Invoking a Function Procedure

Example 1 – assigning the return value to a variable

```
dblNewPrice = GetNewPrice(dblPrice)
        or
dblPrice = GetNewPrice(dblPrice)
```

Example 2 – using the return value in a calculation

```
dblTotalDue = intQuantity * GetNewPrice(dblPrice)
```

The assignment statement multiplies the value in the `intQuantity` variable by the function's return value and then assigns the result to the `dblTotalDue` variable

Example 3 – displaying the return value

```
lblNewPrice.Text =
    GetNewPrice(dblPrice).ToString("C2")
```

Figure 17-4 Examples of invoking the GetNewPrice function

© Cengage Learning 2013

Price Calculator Application

Figure 17-5 shows the interface for the Price Calculator application. (The image in the picture box was downloaded from the Open Clip Art Library at *http://openclipart.org*.) The interface provides a text box for the user to enter the current price of an item. When the user clicks the Calculate button, the button's Click event procedure will use the GetNewPrice function to calculate and return the new price, which will be 5% more than the current price. The Click event procedure will then display the function's return value in the interface.

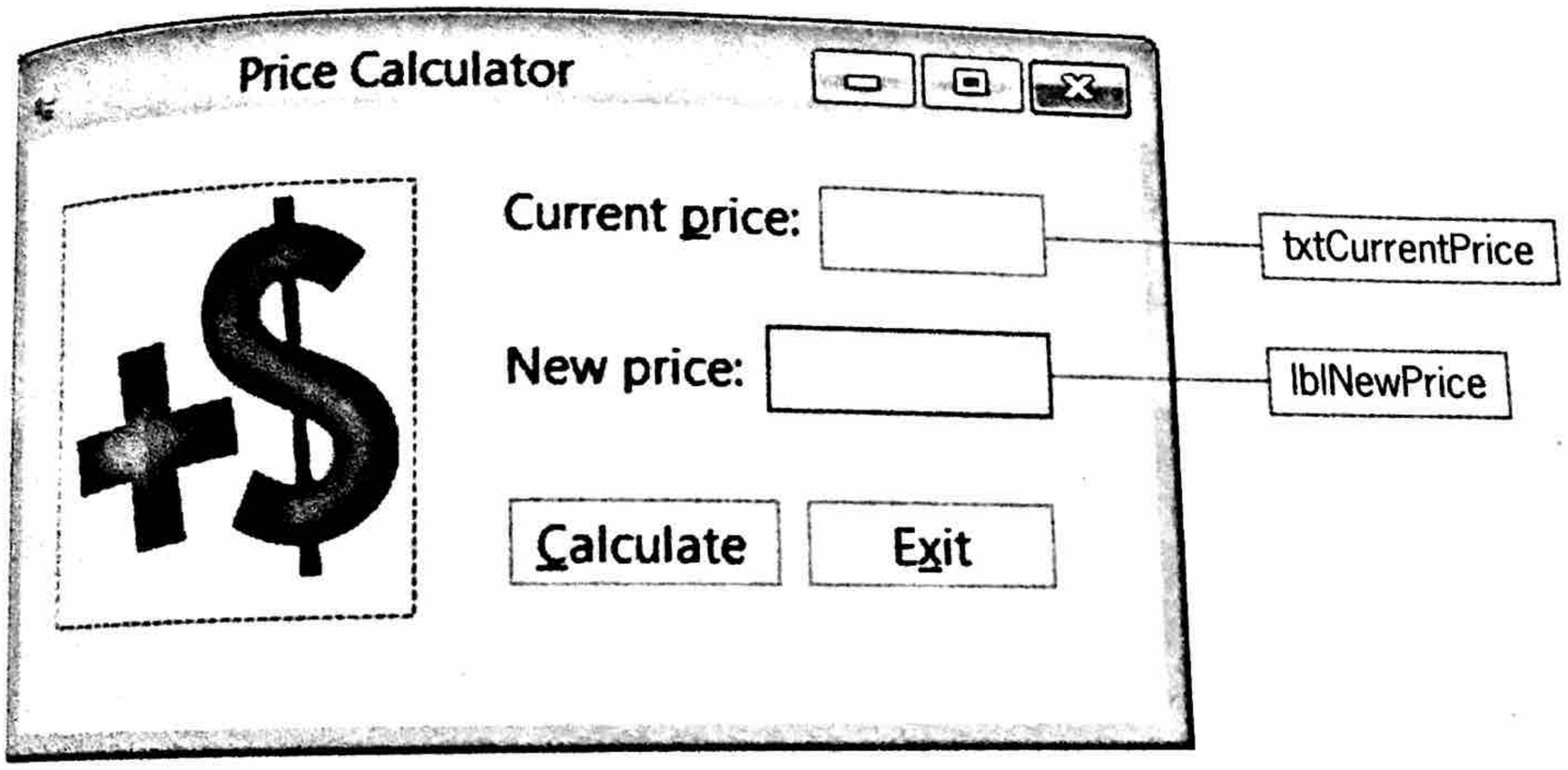

Figure 17-5 Interface for the Price Calculator application

OpenClipArt.org/baroquon/Jos Buivenga

To code the **Price Calculator application:**

1. Start Visual Studio 2012. Open the **Price Solution** (**Price Solution.sln**) file contained in the ClearlyVB2012\Chap17\Price Solution folder. If necessary, open the designer window.
2. Open the Code Editor window. First, you will enter the GetNewPrice function. Click the **blank line** below the `Public Class frmMain` clause and then press **Enter** to insert another blank line. Enter the GetNewPrice function shown in Figure 17-6.

enter this comment and these lines of code

```
Public Class frmMain

    Private Function GetNewPrice(ByVal dblOld As Double) As Double
