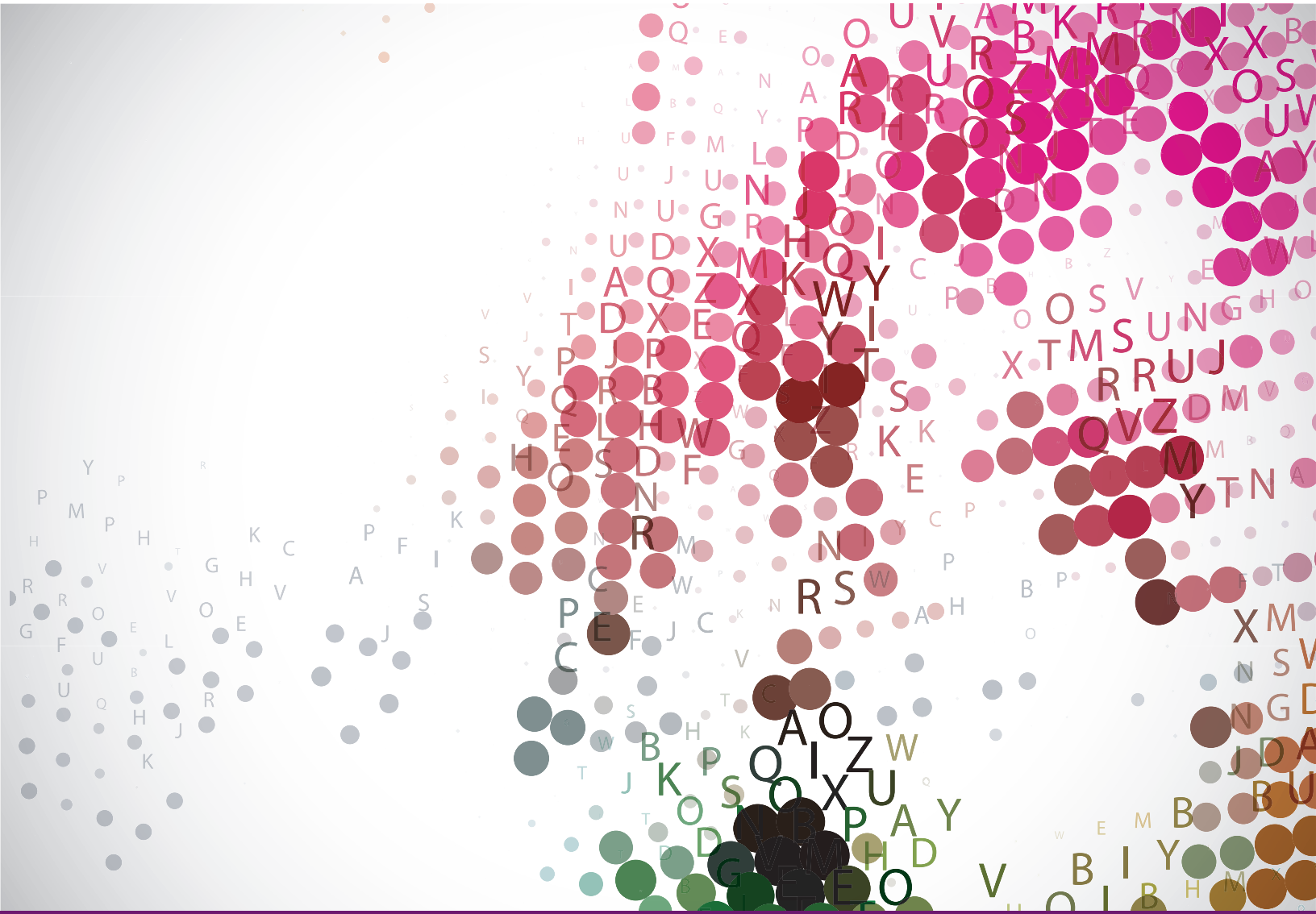
INTERNATIONAL EDITION

An Introduction To Programming
Using Visual Basic® 2012

NINTH EDITION

David I. Schneider



ALWAYS LEARNING

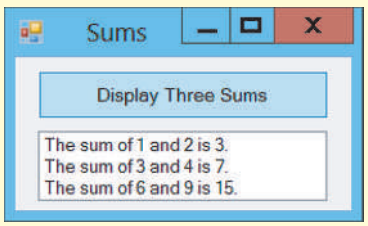
PEARSON

Example 1 **Add Numbers** 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.

