8 Objectives

► Create a splash screen
► Pause the splash screen
► Add a ComboBox object to a Windows Form
► Write Code for a SelectedIndexChanged event
► Code a Sub procedure
Objectives

► Pass an argument to a procedure by value
► Pass an argument to a procedure by reference
► Code a Function procedure to return a value
► Create a class-level variable
► Catch an exception using a Try-Catch block
Introduction

► As an application grows, it is important to divide each facet of a problem into separate sections of code called **procedures**
► Try-Catch blocks can check for any error a user might commit
► One way to make your programs more professional is to display a splash screen while the full program loads
Chapter 8: Using Procedures and Exception Handling
Creating a Splash Screen

► Create a Windows application named Ocean Tours. Name the form frmTours. Click Project on the menu bar and then click Add New Item on the Project menu.

► In the Add New Item dialog box, select Splash Screen in the Templates section. Name the Splash Screen frmOceanToursSplashScreen in the Name text box.

► Click the Add button in the Add New Item dialog box.

► Click the splash screen form in the left side of the form to select the form. To set the application to display the splash screen first, right-click OceanTours in the Solution Explorer.

► Click Properties on the shortcut menu.
Creating a Splash Screen

► In the Windows application framework properties section, click the Splash screen list arrow, and then click frmOceanToursSplashScreen to select the form as the splash screen used for project.

► Click the Assembly Information Button on the Properties Designer to open the Assembly Information dialog box.

► To customize the splash screen, change the Title to Ocean Tours and the Copyright to the present year. The File Version can be changed as you update the application.
8 Creating a Splash Screen

- Click the OK button on the Assembly Information dialog box. Close the OceanTours* Project Designer window. To change the predefined image, first download the ocean.bmp picture from the scsite/vb2005/ch8/images Web site and store the image in a location you remember. Then, click the frmOceanToursSplashScreen.vb [Design] tab. Click the left side of the splash screen, making sure to select the entire splash screen form. The Properties window should identify MainLayoutPanel if you have selected the entire splash screen form. Click to the right of the Background Image property in the Properties window, and then click the ellipsis button. In the Select Resource dialog box, click the Project resource file radio button if necessary. Import the ocean.jpg picture by clicking the Import button in the Select Resource dialog box and selecting the ocean.bmp image from the location where you stored it. Click the OK button in the Select Resource dialog box.

- Run the application by clicking the Start Debugging button on the Standard toolbar.
Creating a Splash Screen

Ocean Tours
Version 1.00
Copyright © 2009
After the splash screen loads, the application executes any code in the form load event handler. To display the splash screen for five seconds, the code that calls the Sleep procedure should be in the form load event handler. To open the code editor window and the form load event handler, double-click the background of the frmTours Windows Form object in the Design window.

Click inside the frmToursLoad event handler. Type Threading. To cause IntelliSense to display a list of possible entries. If necessary, type T to select Thread from the IntelliSense list. Type .S to select Sleep from the IntelliSense list. Type (5000)
Chapter 8: Using Procedures and Exception Handling
Adding a ComboBox Object

- Drag the ComboBox .NET component from the Common Controls category of the Toolbox to the approximate location where you want to place the ComboBox object.

- With the ComboBox object selected, scroll in the Properties window to the (Name) property. Double-click in the right column of the (Name) property and then enter the name `cboIsland`.

- In the Properties window, scroll to the Text property. Click to the right of the Text property and enter `Select Island Location:` to specify the text that appears in the combo box. Resize the ComboBox object as needed to display the data in the box.
8 Adding a ComboBox Object

► In the Properties window, scroll to the Items property, and click to the right of the Items property on the word (Collection). Click the ellipsis button. The String Collection Editor dialog box opens. Enter the island locations Aruba (press ENTER), Jamaica (press ENTER), and Key West.

