibilants (that is, letters S or Z) in the sentence. The counting should be carried out by a function.

30. Request a number, n, from 1 to 30 and one of the letters S or P. Then, depending upon whether S or P was selected, calculate the sum(s) or product(s) of the numbers from 1 to n. The calculations should be carried out in Function procedures.

Table 6.3. Personal computer sales and revenues (in millions).

Year	U.S. Sales	Worldwide Sales	U.S. Revenues
1998	34.6	98.4	74,920
1999	40.1	116.2	80,200
2000	46.0	128.5	88,110
2001	43.5	132.0	77,000

31. Suppose $800 is deposited into a savings account earning 4 percent interest compounded annually, and $100 is added to the account at the end of each year. Calculate the amount of money in the account at the end of 10 years. (Determine a formula for computing the balance at the end of one year based on the balance at the beginning of the year. Then write a program that starts with a balance of $800 and makes 10 passes through a loop containing the formula to produce the final answer.)

32. A TV set is purchased with a loan of $563 to be paid off with five monthly payments of $116. The interest rate is 1 percent per month. Display a table giving the balance on the loan at the end of each month.

33. Radioactive Decay. Cobalt 60, a radioactive form of cobalt used in cancer therapy, decays or dissipates over a period of time. Each year, 12 percent of the amount present at the beginning of the year will have decayed. If a container of cobalt 60 initially contains 10 grams, determine the amount remaining after five years.

34. Supply and Demand. This year's level of production and price (per bushel) for most agricultural products greatly affects the level of production and price next year. Suppose the current crop of soybeans in a certain country is 80 million bushels and experience has shown that for each year,

[price this year] = 20 - .1 * [quantity this year]

[quantity next year] = 5 * [price this year] - 10,

where quantity is measured in units of millions of bushels. Generate a table to show the quantity and price for each of the next 12 years.

35. Request a number greater than 3, and display a hollow rectangle similar to the one in Figure 6.11(a) with each outer row and column having that many stars. Use a fixed-width font such as Courier New so that the spaces and asterisks will have the same width.

Figure 6.11. Outputs for Exercises 35 and 36.

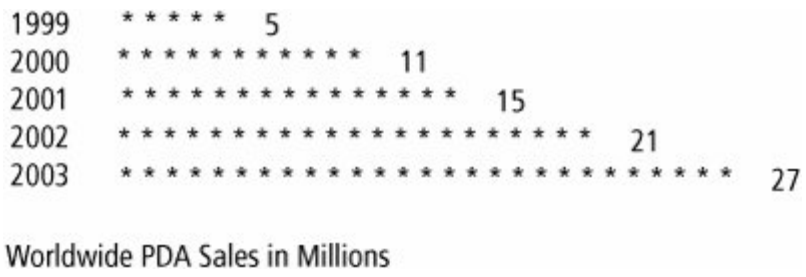
36. Request an odd number, and display a triangle similar to the one in Figure 6.11(b)with the input number of stars in the top row.

[Page 291]

37. A man pays $1 to get into a gambling casino. He loses half of his money there and then has to pay $1 to leave. He goes to a second casino, pays another $1 to get in, loses half of his money again, and pays another $1 to leave. Then, he goes to a third casino, pays another $1 to get in, loses half of his money again, and pays another $1 to get out. After this, he's broke. Write a program to determine the amount of money he began with by testing $5, then $6, and so on.

38. Create the histogram in Figure 6.12. A file should hold the years and values. The first line in the file could be used to hold the title for the histogram.

Figure 6.12. Histogram for Exercise 38.



39. Write a program to estimate how much a young worker will make before retiring at age 65. Request the worker's name, age, and starting salary as input. Assume the worker receives a 5 percent raise each year. For example, if the user enters Helen, 25, and 20000, then the text box should display the following:

```
Helen will earn about $2,415,995
```

40. Write a program that accepts a word as input and determines if its letters are in alphabetical order. (Test the program with the words "almost," "imply," and "biopsy.")

Solutions to Practice Problems 6.3

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

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

or

```
For i = 1 To 15
```

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

PREV PREV NEXT NEXT

[Page 291 (continued)]

6.4. A Case Study: Analyze a Loan

This case study develops a program to analyze a loan. Assume the loan is repaid in equal monthly payments and interest is compounded monthly. The program should request the amount (principal) of the loan, the annual rate of interest, and the number of years over which the loan is to be repaid. The four options to be provided by buttons are as follows:

1. Calculate the monthly payment. The formula for the monthly payment is

   payment = P*r/(1 - (1 + r)^(-n)),

[Page 292]

   where p is the principal of the loan, r is the monthly interest rate (annual rate divided by 12) given as a number between 0 (for 0 percent) and 1 (for 100 percent), and n is the number of months over which the loan is to be repaid. Because a payment computed in this manner can be expected to include fractions of a cent, the value should be rounded up to the next nearest cent. This corrected payment can be achieved using the formula

   correct payment = Math.Round(payment + .005, 2).

