Contrôle du flux de code

Les méthodes que vous écrivez s'exécutent une ligne à la fois de haut en bas, de gauche à droite. Il peut arriver que vous souhaitiez que votre application exécute une partie de son code en fonction de certaines conditions (à l'aide de comparaisons). Lorsque la logique de votre application doit prendre des décisions, on parle de création de branches. Cela vous permet de contrôler quel code est exécuté et quand. Il y a deux instructions à embranchement

If... Then... EndIf

Le iF... Then... End If est utilisée lorsque votre code doit tester une expression booléenne (True ou False), puis exécuter du code en fonction de son résultat. Si l'expression que vous testez est True, alors les lignes de code que vous placez entre les champs If... Then et la ligne End If sont exécutées, sinon elles sont sautées.

```
If condition Then
  ' [Votre Code là]
End If
```

Pour exécuter votre code lorsque la valeur Integer month est égale à 1 :

```
If mois = 1 Then
  ' [Votre Code là]
End If
```

La partie « mois = 1 » est une expression booléenne ; c'est soit True, soit False. Le mois variable est soit 1, soit pas 1. Supposons que vous disposiez d'un bouton qui effectue une tâche supplémentaire si une case à cocher particulière est cochée. La propriété value d'une CheckBox est booléenne, vous pouvez donc la tester dans une instruction If comme celle-ci :

```
If CheckBox1.Value Then
  ' [Votre Code là]
End If
```

N'oubliez pas que vous pouvez déclarer des variables locales à l'aide de l'instruction Var à l'intérieur d'une instruction If. Toutefois, ces variables sortent de la portée après l'instruction End If. Par exemple:

```
If error = -123 Then
  Var a As String
  a = "Bonjour !"
End If

Label1.Text = a ' Variable Hors de portée!
```

Si vous avez besoin de la variable après l'instruction End If, vous devez la déclarer localement pour l'ensemble de la méthode, et non dans l'instruction If... End If.

If... Then... Else... EndIf

Un If... Then... L'instruction End If peut être développée en incluant une clause Else supplémentaire. Dans certains cas, vous devez exécuter du code si une expression booléenne est True, mais un ensemble de code différent si l'expression booléenne est False. Dans ces situations, vous ajoutez la clause Else pour le code qui s'exécutera lorsque l'expression booléenne sera évaluée à False.

Ce code définit le texte en fonction d'une variable mensuelle :

```
Var mois As Integer = 6
Var resultat As String

If mois = 1 Then
  resultat = "Janvier"
Else ‘ mois <> 1
  resultat = "Pas janvier"
End If

' resultat = "Pas janvier"
```

iF... Then... ElseIf... EndIf

La prochaine étape dans l'amélioration de votre... Then... End If doit comporter plusieurs tests lorsque l'expression booléenne initiale est False. Pour chaque test supplémentaire, vous utilisez l'instruction ElseIf.

```
Var mois As Integer
Var resultat As String

If mois = 1 Then
  resultat = "janvier."
ElseIf mois < 4 Then
  resultat = "c'est l'hiver !"
Else
  resultat = "C'est pas l'hiver !
End If
```

Vous pouvez, bien sûr, utiliser un If... Alors... End If dans la partie Else de la première instruction If pour effectuer un autre test. Cependant, cela ajoute un autre End If et complique inutilement votre code. Au lieu de cela, vous pouvez utiliser autant d'instructions ElseIf que nécessaire, suivies d'une instruction Else finale facultative :

```
Var resultat As String
Var mois As Integer

If mois = 1 Then
  resultat = "janvier."
ElseIf mois < 4 Then
  resultat = "Hiver."
ElseIf mois < 6 Then
  resultat = "peut etre le printemps
Else
  resultat = "Eté ou automne".
End If
```

Si la condition initiale est False, votre code continue de tester les conditions ElseIf jusqu'à ce qu'il en trouve une qui est True. Il exécute ensuite le code associé à cette instruction ElseIf et continue d'exécuter les lignes de code qui suivent l'instruction End If.

If… Then… Else

En réduisant un peu, une simple instruction If peut être écrite sur une seule ligne, à condition que le code qui suit les instructions Then et Else (facultative) puisse être écrit sur une seule ligne. Lorsque vous utilisez cette syntaxe, vous omettez l'instruction End If. Quelques exemples :

```
Var s As String

If error = 123 Then s = "An error occurred."
If error = 123 Then s = "An error occurred." Else s = "Success"
If error = 103 Then Break
```

Avertissement

Les instructions If sur une seule ligne ne sont pas recommandées dans la plupart des cas, car il sera difficile lors du débogage de savoir si la condition s'est produite ou non. L'exception est lorsqu'elle est utilisée avec la méthode Break (comme dans le dernier exemple ci-dessus) pour afficher le débogueur lorsqu'une condition particulière se produit.

Si l'opérateur

Dans les situations où vous devez simplement renvoyer un résultat basé sur une expression booléenne, vous pouvez utiliser l'opérateur If.

```
If(condition, resultatIfTrue, resultatIfFalse)
```

La condition est évaluée et si elle est True, le *resultatIfTrue* est renvoyé, sinon *resultatIfFalse* est renvoyé.