► In the String Collection Editor dialog box, click the OK button. Click the Start Debugging button on the Standard toolbar to run the application. Click the list arrow on the right of the ComboBox object to view the contents. You can select a choice from the list.
8 Adding a ComboBox Object
Determining the ComboBox Selected Index

```csharp
32 intIslandChoice = Me.cboIsland.SelectedIndex()
```
Determining the ComboBox Selected Index

► Select the ComboBox object named cboIsland on the Windows Form object
► Double-click the ComboBox object. Close the Toolbox
When a program is broken into manageable parts, each part is called a procedure

- A procedure is a named set of code that performs a given task

A Sub procedure is a procedure that completes its task but does not return any data to the calling procedure

A Sub procedure is the series of Visual Basic statements enclosed by the Sub and End Sub statements

A Sub procedure is called with a statement consisting of the procedure name and a set of parentheses in the form of a procedure call
Coding a Sub Procedure

General Format: Procedure Call

The procedure call is made:

```
ProcedureName()
```

The **procedure declaration** that begins the Sub procedure has the form:

```
Private Sub ProcedureName()

' Line(s) of code

End Sub
```
Coding a Sub Procedure

Private Sub cboIsland_SelectedIndexChanged(ByVal sender As System.Object, ByVal e As System.EventArgs) Handles cboIsland.SelectedIndexChanged
    ' This event handler allows the user to enter the island choice
    ' and then calls subprocedures to place the island activities
    ' in the list.

    Dim intIslandChoice As Integer

    intIslandChoice = Me.cboIsland_SelectedIndexChanged
    Me.lstTours.Items.Clear()
    Select Case intIslandChoice
        Case 0
            ArubaTours()
        Case 1
            JamaicaTours()
        Case 2
        Case Else
            KeyWestTours()
    End Select
When a procedure is called, however, the call statement can pass an argument to the called procedure.

```vbnet
Private Sub btnDaysOfWeek_Click(ByVal sender As System.Object, ByVal e As System.EventArgs) Handles btnDaysOfWeek.Click
    Dim intNumericDayOfWeek As Integer
    intNumericDayOfWeek = Convert.ToInt32(Me.txtDay.Text)
    Weekday(intNumericDayOfWeek)
    MessageBox.Show("Have a Great Week!", "Goodbye")
End Sub

Private Sub Weekday(ByVal intDay As Integer)
    If intDay = 1 Or intDay = 7 Then
        Me.lblDisplayDay.Text = "Weekend"
    End If
    If intDay >= 2 And intDay <= 6 Then
        Me.lblDisplayDay.Text = "Weekday"
    End If
End Sub
```
When a procedure is called, however, the call statement can pass an argument to the called procedure.

The value is copied into a variable whose name is specified in the Sub procedure declaration statement.
Passing Arguments by Value (ByVal)

```vbnet
Private Sub btnShowMessages_Click(ByVal sender As System.Object, ByVal e As System.EventArgs) Handles btnShowMessages.Click

    Dim strMessage As String

    strMessage = "The Original Welcome Message"
    MessageBox.Show(strMessage, "First Message")
    DisplayMessage(strMessage)
    MessageBox.Show(strMessage, "Fourth Message")

End Sub

Private Sub DisplayMessage(ByVal strShowMessage As String)

    MessageBox.Show(strShowMessage, "Second Message")
    strShowMessage = "The Changed Welcome Message"
    MessageBox.Show(strShowMessage, "Third Message")

End Sub
```
Passing Arguments by Value (ByVal)
The second way in which to pass an argument from a calling procedure to a called Sub procedure is by reference.

You specify you want to pass a value by reference by entering the keyword ByVal in the Sub procedure declaration.

Passing a value by reference allows code in the Sub procedure to modify the contents of the variable that is being passed because when you use ByRef, you are passing a reference to the variable that holds the value instead of the value as when you use ByVal.
Passing Arguments by Reference

Private Sub btnDisplayMessage_Click(ByVal sender As System.Object, ByVal e As System.EventArgs) Handles btnDisplayMessage.Click
    Dim strFavoriteArtist As String
    strFavoriteArtist = "Vincent Van Gogh"
    MessageBox.Show("Favorite Artist is " & _
                    strFavoriteArtist, "First Message")
    DisplayMessage(strFavoriteArtist)
    MessageBox.Show("Favorite Artist is now " & _
                    strFavoriteArtist, "Fourth Message")
End Sub

Private Sub DisplayMessage(ByVal strShowArtist As String)
    MessageBox.Show("Favorite Artist is " & _
                    strShowArtist, "Second Message")
    ' The artist name is changed
    strShowArtist = "Paul Cezanne"
    MessageBox.Show("Favorite Artist is " & _
                    strShowArtist, "Third Message")
End Sub
Chapter 8: Using Procedures and Exception Handling

8 Passing Arguments by Reference

---

First Message
Favorite Artist is Vincent Van Gogh
OK

Second Message
Favorite Artist is Vincent Van Gogh
OK

Third Message
Favorite Artist is Paul Cezanne
OK

Fourth Message
Favorite Artist is now Paul Cezanne
OK
Passing Multiple Arguments

- You can pass as many arguments as needed to a Sub procedure.
- If you have more than one argument, the variables are passed in the same order in which they appear in the procedure call statement.
A **Function procedure** is similar to a Sub procedure except that a Function procedure returns a single value to the calling procedure.