2. Display an amortization schedule, that is, a table showing the balance on the loan at the end of each month for any year over the duration of the loan. Also show how much of each monthly payment goes toward interest and how much is used to repay the principal. Finally, display the

total interest paid over the duration of the loan. The balances for successive months are calculated with the formula

balance = (1 + r) * b - m,

where r is the monthly interest rate (annual rate / 12, a fraction between 0 and 1), b is the balance for the preceding month (amount of loan left to be paid), and m is the monthly payment.

3. Show the effect of changes in the interest rate. Display a table giving the monthly payment for each interest rate from 1 percent below to 1 percent above the specified annual rate in steps of one-eighth of a percent.

4. Quit.

Designing the Analyze-a-Loan Program

For each of the tasks described in preceding options 1 to 4, the program must first look at the text boxes to obtain the particulars of the loan to be analyzed. Thus, the first division of the problem is into the following tasks:

1. Input the principal, interest, and duration.

2. Calculate the monthly payment.

3. Calculate the amortization schedule.

4. Display the effects of interest-rate changes.

5. Quit.

Task 1 is a basic input operation and Task 2 involves applying the formula given in Option 1; therefore, these tasks need not be broken down any further. The demanding work of the program is done in Tasks 3 and 4, which can be divided into smaller subtasks.

3. Calculate amortization schedule. This task involves simulating the loan month by month. First, the monthly payment must be computed. Then, for each month, the new balance must be computed together with a decomposition of the monthly payment into the amount paid for interest and the amount going toward repaying the principal. That is, Task 3 is divided into the following subtasks:

[Page 293]

3.1 Calculate monthly payment.

3.2 Calculate new balance.

3.3 Calculate amount of monthly payment for principal.

3.4 Calculate amount of monthly payment for interest.

3.5 Calculate total interest paid.

4. Display the effects of interest-rate changes. A table is needed to show the effects of changes in the interest rate on the size of the monthly payment. First, the interest rate is reduced by one percentage point and the new monthly payment is computed. Then the interest rate is increased by regular increments until it reaches one percentage point above the original rate, with new monthly payment amounts computed for each intermediate interest rate. The subtasks for this task are then as follows:

4.1 Reduce the interest rate by 1 percent.

4.2 Calculate the monthly payment.

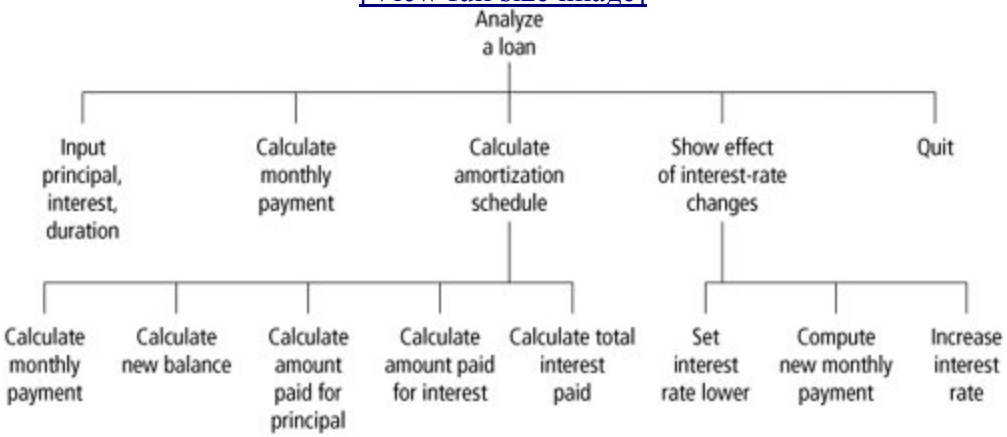
4.3 Increase the interest rate by 1/8 percent.

4.4 Repeat until a certain condition is met.

The hierarchy chart in Figure 6.13 shows the stepwise refinement of the problem.

Figure 6.13. Hierarchy chart for the Analyze-a-Loan Program.

[View full size image]



The User Interface

Figure 6.14 shows a possible form design and Table 6.4 gives the initial settings for the form and its controls. Figures 6.15, 6.16, and 6.17 (following the program) show possible runs of the program for each task available through the buttons. The width and height of the list box were adjusted by trial and error to handle the extensive output generated.

[Page 294]

Figure 6.14. Template for the Analyze-a-Loan program.

