การควบคุมทิศทางการทำงานของโปรแกรม

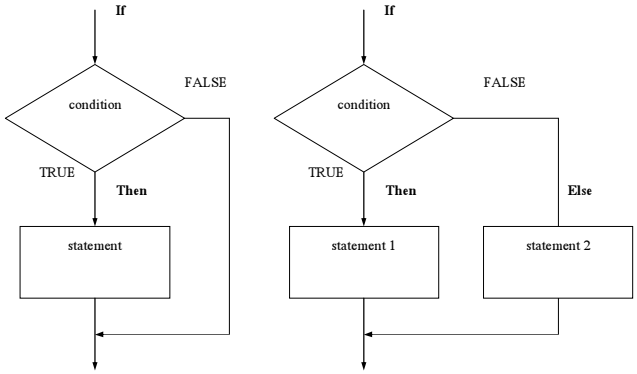
เรียกว่า Control-Flow เป็นสิ่งที่ผู้เขียนโปรแกรมต้องทำความเข้าใจ แล้วเลือกการควบคุมทิศทางให้เหมาะสมกับโปรแกรมของตนเอง

การควบคุมทิศทางการทำงานของโปรแกรม

- มี 2 รูปแบบ คือ
  1. **การตัดสินใจ (Decision)**
     เป็นการเลือกทางใดทางหนึ่งจากตัวเลือกที่มีให้ โดยการเลือกจะพิจารณาจากตัวแปรที่เป็นเงื่อนไขในการเลือก ซึ่งจะใช้ตัวดำเนินการแบบต่างๆ มาใช้ในการตัดสินใจเลือก
  2. **การทำงานแบบวนซ้ำ (Iteration)**
     เป็นการทำงานแบบซ้ำไปซ้ำมาตามเงื่อนไขที่กำหนดไว้

การตัดสินใจเลือกจาก 2 ทางเลือก

- ใช้คำสั่ง If…Then…Else สามารถเขียนแผนภูมิการทำงาน (Flow Chart) ได้ดังนี้

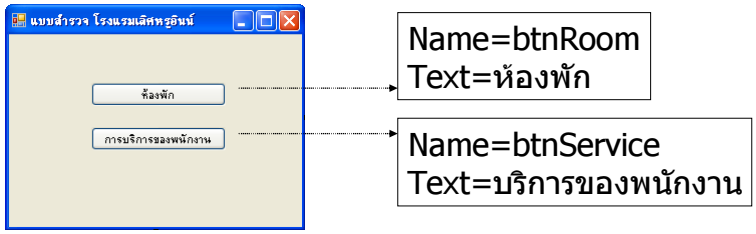


แผนภูมิการทำงาน คำสั่ง If...Then และ If...Then...Else

ต.ย.ที่ 2.1 การใช้คำสั่ง `If...Then...Else`

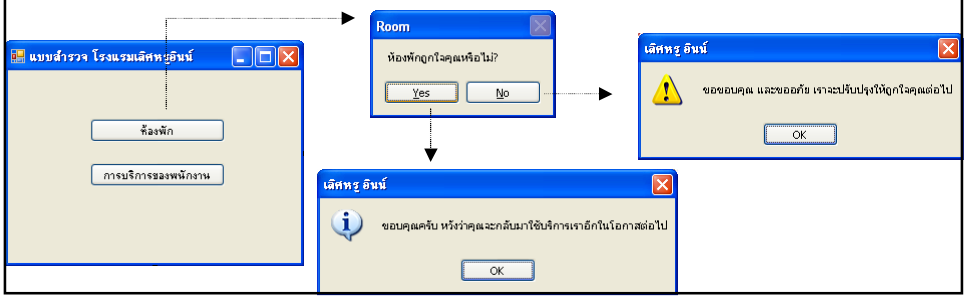
```
Dim Counter As Byte      ' ประกาศตัวแปรชื่อ Counter เป็นประเภทไบท์
Counter = 0              ' กำหนดให้ Counter มีค่าเป็น 0
If Counter > 0 Then      ' ตรวจสอบว่าตัวแปร Counter มีค่ามากกว่า 0 หรือไม่
    MessageBox.Show("Greater than zero")
Else
    MessageBox.Show("Less than or equal zero ")
End If
```

