6

Repetition

6.1 Do Loops

A **loop**, one of the most important structures in programming, 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".

Pretest Form of a Do Loop

VideoNote
Do loops

When Visual Basic encounters 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 statement Loop. 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 as long as (that is, while) the condition is true. Figure 6.1 contains the pseudocode and flowchart for this loop.

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

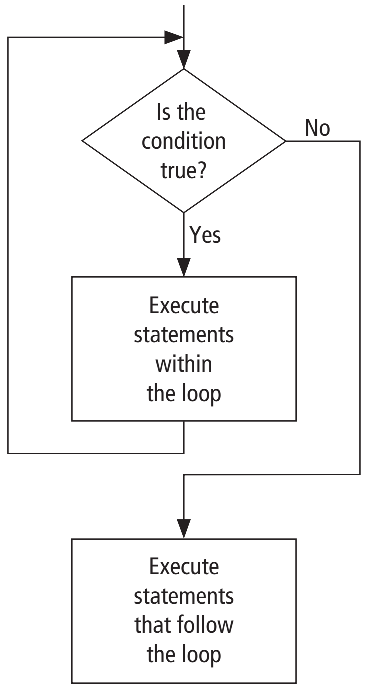

Is the condition true?
No
Yes
Execute statements within the loop
Execute statements that follow the loop

FIGURE 6.1 **Pseudocode and flowchart for a Do loop with the condition tested at the top.**

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 terminates, the value of *num* will be 8.)

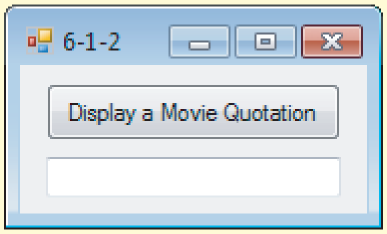
```
Private Sub btnDisplay_Click(...) Handles btnDisplay.Click
  'Display the numbers from 1 to 7
  Dim num As Integer = 1
  Do While num <= 7
    lstNumbers.Items.Add(num)
    num += 1     'Add 1 to the value of num
  Loop
End Sub
```

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

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

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

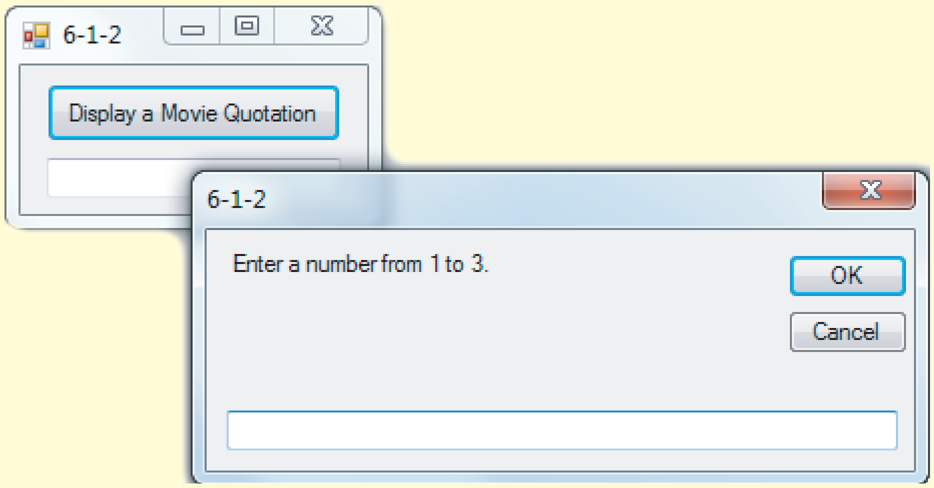
Example 2 The following program requires the user to enter a number from 1 through 3. The Do loop repeats the request until the user gives a proper response.