[View full size image]

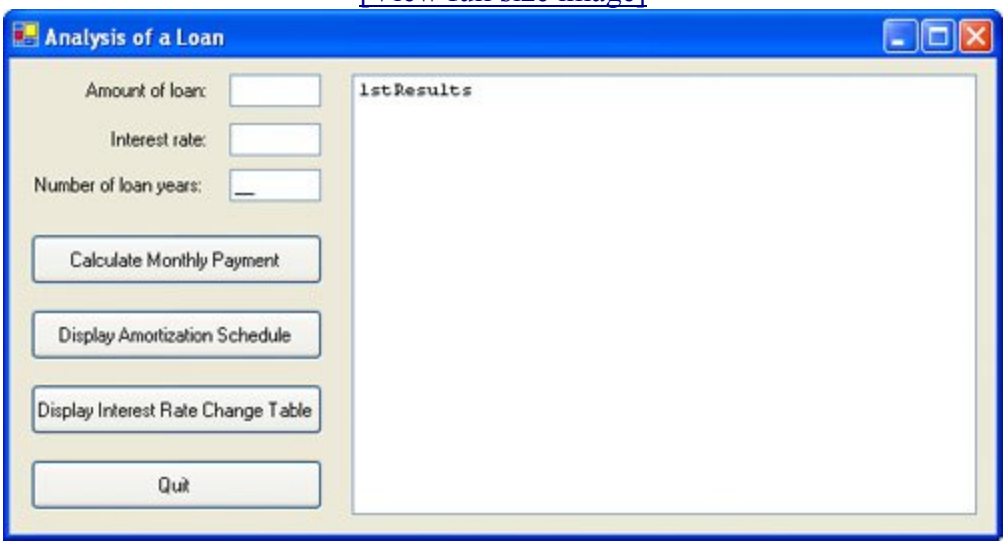


Table 6.4. Objects and initial properties for the Analyze-a-Loan program.

Object	Property	Setting
frmLoan	Text	Analysis of a Loan
lblPrincipal	Text	Amount of loan:
txtPrincipal		
lblYearly Rate	Text	Interest rate:
txtYearly Rate		
lblNumYears	Text	Number of loan years:
txtNumYears		
btnPayment	Text	Calculate Monthly Payment
btnAmort	Text	Display Amortization Schedule
btnRateTable	Text	Display Interest Rate Change Table
btnQuit	Text	Quit
lstResults		

[Page 295]

Figure 6.15. Monthly payment on a 30-year loan.

[View full size image]

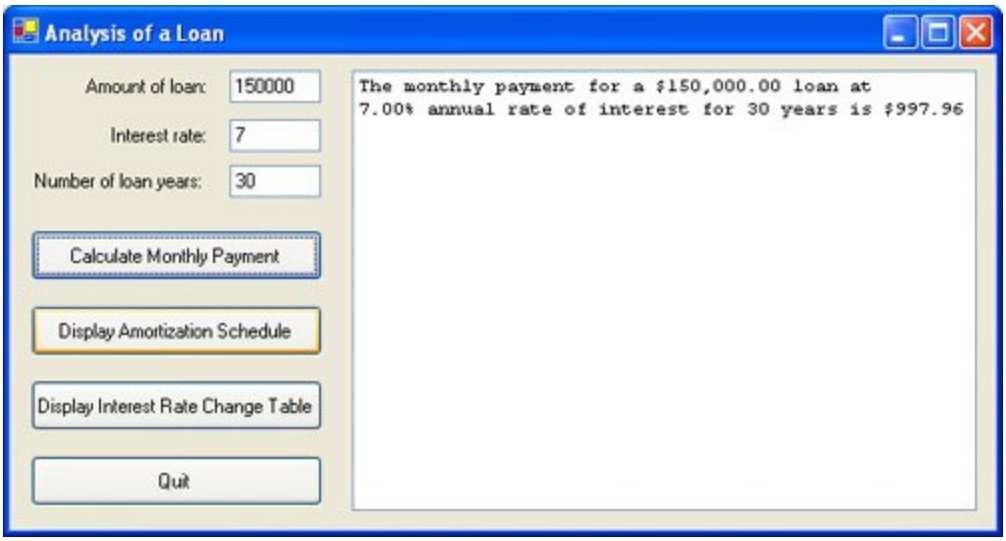
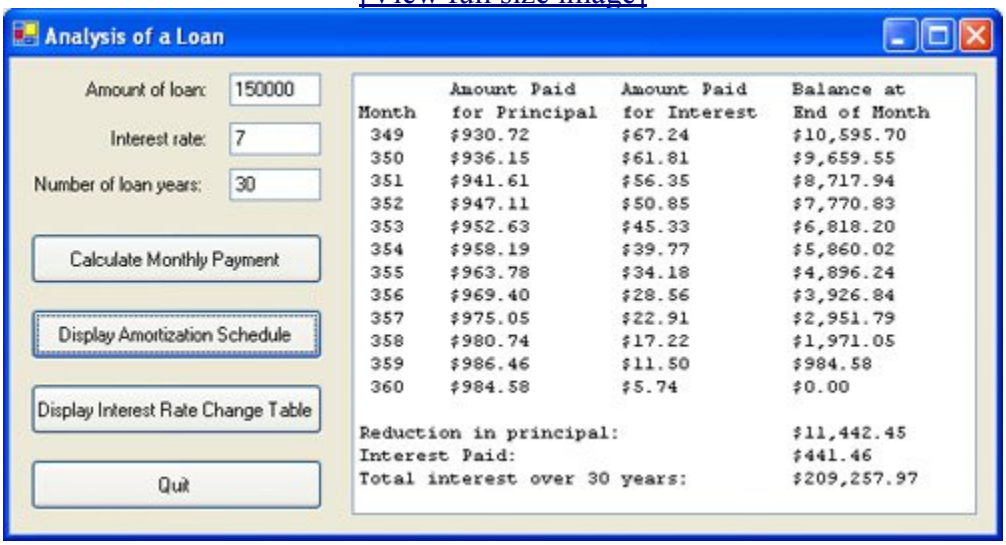