### General Format: Function Procedure Call

The Function procedure call is made:

```plaintext
VariableName = FunctionProcedureName()
```

The **procedure declaration** that begins the Function procedure has the form:

```plaintext
Private Function FunctionProcedureName() as DataType
    ' Line(s) of code
    Return VariableName
End Function
```
The Function procedure is different in appearance from a Sub procedure in the following ways:

- The Function procedure call has a receiving variable that is assigned the returned value from the Function procedure.
- The data type of the return value is listed in the procedure declaration.
- The keyword Return is used in the Function procedure to return a single value.
Function Procedures

Private Sub btnCompute_Click(ByVal sender As System.Object, ByVal e As System.EventArgs) Handles btnCompute.Click
    Dim decMiles As Decimal
    Dim decGallons As Decimal
    Dim decMilesPerGallon As Decimal

    decMiles = Convert.ToDecimal(Me.txtMiles.Text)
    decGallons = Convert.ToDecimal(Me.txtGallons.Text)
    decMilesPerGallon = ComputeGasMileage(decMiles, decGallons)
    MessageBox.Show("You are getting " & decMilesPerGallon.ToString("P1") & " miles per gallon", "MPG")
End Sub

Function ComputeGasMileage(ByVal decMiles As Decimal, ByVal decGallons As Decimal) As Decimal
    Dim decMileage As Decimal

    decMileage = decMiles / decGallons
    ' The following statement returns control to the calling procedure and returns the value in the decMileage variable.
    Return decMileage
End Function
Creating a Private Class-Level Variable

► When a class-level variable cannot be referenced outside the class in which it is declared, the variable is said to have Private access.

```csharp
' Class variables
Private _intTwoHours As Integer = 2
Private _intThreeHours As Integer = 3
Private _intFourHours As Integer = 4
Private _intEightHours As Integer = 8
Private _decTwentyPercentDiscount As Decimal = 0.2D
Private _decTwentyFivePercentDiscount As Decimal = 0.25D
Private _strDeepSeaFishing As String = "Deep Sea Fishing"
Private _strKayaking As String = "Kayaking"
Private _strScuba As String = "Scuba"
Private _strSnorkeling As String = "Snorkeling"
Private _strGlassBottomBoat As String = "Glass Bottom Boat"
Private _strParasailing As String = "Parasailing"
```
The **Try-Catch** set of statements detects exceptions and takes corrective action.

**General Format: Try-Catch block**

- **Try**
  - ‘Try Block of Code – Executable statement(s) that may generate an exception.
- **Catch** (filter for possible exceptions)
  - ‘Catch Block of Code for handling the exception
- [Optional: Additional Catch blocks]
- [Optional Finally]
  - ‘Optional statements that will always execute before finishing the Try block
- **End Try**
8 Exception Handling

Dim decNumerator As Decimal
Dim decDenominator As Decimal
Dim decDivision As Decimal

decNumerator = Convert.ToDecimal(Me.txtNum.Text)
decDenominator = Convert.ToDecimal(Me.txtDen.Text)

Try
    decDivision = decNumerator / decDenominator
Catch Exception As DivideByZeroException
    MessageBox.Show("Attempt to divide by zero")
End Try
## Exception Handling

<table>
<thead>
<tr>
<th>Exception Type</th>
<th>Condition when Exception Occurs</th>
<th>Code Example</th>
</tr>
</thead>
</table>
| ArgumentNullException          | A variable that has no value is passed to a procedure                                            | `Dim strTerm As String`  
`Me.1stDisplay.Items.Add(strTerm)` |
| DivideByZeroException        | A value is divided by zero                                                                        | `intValue = intResult = intNum / 0`                                        |
| FormatException              | A variable is converted to another type that is not possible                                      | `strTerm = "Code"`  
`intValue = Convert.ToInt32(strTerm)`                                      |
| NullReferenceException       | A procedure is called when the result is not possible                                             | `Dim strTerm as String`  
`intValue = strTerm.Length`                                                    |
| OverflowException            | A value exceeds its assigned data type                                                             | `Dim intCost as Integer`  
`intCost = 58 ^ 4000000000`                                                    |
| SystemException              | Generic                                                                                            | Catches all other exceptions                                                |
Private Function ValidateNumberInParty() As Boolean
    ' This procedure validates the value entered for the number in party