OBJECT	PROPERTY	SETTING
frmMovie	Text	6-1-2
btnDisplay	Text	Display a Movie Quotation
txtQuotation	ReadOnly	True

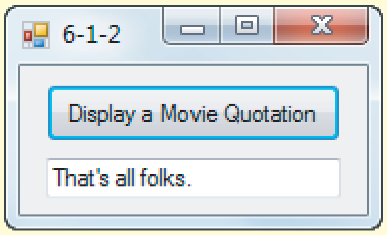
```
Private Sub btnDisplay_Click(...) Handles btnDisplay.Click
  Dim response As Integer, quotation As String = ""
  response = CInt(InputBox("Enter a number from 1 to 3."))
  Do While (response < 1) Or (response > 3)
    response = CInt(InputBox("Enter a number from 1 to 3."))
  Loop
  Select Case response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



Enter a number from 1 to 3.

[Type 3 into the box and click on the *OK* button.]



That's all folks.

Do loops often are used to process data input from a file or from the user.

Example 3 The following program finds the average of a sequence of numbers entered by the user from input dialog boxes. The user should type in the number −1 to indicate the end of data entry. Since the first input dialog box appears before the loop is entered, there is the possibility that the entire loop will be skipped.

```
Private Sub btnCompute_Click(...) Handles btnCompute.Click
  Dim num As Double = 0
  Dim count As Integer = 0
  Dim sum As Double = 0
  Dim prompt As String = "Enter a nonnegative number.  " &
                         "Enter −1 to terminate entering numbers."
  num = CDbl(InputBox(prompt))
  Do While num <> −1
    count += 1
    sum += num
    num = CDbl(InputBox(prompt))
    Loop
    If count > 0 Then
      MessageBox.Show("Average: " & sum / count)
    Else
      MessageBox.Show("No numbers were entered.")
    End If
  End Sub
```

[Run, click on the button, and respond to the requests for input with 80, 90, and −1. The following is displayed in the message box.]

```
Average: 85
```

In Example 3, the variable *count* is called a **counter variable**, the variable *sum* is called an **accumulator variable**, the number −1 is called a **sentinel value**, and the loop is referred to as having **sentinel-controlled repetition**.

Posttest Form of a Do Loop

In Examples 1 and 2, the condition was checked at the top of the loop—that is, before the statements were executed. Alternatively, the condition can be checked at the bottom of the loop when the Loop statement 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.

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

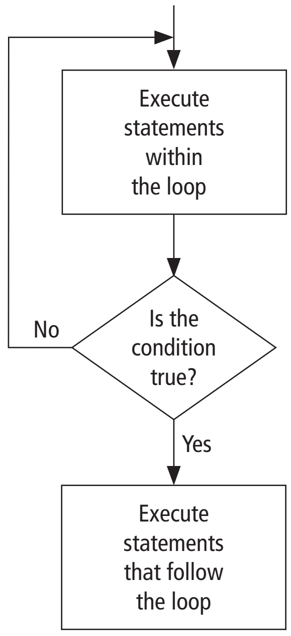

Execute statements within the loop
No
Is the condition true?
Yes
Execute statements that follow the loop

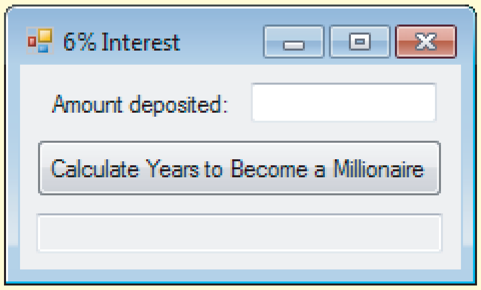
FIGURE 6.2 Pseudocode and flowchart for a Do loop with the condition tested at the bottom.

Example 4 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.

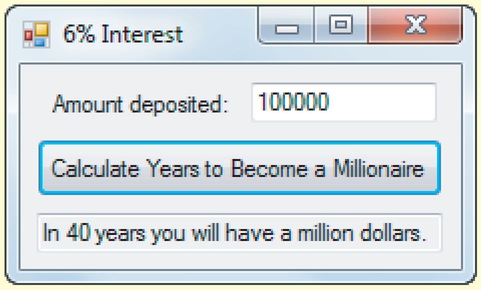
Example 5 Suppose you deposit money into a savings account and let it accumulate at 6% interest compounded annually. The following program determines when you will be a millionaire:



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 btnCalculate_Click(...) Handles btnCalculate.Click
  'Compute years required to become a millionaire
  Dim balance As Double, numYears As Integer
  balance = CDbl(txtAmount.Text)
  Do While 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 click on the button.]