Figure 6.16. Amortization for year 30 of a loan.

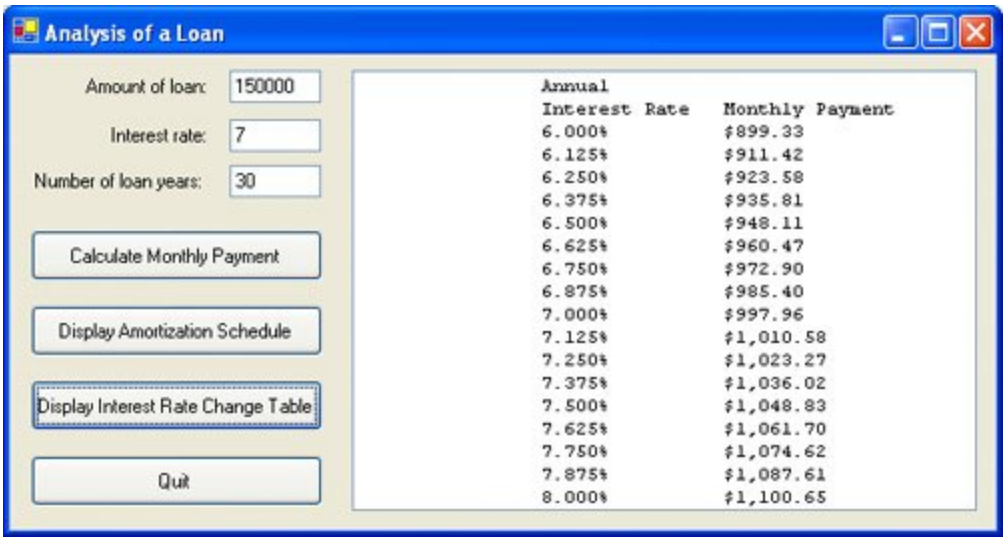
[View full size image]



[Page 296]

Figure 6.17. Interest-rate-change table for a 30-year loan.

[View full size image]



Pseudocode for the Analyze-a-Loan Program

Calculate Monthly Payment button:

INPUT LOAN DATA (Sub procedure InputData)

COMPUTE MONTHLY PAYMENT (Function Payment)

DISPLAY MONTHLY PAYMENT (Sub procedure ShowPayment)

Display Amortization Schedule button:

INPUT LOAN DATA (Sub procedure InputData)

DISPLAY AMORTIZATION SCHEDULE (Sub procedure ShowAmortSched)

Compute monthly interest rate

COMPUTE MONTHLY PAYMENT (Function Payment)

Calculate and display amortization table

Display total interest paid

Display Interest Rate Change Table button:

INPUT LOAN DATA (Sub procedure InputData)

DISPLAY INTEREST RATE CHANGE TABLE (Sub procedure ShowInterestChanges)

Decrease annual rate by .01

Do

Display monthly interest rate

COMPUTE AND DISPLAY MONTHLY PAYMENT (Function Payment)

Increase annual rate by .00125

Loop Until annual rate > original annual rate + .01

[Page 297]

Writing the Analyze-a-Loan Program

Table 6.5 shows each task discussed before and the procedure that carries out the task.

Table 6.5. Tasks and their procedures.

