2. The two arguments in **`FindLetter(word, num)`** are in the wrong order. Since the two parameters in the header for the Function procedure have types Integer and String, in that order, the arguments must have the same types and order when the Function procedure is called. The function call should be **`FindLetter(num, word)`**. Visual Basic matches arguments to parameters based on their positions, not on their names.

5.2 Sub Procedures, Part I

Sub procedures share several features with Function procedures.

- Both are written as a separate block of code that can be called to perform a specific task.
- Both are used to break complex problems into small problems.
- Both are used to eliminate repetitive code.
- Both can be reused in other programs.
- Both make a program easier to read by separating it into logical units.
- Both have parameters that are declared in a header.

Sub procedures, however, do not return a value associated with their name. The most common uses of Sub procedures are to receive input, process input, or display output.

Defining and Calling Sub Procedures

VideoNote
Sub procedures

Sub procedures are defined by blocks of the form

```
Sub ProcedureName(ByVal par1 As Type1,
                  ByVal par2 As Type2,
                    :
                  ByVal parN As TypeN)
  statement(s)
End Sub
```

In the block above, one or more of the ByVals can be replaced with the keyword ByRef. (The use of ByRef will be discussed in the next section.)

Like Function procedure names, the names of Sub procedures must conform to the rules for naming variables. By convention, Sub procedure names begin with an uppercase letter and describe its purpose. In this section all parameters will be preceded by the keyword ByVal. The primary difference will be that Sub procedures will perform some task (such as displaying output) rather than return a value.

Sub procedures are called by statements of the form

```
ProcedureName(arg1, arg2, ..., argN)
```

When a Sub procedure is called, the value of each argument is assigned to the corresponding parameter, the statement(s) inside the procedure block are carried out, and execution continues with the statement following the calling statement.

Here is an example of a Sub procedure.

```
Sub DisplaySum(ByVal num1 As Double, ByVal num2 As Double)
  Dim z As Double
  z = num1 + num2
  lstOutput.Items.Add(z)
End Sub
```

When a statement such as

```
DisplaySum(3, 4)
```

is executed in an event procedure, the number 3 is assigned to the parameter *num1*, the number 4 is assigned to the parameter *num2*, and the three statements inside the Sub procedure block are carried out. As a result, the number 7 is displayed in the list box. We say that the numbers 3 and 4 are **passed** to the Sub procedure. See Fig. 5.3.

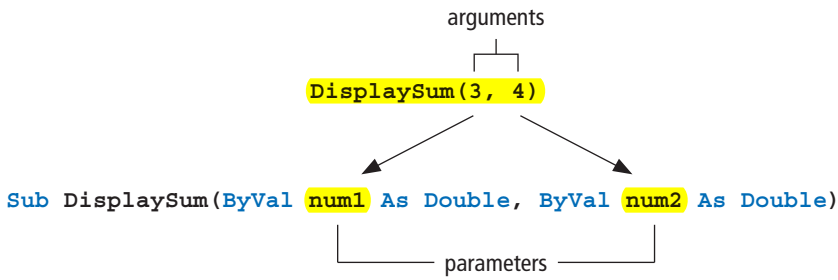


FIGURE 5.3 **Passing arguments to a procedure.**

Variables and Expressions as Arguments

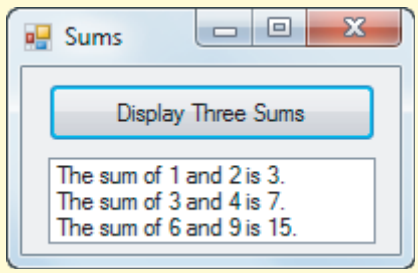
Just as with function calls, the arguments in Sub procedure calls can be literals (as in Fig. 5.3), variables, or expressions.

Example 1 The following program calls an expanded version of the Sub procedure DisplaySum three times. The first time the arguments are literals, the second time the arguments are variables, and the third time the arguments are expressions. In the second call of DisplaySum, the values of the variables are passed to the Sub procedure. In the third call, the expressions are evaluated and the resulting numbers are passed to the Sub procedure.