```
100000
In 40 years you will have a million dollars.
```

Comments

1. Be careful to avoid infinite loops—that is, loops that are never exited. The following loop is infinite, because the condition "balance < 1000" will always be true. This logic error can be avoided by initializing *intRate* with a value greater than 0.

```
Private Sub btnButton_Click(...) Handles btnButton.Click
  '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
```

   Important: While an infinite loop is executing, the program can be terminated by clicking on the *Stop Debugging* button on the Toolbar.

2. Visual Basic provides a way to break out of a Do loop before the loop condition is met. When the statement ~~`Exit Do`~~ is encountered in the body of a loop, execution jumps immediately to the statement following the Loop statement.
3. A variable declared inside a Do loop has block-level scope; that is, the variable cannot be referred to by code outside of the loop.
4. Visual Basic allows the use of the words "While" and "Until" at either the top or bottom of a Do loop. For instance, the fourth line in the program in Example 1 can be replaced with

```
Do Until num > 7
```

   and the fifth line of the program in Example 4 can be replaced with

```
Loop While (response < 1) Or (response > 3)
```

Practice Problem 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 that the loop will execute 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 on.

1.
```
Private Sub btnDisplay_Click(...) Handles btnDisplay.Click
  Dim num As Integer = 3
  Do While num < 15
    num += 5
  Loop
  txtOutput.Text = CStr(num)
End Sub
```

2.
```
Private Sub btnDisplay_Click(...) Handles btnDisplay.Click
  Dim num As Integer = 3
  Do
    num = 2 * num
  Loop Until num > 15
  txtOutput.Text = CStr(num)
End Sub
```

3.
```
Private Sub btnDisplay_Click(...) Handles btnDisplay.Click
  Dim total As Double = 0
  Dim num As Integer = 1
  Do While num < 5
    total += num
    num += 1
  Loop
  txtOutput.Text = CStr(total)
End Sub
```

4.
```
Private Sub btnDisplay_Click(...) Handles btnDisplay.Click
  Dim total As Double = 0
  Dim num As Integer = 1
  Do
    total += num
    num += 1
  Loop Until num >= 5
  txtOutput.Text = CStr(total)
End Sub
```

5.
```
Private Sub btnCompute_Click(...) Handles btnCompute.Click
  Dim num As Double = 0
  Dim max As Double = -1
  Dim prompt As String = "Enter a nonnegative number.  " &
                         "Enter -1 to terminate entering numbers."
```

```
    num = CDbl(InputBox(prompt))
    Do While num >= 0
      If num > max Then
        max = num
      End If
      num = CDbl(InputBox(prompt))
    Loop
    If max <> -1 Then
      MessageBox.Show("Maximum number: " & max)
    Else
      MessageBox.Show("No numbers were entered.")
    End If
  End Sub
```

(Assume that the responses are 4, 7, 3, and −1.)

6.
```
  Private Sub btnDisplay_Click(...) Handles btnDisplay.Click
    Dim numTries As Integer
    Dim yr As Integer
    Dim msg As String = "In what year did the Beatles invade the U.S.?"
    Do
      numTries += 1
      yr = CInt(InputBox(msg, "Try #" & numTries))
      Select Case yr
        Case 1964
          MessageBox.Show("They appeared on the Ed Sullivan show in " &
                          "February 1964." & " You answered the question " &
                          "correctly in " & numTries & " tries.", "Correct")
        Case Is < 1964
          MessageBox.Show("Later than " & yr & ".")
        Case Is > 1964
          MessageBox.Show("Earlier than " & yr & ".")
      End Select
    Loop Until (yr = 1964) Or (numTries = 7)
    If yr <> 1964 Then
      MessageBox.Show("Your 7 tries are up, the answer is 1964.", "Sorry")
    End If
  End Sub
```

(Assume that the responses are 1950, 1970, and 1964.)

In Exercises 7 through 10, identify the errors.

7.
```
  Private Sub btnDisplay_Click(...) Handles btnDisplay.Click
    Dim q As Double = 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`

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 in a list box. Entries in the table should range from 10 to 95 degrees Celsius in increments of 5 degrees. ***Note:*** The formula $f = (9/5 * c) + 32$ converts Celsius to Fahrenheit.

24. 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% 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.