    Dim intPartyNumber As Integer
    Dim blnValidityCheck As Boolean = False
    Dim strNumberInPartyErrorMessage As String = _
    "Please enter the number of people in your party (1-99)"
    Dim strMessageBoxTitle As String = "Error"

    Try
        intPartyNumber = CInt(Me.txtNumberInParty.Text)
        If intPartyNumber > 0 And intPartyNumber < 100 Then
            blnValidityCheck = True
        Else
            MessageBox.Show(strNumberInPartyErrorMessage, _
                             strMessageBoxTitle)
            Me.txtNumberInParty.Focus()
            Me.txtNumberInParty.Clear()
        End If
    Catch Exception As FormatException
        MessageBox.Show(strNumberInPartyErrorMessage, _
                         strMessageBoxTitle)
        Me.txtNumberInParty.Focus()
        Me.txtNumberInParty.Clear()
    Catch Exception As OverflowException
        MessageBox.Show(strNumberInPartyErrorMessage, _
                         strMessageBoxTitle)
        Me.txtNumberInParty.Focus()
        Me.txtNumberInParty.Clear()
    Catch Exception As SystemException
        MessageBox.Show(strNumberInPartyErrorMessage, _
                         strMessageBoxTitle)
        Me.txtNumberInParty.Focus()
        Me.txtNumberInParty.Clear()
    End Try

    Return blnValidityCheck
End Function
**Chapter 8: Using Procedures and Exception Handling**

---

**REQUIREMENTS DOCUMENT**

<table>
<thead>
<tr>
<th>Date submitted:</th>
<th>May 19, 2009</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Application title:</strong></td>
<td>Ocean Tours Trip Selection</td>
</tr>
<tr>
<td><strong>Purpose:</strong></td>
<td>This Windows application allows a customer to view ocean tours available in the Caribbean islands.</td>
</tr>
<tr>
<td><strong>Program Procedures:</strong></td>
<td>From a Windows application, the user can select ocean tours and find out pricing information.</td>
</tr>
</tbody>
</table>
| **Algorithms, Processing, and Conditions:** | 1. The user first selects the island location. No other objects are displayed at this time.  
2. When the user selects an island, the following items are displayed in the window: Number in party text box, a custom list of the available tours on the chosen island, and a combo box that lists the months of the year. In addition, a button to find the cost of the tour and a button to clear the form are included.  
3. The user enters the number of guests in their party.  
4. From the custom list of tours that are available for the chosen island, the user selects the tour desired.  
5. The user selects the month of the tour to determine if any discounts are available during that month.  
6. The total price for the ocean tour for the group is displayed. |
| **Notes and Restrictions:** | 1. Validate numeric input with Try-Catch blocks.  
2. Use multiple procedures to break the application into manageable sections. |
| **Comments:** | 1. The ocean.bmp picture used in the Window is available at scsite/vb2005/ch8/images.  
2. A splash screen is shown for approximately 5 seconds before the main window is displayed. |
USE CASE DEFINITION

1. A splash screen welcomes the user for approximately 5 seconds.
2. The user selects an island location.
3. The program displays a text box for the number of people in the party, a list of available ocean tours for the selected island, and a list of months to specify when travel will occur.
4. The user enters the number of people in the party, selects a tour, selects the month, and clicks the Find Cost of Tour button.
5. The program identifies the tour, calculates and displays the tour cost for the entire party, specifies the length in hours of the tour, and identifies any discount percentage that was applied to the price.
6. The user can change any of the entries (island choice, number in party, tour, and date) and click the Find Cost of Tour button to recalculate the tour cost.
7. The user can clear the form by clicking the Clear Form button.
# Program Design