Task		Procedure
1.	Input principal, interest, and duration.	InputData
2.	Calculate monthly payment.	ShowPayment
3.	Calculate amortization schedule.	ShowAmortizationSchedule
	3.1 Calculate monthly payment.	Payment
	3.2 Calculate new balance.	Balance
	3.3 Calculate amount paid for principal.	ShowAmortizationSchedule
	3.4 Calculate amount paid for interest.	ShowAmortizationSchedule
	3.5 Calculate total interest paid.	ShowAmortizationSchedule
4.	Show effect of interest-rate changes.	ShowInterestChanges
	4.1 Reduce interest rate.	ShowInterestChanges
	4.2 Compute new monthly payment.	Payment
	4.3 Increase interest rate.	ShowInterestChanges
	4.4 Repeat until a certain condition is met.	ShowInterestChanges

```
Private Sub btnPayment_Click(...) Handles btnPayment.Click
  Dim principal As Double       'Amount of loan
  Dim yearlyRate As Double      'Annual rate of interest
  Dim numMonths As Integer      'Number of months to repay loan
  InputData(principal, yearlyRate, numMonths)
  ShowPayment(principal, yearlyRate, numMonths)
End Sub

Private Sub btnAmort_Click(...) Handles btnAmort.Click
  Dim principal As Double       'Amount of loan
```

```
  Dim yearlyRate As Double      'Annual rate of interest
  Dim numMonths As Integer      'Number of months to repay loan
  InputData(principal, yearlyRate, numMonths)
  ShowAmortizationSchedule(principal, yearlyRate, numMonths)
End Sub

Private Sub btnRateTable_Click(...) Handles btnRateTable.Click
  Dim principal As Double       'Amount of loan
  Dim yearlyRate As Double      'Annual rate of interest
  Dim numMonths As Integer      'Number of months to repay loan
  InputData(principal, yearlyRate, numMonths)
  ShowInterestChanges(principal, yearlyRate, numMonths)
End Sub

Private Sub btnQuit_Click(...) Handles btnQuit.Click
  End
End Sub
```

[Page 298]

```
Sub InputData(ByRef principal As Double, _
              ByRef yearlyRate As Double, ByRef numMonths As Integer)
  'Input the loan amount, yearly rate of interest, and duration
  Dim percentageRate As Double, numYears As Integer
  principal = CDbl(txtPrincipal.Text)
  percentageRate = CDbl(txtYearlyRate.Text)
  yearlyRate = percentageRate / 100
  numYears = CInt(txtNumYears.Text)
  numMonths = numYears * 12
End Sub

Sub ShowPayment(ByVal principal As Double, _
                ByVal yearlyRate As Double, ByVal numMonths As Integer)
  Dim monthlyRate As Double = yearlyRate / 12
  Dim monthlyPayment As Double
  'Calculate monthly payment
  monthlyPayment = Payment(principal, monthlyRate, numMonths)
  'Display results
  lstResults.Items.Clear()
  lstResults.Items.Add("The monthly payment for a " & _
              FormatCurrency(principal) & " loan at")
  lstResults.Items.Add(FormatNumber(yearlyRate * 100) & _
             "% annual rate of interest for " _
      & FormatNumber(numMonths / 12, 0) & " years is " & _
              FormatCurrency(monthlyPayment))
End Sub

Sub ShowAmortizationSchedule(ByVal principal As Double, _
           ByVal yearlyRate As Double, ByVal numMonths As Integer)
  Dim questionYear As String
  Dim startMonth As Integer
  Dim monthlyRate As Double = yearlyRate / 12
  Dim monthlyPayment As Double
  Dim totalInterest As Double = 0
  Dim yearInterest As Double = 0
  Dim oldBalance, newBalance As Double
  Dim numYears As Double = numMonths / 12
  Dim principalPaid, interestPaid As Double
  Dim principalReduced As Double
  Dim fmtStr As String
```

```
  'Ask for the year to display
  questionYear = "Please enter year (1-" & numYears & _
                 ") for which amortization is to be shown:"
  startMonth = 12 * CInt(InputBox(questionYear, "Which Year?")) - 11
  'Display column headings
  lstResults.Items.Clear()
  fmtStr = "{0,-8}{1,-15}{2,-15}{3,-15}"
  lstResults.Items.Add(String.Format(fmtStr, "", "Amount Paid", _
                                     "Amount Paid", "Balance at"))
```

```
  lstResults.Items.Add(String.Format(fmtStr, "Month", _
                       "for Principal", "for Interest", "End of Month"))
  monthlyPayment = Payment(principal, monthlyRate, numMonths)
  oldBalance = principal
  fmtStr = "{0,4}     {1,-15:C}{2,-15:C}{3,-15:C}"
  'Loop for each month of the loan period
  For monthNum As Integer = 1 To numMonths
     'Calculate the relevant figures
     newBalance = Balance(monthlyPayment, oldBalance, monthlyRate)
     principalPaid = oldBalance - newBalance
     interestPaid = monthlyPayment - principalPaid
     totalInterest = totalInterest + interestPaid
     'Display results if current year is the desired year to display
     If (monthNum >= startMonth)And (monthNum < startMonth + 12) Then
       lstResults.Items.Add(String.Format(fmtStr, monthNum, _
                            principalPaid, interestPaid, newBalance))
       yearInterest = yearInterest + interestPaid
     End If
     'Update the old balance for the next pass through the For loop
     oldBalance = newBalance
  Next
  'Display totals
  principalReduced = 12 * monthlyPayment - yearInterest
  fmtStr = "{0,-38}{1,-15:C}"
  lstResults.Items.Add("")
  lstResults.Items.Add(String.Format(fmtStr, _
                       "Reduction in principal:", principalReduced))
  lstResults.Items.Add(
     String.Format(fmtStr, "Interest Paid:", yearInterest))
  lstResults.Items.Add(String.Format(fmtStr, "Total interest over " & _
                       numYears & " years:", totalInterest))
End Sub

Function Balance(ByRef payment As Double, _
  ByVal principal As Double, ByVal monthlyRate As Double) As Double
  'Compute balance at End of month
  Dim newBalance As Double
  newBalance = (1 + monthlyRate) * principal
  'If balance is less than payment, then this
  'payment is the last one for this loan
  If newBalance <= payment Then
     payment = newBalance
     Return 0
  Else
     Return newBalance - payment
  End If
End Function
```