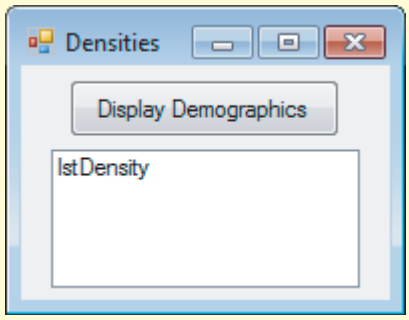
```
Private Sub btnAddNumbers_Click(...) Handles btnAddNumbers.Click
  DisplaySum(1, 2)
  Dim x As Double = 3
  Dim y As Double = 4
  DisplaySum(x, y)
  DisplaySum(2 * x, y + 5)
End Sub

Sub DisplaySum(ByVal num1 As Double, ByVal num2 As Double)
  Dim z As Double
  z = num1 + num2
  lstOutput.Items.Add("The sum of " & num1 & " and " & num2 &
                      " is " & z & ".")
End Sub
```

[Run, and click on the button.]



Example 2 The following program passes a string and two numbers to a Sub procedure. When the Sub procedure is first called, the string parameter *state* is assigned the value "Hawaii", and the numeric parameters *pop* and *area* are assigned the values 1275194 and 6471, respectively. The Sub procedure then uses these parameters to carry out the task of calculating the population density of Hawaii. The second calling statement assigns different values to the parameters.

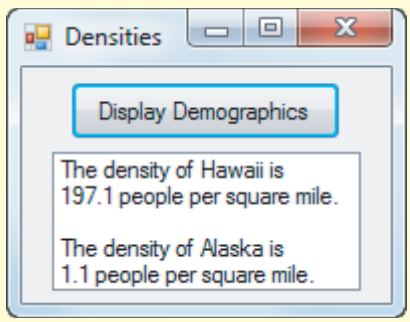


OBJECT	PROPERTY	SETTING
frmDensities	Text	Densities
btnDisplay	Text	Display Demographics
lstDensity		

```
Private Sub btnDisplay_Click(...) Handles btnDisplay.Click
  'Calculate the population densities of states
  lstDensity.Items.Clear()
  Dim state As String, pop As Double, area As Double
  state = "Hawaii"
  pop = 1275194
  area = 6471
  CalculateDensity(state, pop, area)
  lstDensity.Items.Add("")
  state = "Alaska"
  pop = 663661
  area = 591000
  CalculateDensity(state, pop, area)
End Sub

Sub CalculateDensity(ByVal state As String,
                     ByVal pop As Double, ByVal area As Double)
  'The density (number of people per square mile)
  'will be displayed rounded to one decimal place.
  Dim density As Double
  density = pop / area
  lstDensity.Items.Add("The density of " & state & " is")
  lstDensity.Items.Add(FormatNumber(density, 1) & " people per square mile.")
End Sub
```

[Run, and then click on the button.]



Notice that in the calling statement

```
CalculateDensity(state, pop, area)
```

the variable types have the order String, Double, and Double; the same types and order as in the Sub procedure header. This order is essential. For instance, the calling statement cannot be written as

```
CalculateDensity(pop, area, state)
```

In Example 2 the arguments and parameters have the same name. Using same names sometimes makes a program easier to read. However, arguments and their corresponding parameters often have different names. What matters is that the *order*, *number*, and *types* of the arguments and parameters match. For instance, the following code is a valid revision of the btnDisplay_Click event procedure in Example 2. (Figure 5.4 shows how arguments are passed to parameters with this code.)

```
Private Sub btnDisplay_Click(...) Handles btnDisplay.Click
  'Calculate the population densities of states.
  lstDensity.Items.Clear()
  Dim s As String, p As Double, a As Double
  s = "Hawaii"
  p = 1275194
  a = 6471
  CalculateDensity(s, p, a)
  lstDensity.Items.Add("")
  s = "Alaska"
  p = 663661
  a = 591000
  CalculateDensity(s, p, a)
End Sub
```