Par exemple, ce code affiche « Grand » :

```
Var resultat As String
Var monentier As Integer = 41

resultat = If(monentier > 40, "Grand", "Petit")
```

Étant donné que cela renvoie un résultat, les types *resultatIfTrue* et *resultatIfFalse* doivent correspondre (ou pouvoir être convertis entre eux) et doivent correspondre au type de destination.

Select... Case

Lorsque vous devez tester une propriété ou une variable pour l'une des nombreuses valeurs possibles, puis prendre une action en fonction de cette valeur, utilisez une instruction Select Case.

Prenons l'exemple suivant qui utilise If... ElseIf.. End If pour tester une variable (dayNumber) et afficher le jour de la semaine :

```
Var jour As String
Var jourNumerique As Integer

Select Case jourNumerique
Case 2
  dayName = "Lundi"
Case 3
  dayName = "Mardi"
Case 4
  dayName = "Mercredi"
Case 5
  dayName = "Jeudi"
Case 6
  dayName = "Vendredi"
Else
  MessageBox(" Le Weekend.")
End Select

Var resultat As String = "On est : " + jour
```

Le Select... L'instruction Case compare la variable ou la propriété passée dans la première ligne à chaque valeur des instructions Case. Une fois qu'une correspondance est trouvée, le code entre ce cas et le suivant est exécuté.

Vous pouvez utiliser n'importe quel type de variable dans la fenêtre Sélect... Case. Cet exemple compare des valeurs de texte :

```
Var dayName As String
Var dayNumber As Integer

Select Case dayName
Case "Lundi"
  dayNumber = 2
Case "Mardi"
  dayNumber = 3
End Select
```

'Autre notations

```
Select Case dayNumber
Case 2
  Var day As String
  day = "Tuesday"
Else
  Var day As String ' new scope
  day = "It is NOT Tuesday!")
End Select

' day is now out of scope
```

```
Var c As Color
c = &cFF0000 ' pure red

Select Case c
Case &c00FF00 ' green
  MessageBox("Green")
Case &cFF0000 ' red
  MessageBox("Red")
Case &c0000FF ' blue
  MessageBox("Blue")
End Select
```

```
Var c As Color
c = &cFF0000 ' pure red

Select Case c
Case &c00FF00, &cFF0000 ' green, red
  MessageBox("Green or red")
Case &cFF0000 ' red
  MessageBox("Red")
Case &c0000FF ' blue
  MessageBox("Blue")
End Select
```

```
Var c As Color
c = &cFF0000 ' pure red

Select Case c
Case &c00FF00 ' green
  MessageBox("Green")
Case &cFF0000 ' red
  MessageBox("Red")
Case &c0000FF ' blue
  MessageBox("Blue")
Case Else
  MessageBox("None of the above")
End Select
```

```
Var i As Integer = 53

Select Case i
Case 1 To 25
  MessageBox("25 or less")
Case 26 To 50
  MessageBox("26 to 50")
Case 51 To 100
  MessageBox("51 to 100")
End Select
```

In this example, the third case, “51 to 100”, is True.

```
Case 0, 26 To 50, 75, 100 To 200

Var i As Integer = 10

Select Case i
Case Is <= 10
  ' this case selected
Case Is > 10
  ' this case not selected
End Select
```

```
Var i As Integer = 75

Select Case i
Case 0, Is <= 10, 100
  ' case not selected
Case Is > 10, Is < 99
  ' case selected
End Select


Var i As Integer = 4
Var a As Integer = 2

Select Case i
Case CalcSquare(a)
  ' case 1
Case a
  ' case 2
Else
  ' no match
End Select

Function CalcSquare(a As Integer) As Integer
  Return a * a
End Function
```

Ou

```
Case a * a
```

Autre

```
Var d As New MessageDialog
Var b As MessageDialogButton

d.Icon = MessageDialog.GraphicCaution
d.ActionButton.Caption = "Save"
d.CancelButton.Visible = True
d.AlternateActionButton.Visible = True
d.AlternateActionButton.Caption = "Don't Save"
d.Message = "Save changes before closing?"
d.Explanation = "If you don't save your changes, you will lose your work."

b = d.ShowModal

Select Case b
Case d.ActionButton
  ' user pressed Save
Case d.AlternateActionButton
  ' user pressed Don't Save
Case d.CancelButton
  ' user pressed Cancel
End Select
```

Autre

```
Case IsA ClassName

Select Case Me
Case IsA DesktopButton
  MessageBox("I 'm a Button.")
Case IsA DesktopTextField
  MessageBox("Nope!")
End Select
```

The term “Me” refers to the Button, so the first Case statement returns True.

Toutes ces commandes peuvent être imbriquées les unes dans les autres pour obtenir l'effet souhaité. Par exemple, vous pouvez imbriquer une commande If...Then dans une commande Select...Case :`Select Case value`

```
Case 42
  If value2 > 3 Then
    MessageBox("Cancel")
  End If
End Select
```