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

26. Write a program that finds the smallest number in a sequence of nonnegative numbers entered by the user from input dialog boxes. The user should be told to type in the number −1 to indicate that the entire sequence has been entered.

27. Write a program that finds the range of a sequence of nonnegative numbers entered by the user from input dialog boxes. (The *range* is the difference between the largest and the smallest numbers in the sequence.) The user should be told to type in the number −1 to indicate that the entire sequence has been entered.

In Exercises 28 and 29, write a program corresponding to the flowchart.

28. The flowchart in Fig. 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.

VideoNote
Sieve of Eratosthenes (Homework)

29. The flowchart in Fig. 6.4 on the next page 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.

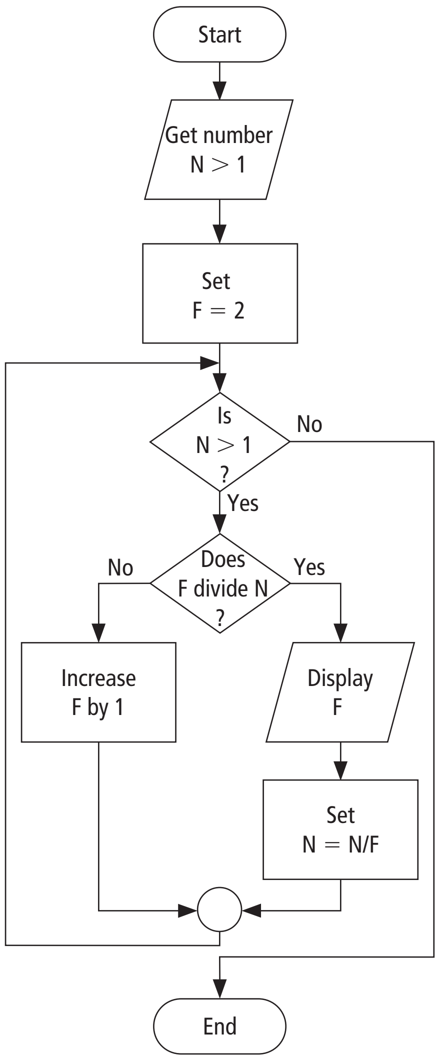


FIGURE 6.3 Prime factors.

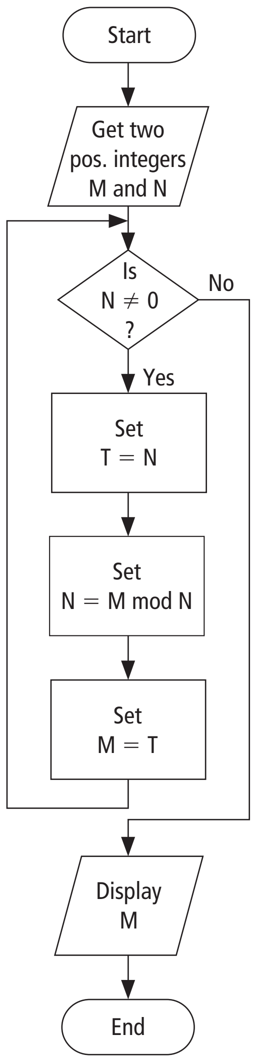
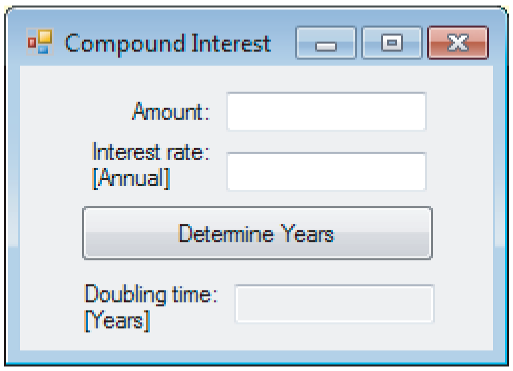


FIGURE 6.4 Greatest common divisor.

30. 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. 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

In Exercises 31 through 38, write a program to answer the question.

31. A person born in 1980 can claim, "I will be *x* years old in the year *x* squared." What is the value of *x*?

32. The world population reached 6.83 billion people in January 2010 and was growing at the rate of 1.12% each year. Assuming that the population will continue to grow at the same rate, when will the population reach 10 billion?