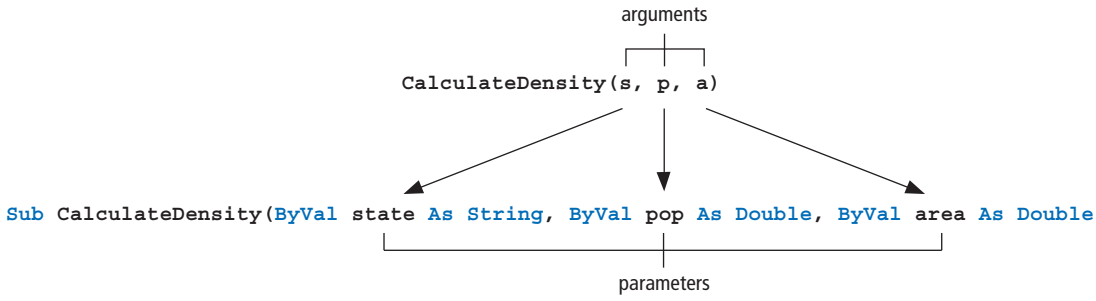


FIGURE 5.4 **Passing arguments to a procedure.**

Sub Procedures Having No Parameters

Sub procedures, like Function procedures, are not required to have any parameters. A parameterless Sub procedure can be used to give instructions or provide a description of a program.

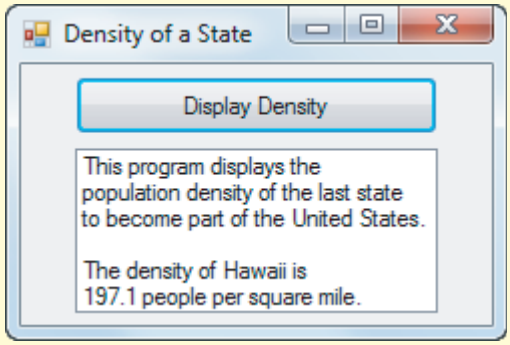
Example 3 The following variation of Example 2 gives the population density of a single state. The parameterless Sub procedure DescribeTask gives an explanation of the program.

```
Private Sub btnDisplay_Click(...) Handles btnDisplay.Click
  DescribeTask()
  CalculateDensity("Hawaii", 1275194, 6471)
End Sub

Sub DescribeTask()
  lstOutput.Items.Clear()
  lstOutput.Items.Add("This program displays the")
  lstOutput.Items.Add("population density of the last state")
  lstOutput.Items.Add("to become part of the United States.")
End Sub

Sub CalculateDensity(ByVal state As String,
                     ByVal pop As Double, ByVal area As Double)
  Dim density As Double
  density = pop / area
  lstDensity.Items.Add("")
  lstDensity.Items.Add("The density of " & state & " is")
  lstDensity.Items.Add(FormatNumber(density, 1) & " people per square mile.")
End Sub
```

[Run, and then click on the button.]



Sub Procedures Calling Other Sub Procedures

A Sub procedure can call another Sub procedure. If so, after the End Sub statement at the end of the called Sub procedure is reached, execution continues with the line in the calling Sub procedure following the calling statement.

Example 4 In the following program, the Sub procedure FirstPart calls the Sub procedure SecondPart. After the statements in SecondPart are executed, execution continues with the remaining statements in the Sub procedure FirstPart before returning to the event procedure. The form contains a button and a list box.

```
Private Sub btnDisplay_Click(...) Handles btnDisplay.Click
  'Demonstrate Sub procedure calling other Sub procedures
  FirstPart()
```

```
    lstOutput.Items.Add(4 & " from event procedure")
End Sub

Sub FirstPart()
    lstOutput.Items.Add(1 & " from FirstPart")
    SecondPart()
    lstOutput.Items.Add(3 & " from FirstPart")
End Sub

Sub SecondPart()
    lstOutput.Items.Add(2 & " from SecondPart")
End Sub
```

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

```
1 from FirstPart
2 from SecondPart
3 from FirstPart
4 from event procedure
```