        ' increases current price by 5% and returns new price

        Dim dblNew As Double
        dblNew = dblOld * 1.05
        Return dblNew
    End Function
```

Figure 17-6 GetNewPrice function

3. In the btnCalc_Click procedure, you will enter a statement that invokes the GetNewPrice function and assigns the function's return value to the `dblPrice` vari e. The statement will need to pass the function a copy of the current price stored in e `dblPrice` variable. Click the **blank line** below the `' get the new price` comment i e btnCalc_Click procedure and then enter the following assignment statement:

dblPrice = GetNewPrice(dblPrice)

Figure 17-7 shows the GetNewPrice function and the btnCalc_Click procedure.

GetNewPrice function

```
Private Function GetNewPrice(ByVal dblOld As Double) As Double
    ' increases current price by 5% and returns new price

    Dim dblNew As Double
    dblNew = dblOld * 1.05
    Return dblNew
End Function

Private Sub btnCalc_Click(sender As Object,
e As EventArgs) Handles btnCalc.Click
    ' calls a function to calculate the new price
    ' and then displays the new price

    Dim dblPrice As Double

    Double.TryParse(txtCurrentPrice.Text, dblPrice)

    ' get the new price
    dblPrice = GetNewPrice(dblPrice)

    lblNewPrice.Text = dblPrice.ToString("C2")
End Sub
```

invokes the GetNewPrice function and assigns the return value to the `dblPrice` variable

Figure 17-7 GetNewPrice function and btnCalc_Click procedure

© Cengage Learning 2013

Before testing the application, you will desk-check it using $8.99 as the current price. When t user clicks the Calculate button, the button's Click event procedure creates and initializes th `dblPrice` variable. The TryParse method converts the contents of the `txtCurrentPrice` control to the Double data type and stores the result in the `dblPrice` variable. Figure 17-8 shows the desk-check table before the GetNewPrice function is invoked.