33. Strontium-90, a radioactive element that is part of the fallout from nuclear explosions, has a half-life of 28 years. This means that a given quantity of strontium-90 will emit radioactive particles and decay to one-half its size every 28 years. How many years are required for 100 grams of strontium-90 to decay to less than 1 gram?

34. The *consumer price index* (CPI) indicates the average price of a fixed basket of goods and services. It is customarily taken as a measure of inflation and is frequently used to adjust pensions. The CPI was 9.9 in July 1913, was 100 in July 1983, and was 215.35 in July 2009. This means that \$9.90 in July 1913 had the same purchasing power as \$100.00 in July 1983, and the same purchasing power as \$215.35 in July 2009. In 2009, the CPI fell for the first time since 1955. However, for most of the preceding 15 years it had grown at an average rate of 2.5% per year. Assuming that the CPI will rise at 2.5% per year in the future, in what year will the July CPI have at least doubled from its July 2009 level? ***Note:*** Each year, the CPI will be 1.025 times the CPI for the previous year.

35. When you borrow money to buy a house or a car, the loan is paid off with a sequence of equal monthly payments incorporating a stated annual interest rate compounded monthly. The amount borrowed is called the *principal*. If the annual interest rate is 6% (or .06), then the monthly interest rate is .06/12 = .005. At any time, the *balance* of the loan is the amount still owed. The balance at the end of each month is calculated as the balance at the end of the previous month, plus the interest due on that balance, and minus the monthly payment. For instance, with an annual interest rate of 6%,

$$\begin{aligned}[\text{new balance}] &= [\text{previous balance}] + .005 \cdot [\text{previous balance}] - [\text{monthly payment}] \\ &= 1.005 \cdot [\text{previous balance}] - [\text{monthly payment}].\end{aligned}$$

Suppose you borrow \$15,000 to buy a new car at 6% interest compounded monthly and your monthly payment is \$290.00. After how many months will the car be half paid off? That is, after how many months will the balance be less than half the principal?

36. An *annuity* is a sequence of equal periodic payments. One type of annuity, called a *savings plan*, consists of monthly payments into a savings account in order to generate money for a future purchase. Suppose you decide to deposit \$100 at the end of each month into a savings account paying 3% interest compounded monthly. The monthly interest rate will be .03/12 or .0025, and the balance in the account at the end of each month will be computed as

$$[\text{balance at end of month}] = (1.0025) \cdot [\text{balance at end of previous month}] + 100.$$

After how many months will there be more than \$3000 in the account, and how much money will be in the account at that time?

37. An *annuity* is a sequence of equal periodic payments. For one type of annuity, a large amount of money is deposited into a bank account and then a fixed amount is withdrawn each month. Suppose you deposit \$10,000 into such an account paying 3.6% interest compounded monthly, and then withdraw \$600 at the end of each month. The monthly interest rate will be .036/12 or .003, and the balance in the account at the end of each month will be computed as

$$[\text{balance at end of month}] = (1.003) \cdot [\text{balance at end of previous month}] - 600.$$

After how many months will the account contain less than \$600, and what will be the amount in the account at that time?

38. Redo Exercise 37 with the amount of money deposited being input by the user.

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"`.

6.2 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 they 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
  For i As Integer = 1 To 5
    lstTable.Items.Add(i & "  " & i ^ 2)
  Next
End Sub
```

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

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

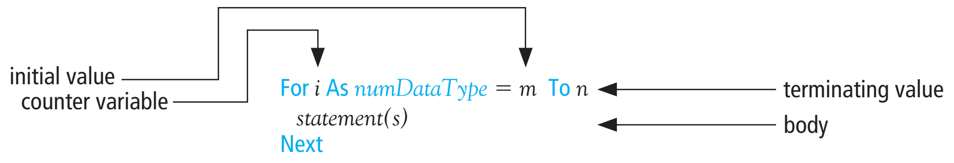
A similar program written with a Do loop is as follows.

```
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
```

VideoNote
For . . . Next loops

General Form of a For . . . Next Loop

In general, a portion of a program of the form



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 declares a numeric variable, called the **counter variable**, that is initialized and then automatically changes after