Comments

1. Sub procedures allow programmers to focus on the main flow of a complex task and defer the details of implementation. Modern programs use them liberally. This method of program construction is known as **modular** or **top-down** design. As a rule, a Sub procedure should perform only one task, or several closely related tasks, and should be kept relatively small.
2. The first line inside a Sub procedure is often a comment statement describing the task performed by the Sub procedure. If necessary, several comment statements are devoted to this purpose. Conventional programming practice also recommends that all variables used by the Sub procedure be listed in comment statements with their meanings. In this text, we give several examples of this practice, but adhere to it only when the variables are especially numerous or lack descriptive names.
3. In Section 5.1, we saw that Word Completion and Parameter Info help us write a function call. These IntelliSense features provide the same assistance for Sub procedure calls. (Of course, Word Completion and Parameter Info work only when the Sub procedure has already been created.) See Fig. 5.5.

```
Private Sub btnAddNumbers_Click(ByVal sender As System.Object,
    DisplaySum(1, 2)
    Dim x As Double = 3
    Dim y As Double = 4
    DisplaySum(
    DisplaySum (num1 As Double, num2 As Double)
```

FIGURE 5.5 The Parameter Info help feature.

4. In early versions of Basic, statements that called Sub procedures had to be written in the form

```
Call ProcedureName(arg1, arg2, ... , argN)
```

Therefore, statements that call Sub procedures are often referred to as **Call statements**.

Practice Problems 5.2

1. What is the difference between an event procedure and a Sub procedure?
2. What is wrong with the following code?

```
Private Sub btnDisplay_Click(...) Handles btnDisplay.Click
  Dim phone As String
  phone = mtbPhoneNum.Text
  AreaCode(phone)
End Sub

Sub AreaCode()
  txtOutput.Text = "Your area code is " & phone.Substring(0, 3)
End Sub
```

EXERCISES 5.2

In Exercises 1 through 20, determine the output displayed when the button is clicked.

1.
```
Private Sub btnDisplay_Click(...) Handles btnDisplay.Click
  Piano(88)
End Sub

Sub Piano(ByVal num As Integer)
  txtOutput.Text = num & " keys on a piano"
End Sub
```

2.
```
Private Sub btnDisplay_Click(...) Handles btnDisplay.Click
  'Opening line of Moby Dick
  FirstLine("Ishmael")
End Sub

Sub FirstLine(ByVal name As String)
  'Display first line
  txtOutput.Text = "Call me " & name & "."
End Sub
```

3.
```
Private Sub btnDisplay_Click(...) Handles btnDisplay.Click
  Dim color As String
  color = InputBox("What is your favorite color?")
  Flattery(color)
End Sub

Sub Flattery(ByVal color As String)
  txtOutput.Text = "You look dashing in " & color & "."
End Sub
```

(Assume the response is *blue*.)

4.
```
Private Sub btnDisplay_Click(...) Handles btnDisplay.Click
  Dim num As Double = 144
  Gross(num)
End Sub
```

```
Sub Gross(ByVal amount As Double)
  txtOutput.Text = amount & " items in a gross"
End Sub
```

5.
```
Private Sub btnDisplay_Click(...) Handles btnDisplay.Click
  Dim hours As Double
  hours = 24
  Minutes(60 * hours)
End Sub

Sub Minutes(ByVal num As Double)
  txtOutput.Text = num & " minutes in a day"
End Sub
```

6.
```
Private Sub btnDisplay_Click(...) Handles btnDisplay.Click
  Dim states, senators As Double
  states = 50
  senators = 2
  Senate(states * senators)
End Sub

Sub Senate(ByVal num As Double)
  txtBox.Text = "The number of U.S. Senators is " & num
End Sub
```

7.
```
Private Sub btnDisplay_Click(...) Handles btnDisplay.Click
  Question()
  Answer()
End Sub

Sub Answer()
  lstOutput.Items.Add("Because they were invented in the northern")
  lstOutput.Items.Add("hemisphere where sundials go clockwise.")
End Sub

Sub Question()
  lstOutput.Items.Add("Why do clocks run clockwise?")
  lstOutput.Items.Add("")
End Sub
```

8.
```
Private Sub btnDisplay_Click(...) Handles btnDisplay.Click
  Answer()
  Question()
End Sub

Sub Answer()
  lstOutput.Items.Add("The answer is 9W.")
  lstOutput.Items.Add("What is the question?")
End Sub

Sub Question()
  'Note: "Wagner" is pronounced "Vagner"
```

```
    lstOutput.Items.Add("Do you spell your name with a V,")
    lstOutput.Items.Add("Mr. Wagner?")
End Sub
```