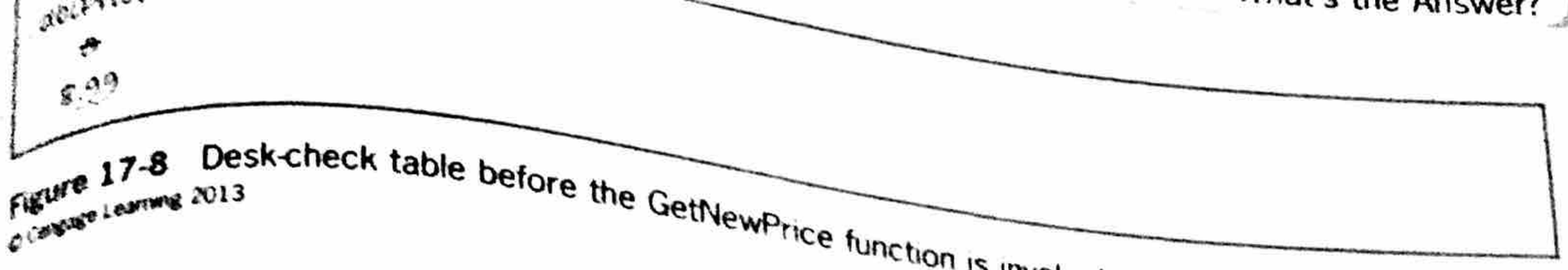

Figure 17-8 Desk-check table before the GetNewPrice function is invoked
© Cengage Learning 2013

Next, the `dblPrice = GetNewPrice(dblPrice)` statement invokes the GetNewPrice function, passing it the `dblPrice` variable *by value*. You can tell that the variable is passed *by value* because the keyword `ByVal` appears before its corresponding parameter in the function header. Recall that passing a variable *by value* means that only a copy of the variable's contents—in this case, the Double number 8.99—is passed to the function. At this point, the computer temporarily leaves the btnCalc_Click procedure to process the GetNewPrice function; the function header is processed first.

The parameterList in the GetNewPrice function header tells the computer to create a procedure-level Double variable named `dbl0ld`. The computer stores the information passed to the function—in this case, the Double number 8.99—in that variable. Next, the computer processes the code contained within the function. The Dim statement creates and initializes a procedure-level Double variable named `dblNew`. The assignment statement calculates the new price by multiplying the contents of the `dbl0ld` variable by 1.05. It stores the new price in the `dblNew` variable. Figure 17-9 shows the desk-check table after the assignment statement is processed.

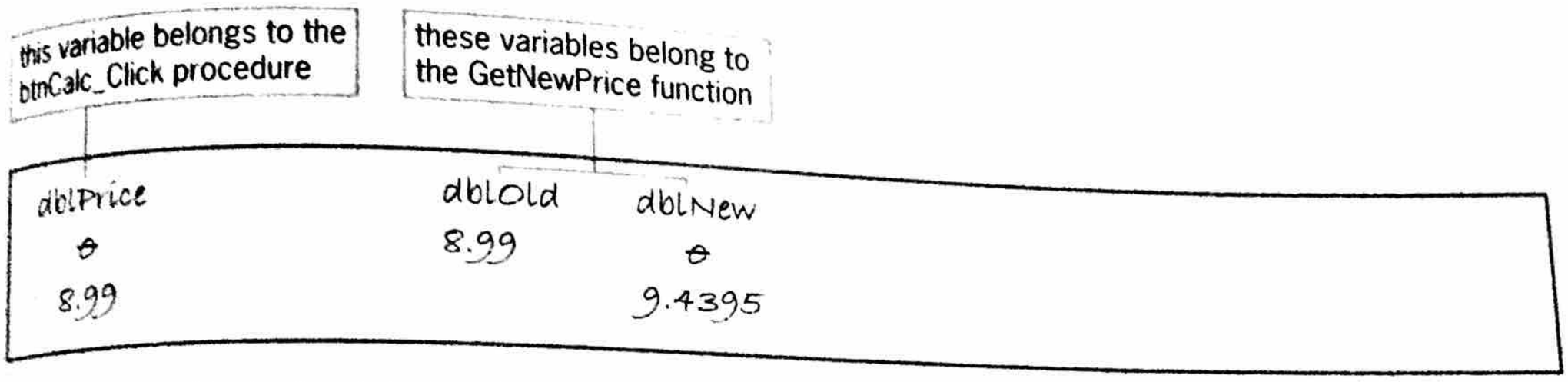

Figure 17-9 Desk-check table after the new price is assigned to the `dblNew` variable
© Cengage Learning 2013

The `Return dblNew` statement returns the contents of the `dblNew` variable to the statement that invoked the GetNewPrice function. That statement is the `dblPrice = GetNewPrice(dblPrice)` statement in the btnCalc_Click procedure. The statement assigns the function's return value to the `dblPrice` variable. The End Function clause is processed next and ends the GetNewPrice function. At this point, the computer removes the `dbl0ld` and `dblNew` variables from its internal memory. Figure 17-10 shows the desk-check table after the GetNewPrice function ends. Notice that the `dblPrice` variable now contains the new price.