<table>
<thead>
<tr>
<th>Location</th>
<th>Tour Type</th>
<th>Tour Length</th>
<th>Tour Cost</th>
<th>Discount</th>
</tr>
</thead>
<tbody>
<tr>
<td>Aruba</td>
<td>Deep Sea Fishing</td>
<td>8 hours</td>
<td>$199</td>
<td>20% Discount: April/May/June</td>
</tr>
<tr>
<td></td>
<td>Kayaking</td>
<td>2 hours</td>
<td>$89</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Scuba</td>
<td>3 hours</td>
<td>$119</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Snorkeling</td>
<td>4 hours</td>
<td>$89</td>
<td>25% Discount: Feb/March</td>
</tr>
<tr>
<td>Jamaica</td>
<td>Glass Bottom Boat</td>
<td>2 hours</td>
<td>$39</td>
<td>20% Discount: Sept/Oct</td>
</tr>
<tr>
<td></td>
<td>Parasailing</td>
<td>2 hours</td>
<td>$119</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Snorkeling</td>
<td>3 hours</td>
<td>$59</td>
<td></td>
</tr>
<tr>
<td>Key West</td>
<td>Deep Sea Fishing</td>
<td>4 hours</td>
<td>$89</td>
<td>20% Discount: April/May/Sept</td>
</tr>
<tr>
<td></td>
<td>Glass Bottom Boat</td>
<td>2 hours</td>
<td>$29</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Scuba</td>
<td>3 hours</td>
<td>$119</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Snorkeling</td>
<td>3 hours</td>
<td>$59</td>
<td>25% Discount: Feb/March</td>
</tr>
</tbody>
</table>
Program Design when using Sub and Function Procedures

► When a program becomes larger, often it is advantageous to break the program into procedures, which perform specific tasks within the program
  • Makes the program easier to read, understand, and debug

► The developer must determine what code should be placed in a procedure. Procedures should:
  • Perform a single task
  • Perform reasonably substantial processing

► Use Sub and Function procedures appropriately
# Event Planning Document

<table>
<thead>
<tr>
<th>Program Name:</th>
<th>Developer:</th>
<th>Object:</th>
<th>Date:</th>
</tr>
</thead>
<tbody>
<tr>
<td>Ocean Tours Trip Selection</td>
<td>Corinne Hoisington</td>
<td>frmSplashScreen</td>
<td>June 4, 2009</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Object</th>
<th>Event Trigger</th>
<th>Event Processing</th>
</tr>
</thead>
<tbody>
<tr>
<td>frmSplashScreen_Load</td>
<td>Load</td>
<td>An opening splash screen opens with the company name, version number, and year</td>
</tr>
</tbody>
</table>
## Event Planning Document

**Program Name:** Ocean Tours Trip Selection  
**Developer:** Corinne Hoisington  
**Object:** frmTours  
**Date:** June 4, 2009

<table>
<thead>
<tr>
<th>EVENT TRIGGER</th>
<th>EVENT PROCESSING</th>
</tr>
</thead>
<tbody>
<tr>
<td>cboIsland_SelectedIndexChanged</td>
<td>Assign island choice selection to an Integer SUB (Aruba(), Jamaica(), KeyWest()): Based on the island selection, display a list of the available tours on the island. Change Visible property for all objects on the form to True. Clear the labels that provide trip information. Set the focus on the number in party text box.</td>
</tr>
<tr>
<td>ArubaTours()</td>
<td>Display list of available tours (deep sea fishing, kayaking, scuba, and snorkeling) in list box</td>
</tr>
<tr>
<td>JamaicaTours()</td>
<td>Display list of available tours (glass bottom boat, parasailing, snorkeling) in list box</td>
</tr>
<tr>
<td>KeyWestTours()</td>
<td>Display list of available tours (deep sea fishing, glass bottom boat, scuba, snorkeling) in list box</td>
</tr>
</tbody>
</table>
# Event Planning Document