9.
```
Private Sub btnDisplay_Click(...) Handles btnDisplay.Click
    'Beginning of Tale of Two Cities
    Times("best")
    Times("worst")
End Sub

Sub Times(ByVal word As String)
    'Display sentence
    lstOutput.Items.Add("It was the " & word & " of times.")
End Sub
```

10.
```
Private Sub btnDisplay_Click(...) Handles btnDisplay.Click
    'Sentence using number, thing, and place
    Sentence(168, "hour", "a week")
    Sentence(76, "trombone", "the big parade")
End Sub

Sub Sentence(ByVal num As Double, ByVal thing As String,
             ByVal where As String)
    lstOutput.Items.Add(num & " " & thing & "s in " & where)
End Sub
```

11.
```
Private Sub btnDisplay_Click(...) Handles btnDisplay.Click
    'The fates of Henry the Eighth's six wives
    CommonFates()
    lstOutput.Items.Add("died")
    CommonFates()
    lstOutput.Items.Add("survived")
End Sub

Sub CommonFates()
    'The most common fates
    lstOutput.Items.Add("divorced")
    lstOutput.Items.Add("beheaded")
End Sub
```

12.
```
Private Sub btndisplay_Click(...) Handles btndisplay.Click
    Dim pres, college As String
    pres = "Bush"
    college = "Yale"
    PresAlmaMater(pres, college)
    pres = "Obama"
    college = "Columbia"
    PresAlmaMater(pres, college)
End Sub

Sub PresAlmaMater(ByVal pres As String, ByVal college As String)
    lstOutput.Items.Add("President " & pres & " is a graduate of " &
                        college & ".")
End Sub
```

13.
```
Private Sub btnDisplay_Click(...) Handles btnDisplay.Click
  HowMany(24)
  lstOutput.Items.Add("a pie.")
End Sub

Sub HowMany(ByVal num As Integer)
  What(num)
  lstOutput.Items.Add("baked in")
End Sub

Sub What(ByVal num As Integer)
  lstOutput.Items.Add(num & " blackbirds")
End Sub
```

14.
```
Private Sub btnDisplay_Click(...) Handles btnDisplay.Click
  'Good advice to follow
  Advice()
End Sub

Sub Advice()
  lstOutput.Items.Add("Keep cool, but don't freeze.")
  Source()
End Sub

Sub Source()
  lstOutput.Items.Add("Source: A jar of mayonnaise.")
End Sub
```

15.
```
Private Sub btnDisplay_Click(...) Handles btnDisplay.Click
  Dim word As String, num As Integer
  word = "Visual Basic"
  num = 6
  FirstPart(word, num)
End Sub

Sub FirstPart(ByVal term As String, ByVal digit As Integer)
  txtOutput.Text = "The first " & digit & " letters are " &
                   term.Substring(0, digit) & "."
End Sub
```

16.
```
Private Sub btnDisplay_Click(...) Handles btnDisplay.Click
  Dim d As Date = Today
  DisplayTypeOfDay(d)
End Sub

Sub DisplayTypeOfDay(ByVal d As Date)
  If IsWeekendDay(d) Then
    txtOutput.Text = "Today is a weekend day."
  Else
    txtOutput.Text = "Today is a weekday."
  End If
End Sub
```

```
Function IsWeekendDay(ByVal d As Date) As Boolean
  Dim when As String = FormatDateTime(d, DateFormat.LongDate)
  If when.StartsWith("Saturday") Or when.StartsWith("Sunday") Then
    Return True
  Else
    Return False
  End If
End Function
```

17.
```
Private Sub btnDisplay_Click() Handles btnDisplay.Click
  Dim cost As Double = 250
  DisplayBill(cost, ShippingCost(cost))
End Sub

Function ShippingCost(ByVal costOfGoods As Double) As Double
  Select Case costOfGoods
    Case Is < 100
      Return 10
    Case Is < 500
      Return 15
    Case Else
      Return 20
  End Select
End Function

Sub DisplayBill(ByVal cost As Double, ByVal addedCost As Double)
  lstOutput.Items.Add("Cost: " & FormatCurrency(cost))
  lstOutput.Items.Add("Shipping cost: " & FormatCurrency(addedCost))
  lstOutput.Items.Add("Total cost: " & FormatCurrency(cost + addedCost))
End Sub
```