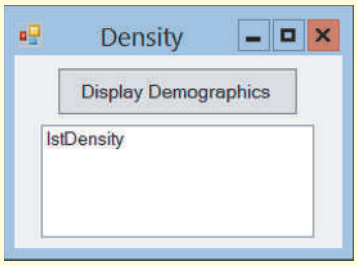
```
Private Sub btnAddNumbers_Click(...) Handles btnAddNumbers.Click
  DisplaySum(1, 2)
  Dim x As Double = 3
  Dim y As Double = 4
  DisplaySum(x, y)
  DisplaySum(2 * x, y + 5)
End Sub

Sub DisplaySum(num1 As Double, 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 **Population Density** 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 *landArea* are assigned the values 1375000 and 6423, 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.

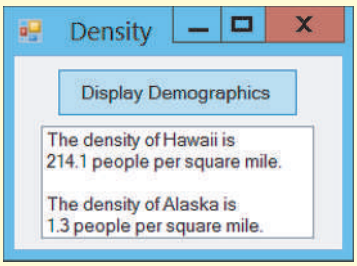


OBJECT	PROPERTY	SETTING
frmDensities	Text	Density
btnDisplay	Text	Display Demographics
lstDensity		

```
Private Sub btnDisplay_Click(...) Handles btnDisplay.Click
  'Calculate the population densities of states
  Dim state As String, pop, landArea As Double
  lstDensity.Items.Clear()
  state = "Hawaii"
  pop = 1375000
  landArea = 6423
  CalculateDensity(state, pop, landArea)
  lstDensity.Items.Add("")
  state = "Alaska"
  pop = 722718
  landArea = 570600
  CalculateDensity(state, pop, landArea)
End Sub

Sub CalculateDensity(state As String, pop As Double, landArea As Double)
  'The density (number of people per square mile)
  'will be displayed rounded to one decimal place.
  Dim density As Double
  density = pop / landArea
  lstDensity.Items.Add("The density of " & state & " is")
  lstDensity.Items.Add(density.ToString("N1") & " people per square mile.")
End Sub
```

[Run, and then click on the button.]



Notice that in the calling statement

```
CalculateDensity(state, pop, landArea)
```

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, landArea, 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 5.18 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 = 1375000
  a = 6423
  CalculateDensity(s, p, a)
  lstDensity.Items.Add("")
  s = "Alaska"
  p = 722718
  a = 570600
  CalculateDensity(s, p, a)
End Sub
```

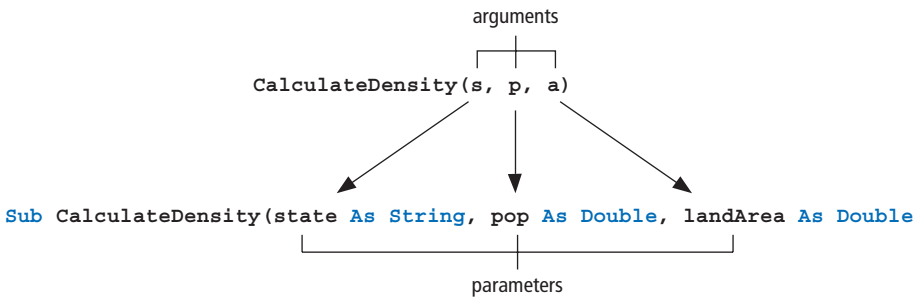


FIGURE 5.18 Passing arguments to a procedure.

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.

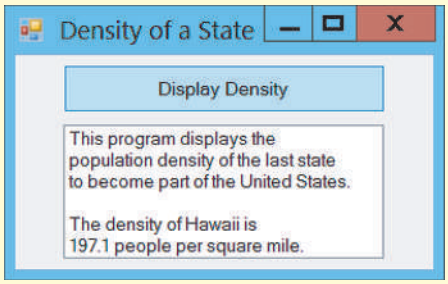
Example 3 Population Density The following variation of Example 2 gives the population density of a single state. The parameterless Sub procedure DescribeTask gives an explanation of the program.

```
Private Sub btnDisplay_Click(...) Handles btnDisplay.Click
  DescribeTask()
  CalculateDensity("Hawaii", 1375000, 6423)
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(state As String, pop As Double, landArea As Double)
  Dim density As Double
  density = pop / landArea
  lstOutput.Items.Add("") 'insert a blank line
  lstOutput.Items.Add("The density of " & state & " is")
  lstOutput.Items.Add(density.ToString("N1") & " 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 **Call Sub Procedures** 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 should be used. 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 after the Sub procedure has been created.) See Fig. 5.19.

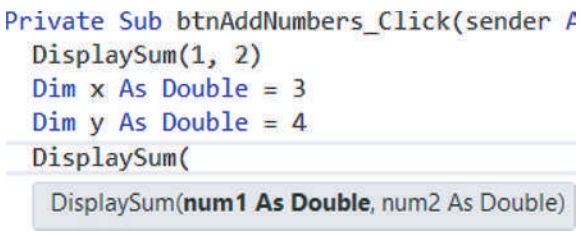


FIGURE 5.19 The Parameter Info help feature.

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(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(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(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(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(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(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
```