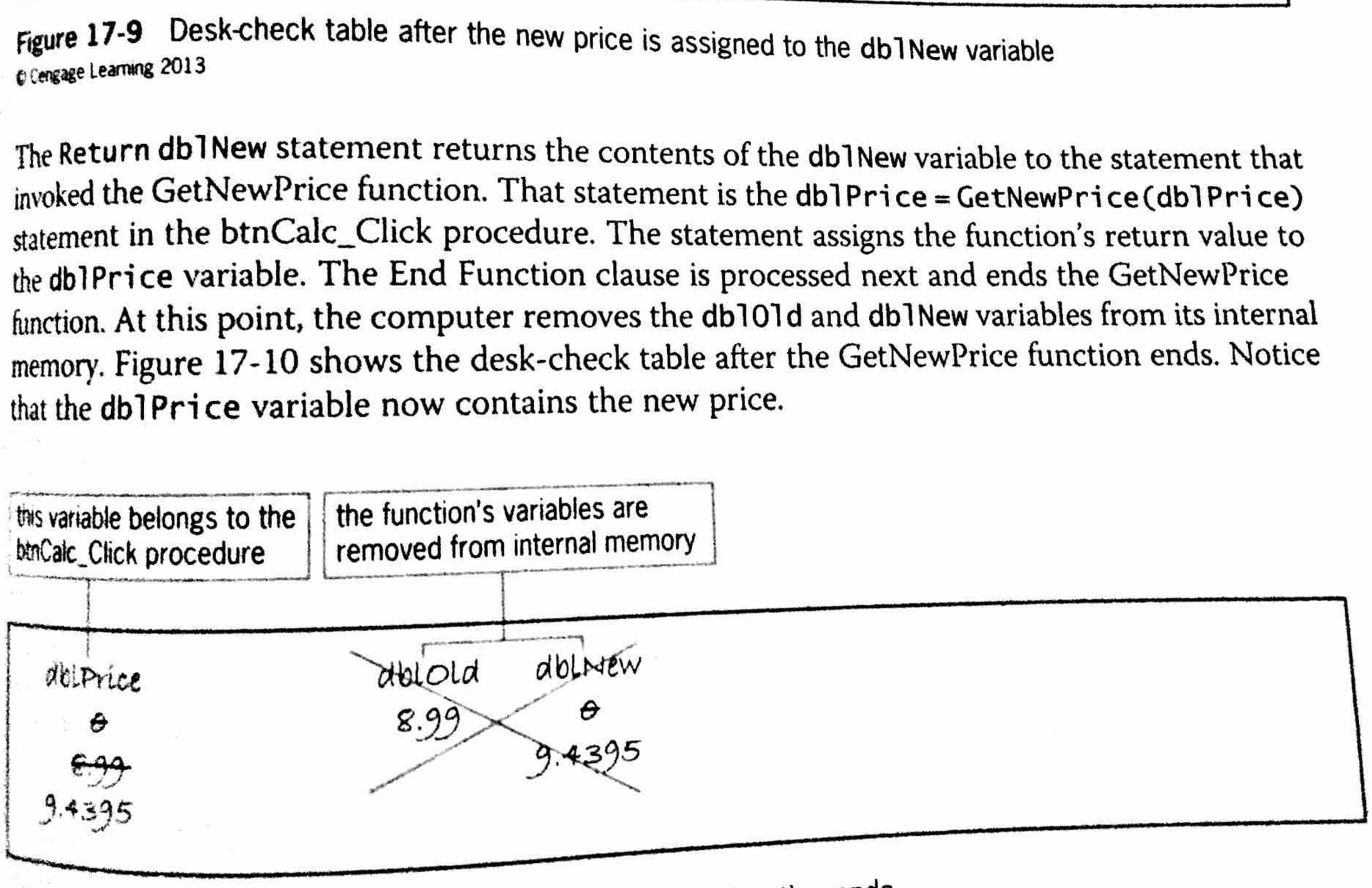

Figure 17-10 Desk-check table after the GetNewPrice function ends
© Cengage Learning 2013