18.
```
Private Sub btnDisplay_Click() Handles btnDisplay.Click
  Dim language As String = "Visual Basic"
  ShowWord(language)
End Sub

Sub ShowWord(ByVal word As String)
  If word.Length < 5 Then
    txtOutput.ForeColor = Color.Red
  Else
    txtOutput.ForeColor = Color.Blue
  End If
  txtOutput.Text = word
End Sub
```

19.
```
Private Sub btnDisplay_Click() Handles btnDisplay.Click
  Dim grade = CDbl(InputBox("What is your numeric grade?", "Grade"))
  ShowResult(grade)
End Sub

Sub ShowResult(ByVal grade As Double)
  If PassedExam(grade) Then
    txtOutput.Text = "You passed with a grade of " & grade & "."
```

```
  Else
    txtOutput.Text = "You failed the exam."
  End If
End Sub

Function PassedExam (ByVal grade As Double) As Boolean
  Select Case grade
    Case Is >= 60
      Return True
    Case Else
      Return False
  End Select
End Function
```

(Assume the response is 92.)

20.
```
Private Sub btnDisplay_Click() Handles btnDisplay.Click
  Dim anyDate As Date
  anyDate = CDate(InputBox("Input a date. (mm/dd/yyyy)"))
  ShowCentury(anyDate)
End Sub

Sub ShowCentury(ByVal anyDate As Date)
  Select Case anyDate
    Case Is >= #1/1/2000#
      txtOutput.Text = "twenty-first century"
    Case Is >= #1/1/1900#
      txtOutput.Text = "twentieth century"
    Case Else
      txtOutput.Text = "prior to the twentieth century"
  End Select
End Sub
```

(Assume the response is 6/5/1955.)

In Exercises 21 through 24, find the errors.

21.
```
Private Sub btnDisplay_Click(...) Handles btnDisplay.Click
  Dim n As Integer = 5
  Alphabet()
End Sub

Sub Alphabet(ByVal n As Integer)
  txtOutput.Text = "abcdefghijklmnopqrstuvwxyz".Substring(0, n)
End Sub
```

22.
```
Private Sub btnDisplay_Click(...) Handles btnDisplay.Click
  Dim word As String, number As Double
  word = "seven"
  number = 7
  Display(word, number)
End Sub
```

```
Sub Display(ByVal num As Double, ByVal term As String)
  txtOutput.Text = num & " " & term
End Sub
```

23.
```
Private Sub btnDisplay_Click(...) Handles btnDisplay.Click
  Dim name As String
  name = InputBox("Name")
  Handles(name)
End Sub

Sub Handles(ByVal moniker As String)
  txtOutput.Text = "Your name is " & moniker
End Sub
```

24.
```
Private Sub btnDisplay_Click(...) Handles btnDisplay.Click
  Dim num As Integer = 2
  Tea(num)
End Sub

Sub Tea()
  txtOutput.Text = "Tea for " & num
End Sub
```

In Exercises 25 through 28, rewrite the program with the output performed by a call to a Sub procedure.

25.
```
Private Sub btnDisplay_Click(...) Handles btnDisplay.Click
  'Display a lucky number
  Dim num As Integer = 7
  txtOutput.Text = num & " is a lucky number."
End Sub
```

26.
```
Private Sub btnDisplay_Click(...) Handles btnDisplay.Click
  'Greet a friend
  Dim name As String = "Jack"
  txtOutput.Text = "Hi, " & name
End Sub
```

27.
```
Private Sub btnDisplay_Click(...) Handles btnDisplay.Click
  'Information about trees
  Dim tree As String, ht As Double
  tree = "redwood"
  ht = 362
  lstBox.Items.Add("The tallest " & tree &
                   " tree in the U.S. is " & ht & " feet.")
  tree = "pine"
  ht = 223
  lstBox.Items.Add("The tallest " & tree &
                   " tree in the U.S. is " & ht & " feet.")
End Sub
```