[Page 300]

```
Function Payment(ByVal principal As Double, _
     ByVal monthlyRate As Double, ByVal numMonths As Integer) As Double
  Dim estimate As Double           'Estimate of monthly payment
  If numMonths = 0 Then
     'If no months then the loan must be repaid immediately
     estimate = principal
  ElseIf monthlyRate = 0 Then
     'If loan has no interest then just repay the principal over time
     estimate = principal / numMonths
  Else
     'Otherwise, use the formula for determining the monthly payment
     estimate = principal * monthlyRate / (1 - _
              (1 + monthlyRate) ^ (-numMonths))
  End If
  'Round the payment up if it there are fractions of a cent
  If estimate = Math.Round(estimate, 2) Then
     Returnestimate
  Else
     ReturnMath.Round(estimate + 0.005, 2)
  End If
End Function
Sub ShowInterestChanges(ByVal principal AsDouble, _
                ByVal yearlyRate AsDouble, ByVal  numMonthsAsInteger)
  Dim  newRate AsDouble, monthlyRate AsDouble, monthlyPayment AsDouble
  Dim fmtStr AsString = "{0,15} {1,-15} {2,-15}"
  'Display effect of interest changes
  lstResults.Items.Clear()
  lstResults.Items.Add(String.Format(fmtStr, "", "Annual", ""))
  lstResults.Items.Add(String.Format(fmtStr, "", _
           "Interest Rate", "Monthly Payment"))
  'Set the annual rate's lower limit for the table
  newRate = yearlyRate - 0.01
  fmtStr = "{0,15} {1,-15} {2,-15:C}"
  'Loop the annual rate from its lower limit to its upper limit
  Do
     'Calculate the monthly payment for the corresponding monthly rate
     monthlyRate = newRate / 12
     monthlyPayment = Payment(principal, monthlyRate, numMonths)
     lstResults.Items.Add(String.Format(fmtStr, "", _
         FormatPercent(newRate, 3), monthlyPayment))
     newRate = newRate + 0.00125   'Increment by 1/8 of one percent for
     'the next pass through the loop
     'To avoid rounding errors, add 0.001 to the upper-limit condition
  LoopUntil newRate > yearlyRate + 0.01 + 0.001
End Sub
```

[Page 301]

Comments

1. Tasks 3.1 and 3.2 are performed by functions. Using functions to compute these quantities simplifies the computations in ShowAmortizationSchedule.

2. Because the payment was rounded up to the nearest cent, it is highly likely that the payment needed in the final month to pay off the loan will be less than the normal payment. For this reason,

ShowAmortizationSchedule checks if the balance of the loan (including interest due) is less than the regular payment and, if so, makes appropriate adjustments.

3. The standard formula for computing the monthly payment cannot be used if either the interest rate is zero percent or the loan duration is zero months. Although both of these situations do not represent reasonable loan parameters, provisions are made in the function Payment so that the program can handle these esoteric situations.

PREV PREV NEXT NEXT

[Page 301 (continued)]

Chapter 6 Summary

1. A Do loop repeatedly executes a block of statements either as long as or until a certain condition is true. The condition can be checked either at the top of the loop or at the bottom.

2. The Peek method can be used to tell us if we have read to the end of a file.

3. As various items of data are processed by a loop, a counter can be used to keep track of the number of items, and an accumulator can be used to sum numerical values.

4. A flag is a Boolean variable, used to indicate whether a certain event has occurred.

5. A For ... Next loop repeats a block of statements a fixed number of times. The control variable assumes an initial value and increments by one after each pass through the loop until it reaches the terminating value. Alternative increment values can be specified with the Step keyword.

PREV PREV NEXT NEXT

[Page 301 (continued)]

Chapter 6 Programming Projects

1. Write a program to display a company's payroll report in a list box. The program should read each employee's name, hourly rate, and hours worked from a file and produce a report in the form of the sample run shown in Figure 6.18. Employees should be paid time-and-a-half for hours in excess of 40.