<table>
<thead>
<tr>
<th>Program Name:</th>
<th>Developer:</th>
<th>Object:</th>
<th>Date:</th>
</tr>
</thead>
<tbody>
<tr>
<td>Ocean Tours Trip Selection</td>
<td>Corinne Hoskington</td>
<td>Free Tours</td>
<td>June 4, 2009</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>OBJECT</th>
<th>EVENT TRIGGER</th>
<th>EVENT PROCESSING</th>
</tr>
</thead>
<tbody>
<tr>
<td>Item FC</td>
<td>Click</td>
<td>SUB (ValidateNumbersInParty): Validate the value in the number in party text box is valid. FUNCTION (ValidateTourSelection): Ensure the user has selected a tour in the list box. If the number in party is valid and a tour is selected: Convert number in party to an integer. Set month to selected month + 1. If the month is greater than zero: Change selected island index to integer. FUNCTION (ArubaFindCosts, JamaicaFindCosts, KeyWestFindCosts): Calculate cost based on island choice. Display tour cost, length and discount. Else display error message.</td>
</tr>
<tr>
<td>ValidateNumbersInParty()</td>
<td>Function procedure call</td>
<td>Set Boolean indicator to False. Convert number in party to integer. If conversion valid: If number &gt;=0 and &lt;100: Set Boolean indicator to True for valid number. If conversion not valid: Catch format, overflow, and system exceptions. Display error message box. Place focus in number in party text box. Clear number in party text box. Return Boolean indicator.</td>
</tr>
<tr>
<td>ValidateTourSelection()</td>
<td>Function procedure call</td>
<td>Convert tour selection index to integer. If conversion successful: Place selected item string in ByRef variable. Set Boolean validity indicator to True. Else: Display error message box. Set Boolean validity indicator to False. Return ocean tour selected index integer.</td>
</tr>
<tr>
<td>ArubaFindCost()</td>
<td>Function procedure call</td>
<td>If tour selected is deep sea fishing: Set cost to Aruba deep sea cost for one person. Set length to Aruba deep sea length. If April, May, June, Discount = 20%. If tour selected is kayaking: Set cost to Aruba kayak for one person. Set length to Aruba kayak. If tour selected is scuba: Set cost to Aruba scuba for one person. Set length to Aruba scuba. If tour selected is snorkel: Set cost to Aruba snorkel for one person. Set length to Aruba snorkel. If Feb or March, Discount = 25%. Calculate cost of trip (Cost = (cost * discount %) * number in party). Return cost of trip.</td>
</tr>
</tbody>
</table>
## Event Planning Document

<table>
<thead>
<tr>
<th>Program Name:</th>
<th>Developer:</th>
<th>Object:</th>
<th>Date:</th>
</tr>
</thead>
<tbody>
<tr>
<td>Ocean Tours Trip Selection</td>
<td>Corinne Hoisington</td>
<td>frmTours</td>
<td>June 4, 2009</td>
</tr>
</tbody>
</table>

### OBJECT  | EVENT TRIGGER  | EVENT PROCESSING
---|---|---
JamaicaFindCost() | Function procedure call | If tour selected is glass bottom boat  
Set cost to Jamaica glass bottom boat cost for one person  
Set length to Jamaica glass bottom length  
If Sept, Oct, Discount = 20%  
If tour selected is parasail  
Set cost to Jamaica parasail for one person  
Set length to Jamaica parasail  
If tour selected is snorkel  
Set cost to Jamaica snorkel for one person  
Set length to Jamaica snorkel  
Calculate cost of trip ((cost = (cost * discount %)) * number in party)  
Return cost of trip

KeyWestFindCost() | Function procedure call | If tour selected is deep sea fishing  
Set cost to Key West deep sea cost for one person  
Set length to Key West deep sea length  
If tour selected is glass bottom boat  
Set cost to Key West glass bottom boat for one person  
Set length to Key West glass bottom  
If Apr, May, Sept, Discount = 20%  
If tour selected is scuba  
Set cost to Key West scuba for one person  
Set length to Key West scuba  
If tour selected is snorkel  
Set cost to Key West snorkel for one person  
Set length to Key West snorkel  
If Feb or March, Discount = 25%  
Calculate cost of trip ((cost = (cost * discount %)) * number in party)  
Return cost of trip

btnClear | Click | Set Select Item cbo text to “Select Island Location”  
Clear text boxes, list, and labels  
Hide all objects except Select Island cbo  
Set month selected index to –1 (no month selected)  
Set month cbo Text to “Which month?”

frmTours_Load | Load | Set sleeping period to 5000 milliseconds
Summary

► Create a splash screen
► Pause the splash screen
► Add a ComboBox object to a Windows Form
► Write Code for a SelectedIndexChanged event
► Code a Sub procedure
Summary

► Pass an argument to a procedure by value
► Pass an argument to a procedure by reference
► Code a Function procedure to return a value
► Create a class-level variable
► Catch an exception using a Try-Catch block
CHAPTER 8 COMPLETE

Using Procedures and Exception Handling