28.
```
Private Sub btnDisplay_Click(...) Handles btnDisplay.Click
  Dim city As String, salary As Double
  lstOutput.Items.Clear()
  city = "San Jose"
```

```
    salary = 83089
    lstOutput.Items.Add("In 2008, the average salary for " & city &
                        " residents was " & FormatCurrency(salary, 0) & ".")
    city = "Hartford"
    salary = 46000
    lstOutput.Items.Add("In 2008, the average salary for " & city &
                        " residents was " & FormatCurrency(salary, 0) & ".")
End Sub
```

In Exercises 29 through 32, write a program that displays the output shown in a list box. The last two lines of the output should be displayed by one or more Sub procedures using data passed by variables from an event procedure.

29. (Assume that the following is displayed.)

```
According to a 2008 survey of college freshmen taken by the Higher
Education Research Institute:
16.7 percent said they intend to major in business.
1 percent said they intend to major in computer science.
```

30. (Assume that the current date is 12/31/2010, the label for txtBox reads "What is your date of birth?", and the user enters 2/3/1984 into txtBox before btnDisplay is clicked.)

```
You are now 26 years old.
You have lived for 9824 days.
```

31. (Assume that the label for txtBox reads "What is your favorite number?", and the user types 7 into txtBox before btnDisplay is clicked.)

```
The sum of your favorite number with itself is 14.
The product of your favorite number with itself is 49.
```

32. (Assume that the following is displayed.)

```
In a recent year,
823 thousand college students took a course in Spanish
206 thousand college students took a course in French
```

33. Write a program to display three verses of "Old McDonald Had a Farm." The primary verse, with variables substituted for the animals and sounds, should be contained in a Sub procedure. The program should pass the following animal and sound pairs to the Sub procedure: lamb, baa; duck, quack; firefly, blink. The first verse of the output should be

```
Old McDonald had a farm. Eyi eyi oh.
And on his farm he had a lamb. Eyi eyi oh.
With a baa baa here, and a baa baa there.
Here a baa, there a baa, everywhere a baa baa.
Old McDonald had a farm. Eyi eyi oh.
```

34. Write a program to compute tips for services rendered. The program should request the person's occupation, the amount of the bill, and the percentage tip as input and pass this information to a Sub procedure to display the person and the tip. A sample run is shown in Fig. 5.6.

35. Write a program that requests three grades as input and then passes the three grades to a Sub procedure that determines and displays the highest two grades. See Fig. 5.7.

36. Write a program that requests a student's first name, last name and the numeric grades on three exams, and then uses a Sub procedure to display the student's name and semester

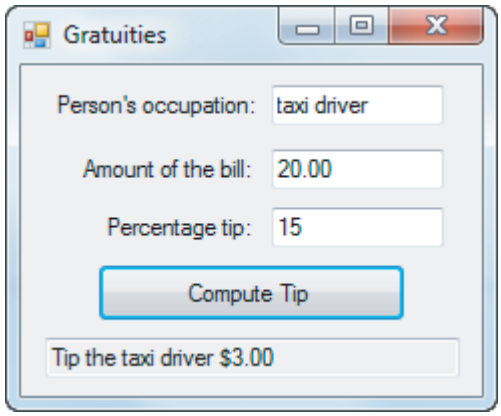


FIGURE 5.6 Sample run of Exercise 34.

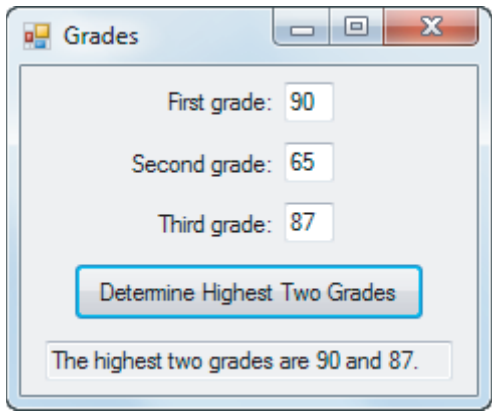


FIGURE 5.7 Sample run of Exercise 35.

grade (A, B, C, D, or F). A Function procedure (called by the Sub procedure) should be used to calculate the semester grade. The lowest grade should be dropped, the semester average should be rounded to the nearest whole number, and the semester grade should be assigned using the following criteria: 90–100 (A), 80–89 (B), ... See Fig. 5.8.