Figure 6.18. Sample output from Programming Project 1.

```
Payroll Report for week ending 11/17/06

Employee        Hourly Rate   Hours Worked    Gross Pay

Al Adams           $6.50           38           $247.00
```

```
Bob Brown        $5.70          50          $313.50
Carol Coe        $7.00          40          $280.00
Final Total                                 $840.50
```

[Page 302]

2. Table 6.6 shows the standard prices for items in a department store. Suppose prices will be reduced for the annual President's Day Sale. The new price will be computed by reducing the old price by 10 percent, rounding up to the nearest dollar, and subtracting 1 cent. If the new price is greater than the old price, the old price is used as the sale price. Write a program to display in a list box the output shown in Figure 6.19.

Table 6.6. President's Day sale.

Item	Original Price
GumShoes	39.00
SnugFoot Sandals	21.00
T-Shirt	7.75
Maine Handbag	33.00
Maple Syrup	6.75
Flaked Vest	24.00
Nightshirt	26.00

Figure 6.19. Output of Programming Project 2.

```
Item                    Sale Price
GumShoes                   35.99
SnugFoot Sandals           18.99
T-Shirt                     6.99
Maine Handbag              29.99
Maple Syrup                 6.75
Flaked Vest                21.99
Nightshirt                 23.99
```

3. The Rule of 72 is used to make a quick estimate of the time required for prices to double

due to inflation. If the inflation rate is r percent, then the Rule of 72 estimates that prices will double in 72/ryears. For instance, at an inflation rate of 6 percent, prices double in about 72/6 or 12 years. Write a program to test the accuracy of this rule. The program should display a table showing, for each value of r from 1 to 20, the rounded value of 72/r and the actual number of years required for prices to double at an r percent inflation rate. (Assume prices increase at the end of each year.) Figure 6.20 shows the first few rows of the output.

Figure 6.20. Rule of 72 used in Programming Project 3.

```
Interest     Rule of 72         Actual
Rate (%)

   1             72               70

   2             36               36

   3             24               24
```

[Page 303]

4. Table 6.7 shows the number of bachelor degrees conferred in 1980 and 2002 in certain fields of study. Tables 6.8 and 6.9 show the percentage change and a histogram of 2002 levels, respectively. Write a program that allows the user to display any one of these tables as an option and quit as a fourth option.

Table 6.7. Bachelor degrees conferred in certain fields.

Field of Study	1980	2002
Business and management	184,867	281,330
Computer and info. science	11,154	47,299
Education	118,038	106,383
Engineering	68,893	59,481
Social sciences	103,662	132,874

Source: U.S. National Center of Educational Statistics

Table 6.8. Percentage change in bachelor degrees conferred.

Field of Study	% Change (1980-2002)
Business and management	52.2

Computer and info. science	324.1
Education	-9.9
Engineering	-13.7
Social sciences	28.2

Table 6.9. Bachelor degrees conferred in 2002 in certain fields.

Business and management	********** 281,330
Computer and info. science	***** 47,299
Education	********** 106,383
Engineering	****** 59,481
Social sciences	********** 132,874

5. Least-Squares Approximation. Table 6.10 shows the 1988 prices of a gallon of gasoline and the amounts of fuel consumed for several countries. Figure 6.21 displays the data as points in the xy plane. For instance, the point with coordinates (1, 1400) corresponds to the USA. Figure 6.21 also shows the straight line that best fits these data in the least-squares sense. (The sum of the squares of the distances of the 11 points from this line is as small as possible.) In general, if $(x_1,y_1),(x_2,y_2), \ldots,(x_n,y_n)$ are n points in the xy coordinate system, then the least-squares approximation to these points is the line y=mx+b,where

$$m = \frac{n * (\text{sum of } x_i * y_i) - (\text{sum of } x_i) * (\text{sum of } y_i)}{n * (\text{sum of } x_i * x_i) - (\text{sum of } x_i)^2}$$

[Page 304]

and

b = ((sum of y^i) - m * (sum of x^i))/n.

Write a program that calculates and displays the equation of the least-squares line. The program should then allow the user to enter a gasoline price and use the equation of the line to predict the corresponding consumption of fuel. (Place the numeric data from the table in a text file.) A sample run is shown in Figure 6.22.

Table 6.10. A comparison of 1988 gasoline prices and per capita fuel use.

Country	Price per gallon in U.S. Dollars	Tons of Fuel per 1000 Persons	Country	Price per gallon in U.S. Dollars	Tons of Fuel per 1000 Persons
USA	$1.00	1400	France	$3.10	580
W. Ger.	$2.20	620	Norway	$3.15	600
England	$2.60	550	Japan	$3.60	410
Austria	$2.75	580	Denmark	$3.70	570
Sweden	$2.80	700	Italy	$3.85	430
Holland	$3.00	490			

Source: World Resources Institute

Figure 6.21. Least-squares fit to data from Table 6.10.

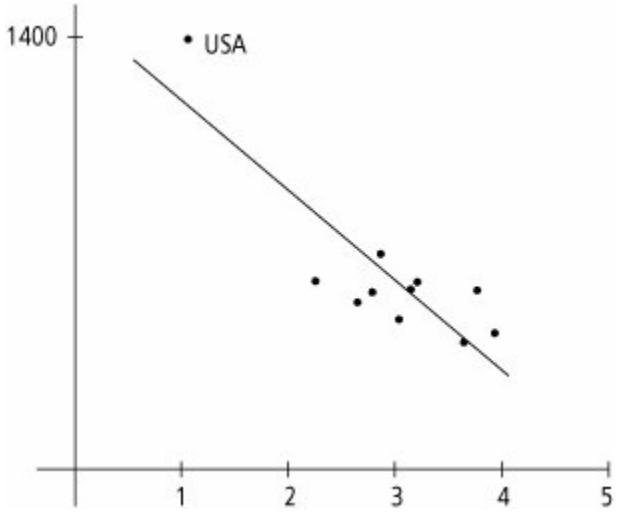
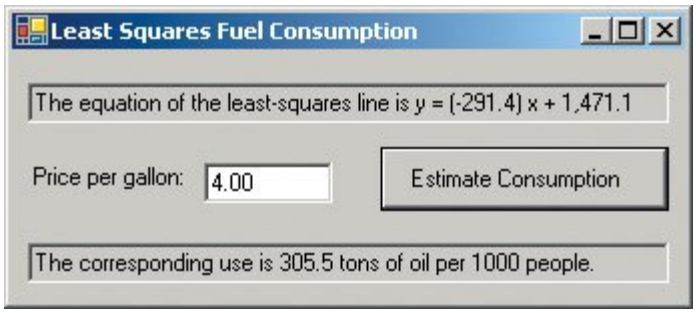


Figure 6.22. Sample run of Programming Project 5.



[Page 305]

6. Write a program to provide information on the height of a ball thrown straight up into the air. The program should request the initial height, h feet, and the initial velocity, v feet per second, as input. The four options to be provided by buttons are as follows:

   a. Determine the maximum height of the ball. Note: The ball will reach its maximum height after v/32 seconds.

   b. Determine approximately when the ball will hit the ground. Hint: Calculate the height after every .1 second and observe when the height is no longer a positive number.

   c. Display a table showing the height of the ball every quarter second for five seconds or until it hits the ground.

   d. Quit.

   The formula for the height of the ball aftert seconds, h + v*t - 16*t*t*, should be specified in a user-defined function. (Test the program with v = 148 and h = 0. This velocity is approximately the top speed clocked for a ball thrown by a professional baseball pitcher.)

7. Depreciation to a Salvage Value of 0. For tax purposes an item may be depreciated over a period of several years, n. With the straight-line method of depreciation, each year the item depreciates by 1/nth of its original value. With the double-declining-balance method of depreciation, each year the item depreciates by 2/nths of its value at the beginning of that year. (In the last year, it is depreciated by its value at the beginning of the year.) Write a program that performs the following tasks:

   a. Requests a description of the item, the year of purchase, the cost of the item, the number of years to be depreciated (estimated life), and the method of depreciation. The method of depreciation should be chosen by clicking one of two buttons.

   b. Displays a depreciation schedule for the item similar to the schedule shown in Figure 6.23.

Figure 6.23. Depreciation schedule for Programming Project 7.

```
Description: Computer

Year of purchase: 2002

Cost: $2,000.00

Estimated life: 5

Method of depreciation: double-declining-balance
```

Year	Value at Beg of Yr	Amount Deprec During Year	Total Depreciation to End of Year
2002	2,000.00	800.00	800.00
2003	1,200.00	480.00	1,280.00
2004	720.00	288.00	1,568.00
2005	432.00	172.80	1,740.80
2006	259.20	259.20	2,000.00

[Page 306]

8. The Twelve Days of Christmas. Each year, PNC Advisors of Pittsburgh publishes a Christmas price list. See Table 6.11. Write a program that requests an integer from 1 through 12 and then lists the gifts for that day along with that day's cost. On the nth day, the n gifts are 1 partridge in a pear tree, 2 turtle doves, ..., n of the nth item. The program also should give the total cost of all twelve days. As an example, Figure 6.24 shows the output in the list box when the user enters 3.

Table 6.11. Christmas price list for 2005.

Item	Cost	Item	Cost
partridge in a pear tree	104.99	swan-a-swimming	600.00
turtle dove	20.00	maid-a-milking	5.15
French hen	15.00	lady dancing	508.46
calling bird	99.99	lord-a-leaping	403.91
gold ring	65.00	piper piping	186.65
geese-a-laying	50.00	drummer drumming	185.36

Figure 6.24. Sample output for Programming Project 8.

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

```
3 french hens
Cost: $189.99
Total cost for the twelve days: $72,608.00
```

PREV NEXT