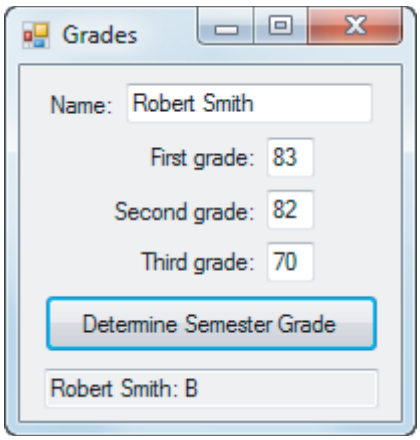


FIGURE 5.8 Sample run of Exercise 36.

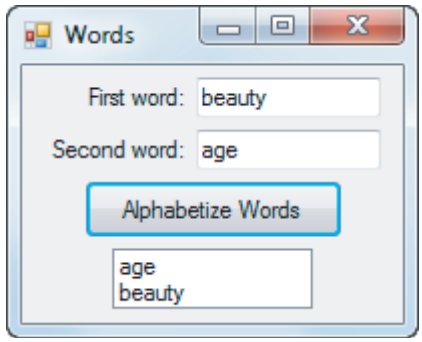


FIGURE 5.9 Sample run of Exercise 37.

37. Write a program that requests two words as input and then passes the words to a Sub procedure that displays the words in alphabetical order. See Fig. 5.9.

38. Write a program that asks a quiz show contestant to select one of the numbers 1, 2, or 3 and then calls a Sub procedure that asks the question having that number and requests the answer. The Sub procedure should then call another Sub procedure to tell the contestant if the answer is correct. Use the following three questions:

1. Who was the only living artist to have his work displayed in the Grand Gallery of the Louvre?
2. Who said, "Computers are useless. They can only give you answers."?
3. By what name is Pablo Blasio better known?

Note: These questions have the same answer, Pablo Picasso.

Solutions to Practice Problems 5.2

1. The header of an event procedure has parameters (such as e and sender) that are provided automatically by Visual Basic, and the procedure is invoked when an event is raised. On the other hand, a Sub procedure is invoked by a line of code containing the name of the Sub procedure.
2. The statement `Sub AreaCode()` must be replaced by `Sub AreaCode(ByVal phone As String)`. Whenever a value is passed to a Sub procedure, the Sub statement must provide a parameter to receive the value.

5.3 Sub Procedures, Part II

In the previous section values were passed to Sub procedures. In this section we show how to pass values back from Sub procedures.

Passing by Value

In Section 5.2, all parameters appearing in Sub procedures were preceded by the word ByVal, which stands for "By Value." When a variable is passed to such a parameter, we say that the variable is "passed by value." A variable that is passed by value will retain its original value after the Sub procedure terminates—regardless of what changes are made to the value of the corresponding parameter inside the Sub procedure. Example 1 illustrates this feature.

Example 1 The following program illustrates the fact that changes to the value of a parameter passed by value have no effect on the value of the argument in the calling statement.

```
Private Sub btnDisplay_Click(...) Handles btnDisplay.Click
  'Illustrate that a change in value of parameter
  'does not alter the value of the argument
  Dim amt As Double = 2
  lstResults.Items.Add(amt & " from event procedure")
  Triple(amt)
  lstResults.Items.Add(amt & " from event procedure")
End Sub

Sub Triple(ByVal num As Double)
  'Triple a number
  lstResults.Items.Add(num & " from Sub procedure")
  num = 3 * num
  lstResults.Items.Add(num & " from Sub procedure")
End Sub
```

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

```
2 from event procedure
2 from Sub procedure
6 from Sub procedure
2 from event procedure
```

When a variable is passed by value, two memory locations are involved. Figure 5.10 shows the status of the memory locations as the program in Example 1 executes. At the time the Sub procedure is called, a temporary second memory location for the parameter is set aside for the Sub procedure's use and the value of the argument is copied into that location. After the completion of the Sub procedure, the temporary memory location is released, and the value in it is lost.