ต.ย.ที่ 2.2 การใช้คำสั่ง `If...Then...Else`



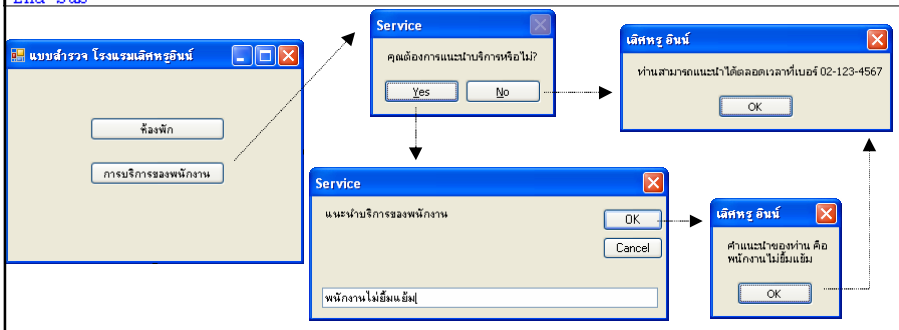
Coding ที่ปุ่ม btnRoom

```
Private Sub btnRoom_Click(ByVal sender As System.Object, ByVal e As System.EventArgs) Hand
    Dim Resp As MsgBoxResult 'เป็นชนิดข้อมูลที่เก็บผลจากการคลิกปุ่มใน MessageBox
    Dim Out As String
    Resp = MessageBox.Show("ห้องพักถูกใจคุณหรือไม่?", "Room", MessageBoxButtons.YesNo)
    If Resp = MsgBoxResult.Yes Then
        Out = "ขอบคุณครับ หวังว่าคุณจะกลับมาใช้บริการเราอีกในโอกาสต่อไป"
        MessageBox.Show(Out, "เลิศหรู อินน์", MessageBoxButtons.OK, MessageBoxIcon.Information)
    Else
        Out = "ขอขอบคุณ และขออภัย เราจะปรับปรุงให้ถูกใจคุณต่อไป"
        MessageBox.Show(Out, "เลิศหรู อินน์", MessageBoxButtons.OK, MessageBoxIcon.Exclamation)
    End If
End Sub
```



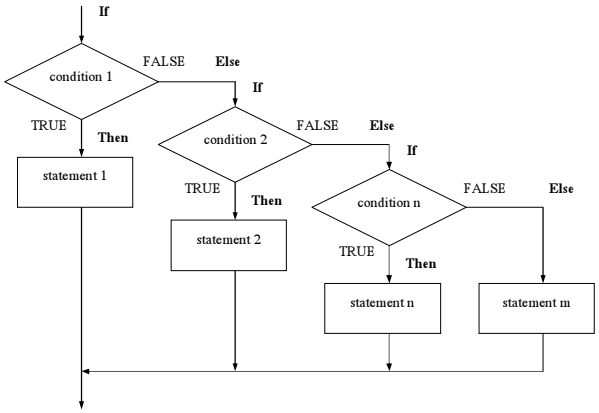
Coding ที่ปุ่ม btnService

```
Private Sub btnService_Click(ByVal sender As System.Object, ByVal e As System.EventArgs)
    Dim Resp As MsgBoxResult
    Dim Comment As String
    Resp = MessageBox.Show("คุณต้องการแนะนำบริการหรือไม่?", "Service", MessageBoxButtons.YesNo)
    If Resp = MsgBoxResult.Yes Then
        Comment = InputBox("แนะนำบริการของพนักงาน", "Service")
        MessageBox.Show("คำแนะนำของท่าน คือ " & vbCrLf & Comment, "เลิศหรู อินน์")
    End If
    MessageBox.Show("ท่านสามารถแนะนำได้ตลอดเวลาที่เบอร์ 02-123-4567", "เลิศหรู อินน์")
End Sub
```



การใช้ If…Then…Else ซ้อนกัน

- ชุดคำสั่ง If สามารถนำมาซ้อนกันได้ (nested if statement) สามารถเขียนเป็นแผนภูมิการทำงาน ได้ดังนี้



ต.ย. ต้องการเก็บชื่อวันภาษาอังกฤษทั้ง 7 วันไว้ในตัวแปรประเภทสตริงสามารถเขียนเป็นคำสั่งได้ดังนี้

การใช้ If...Then ปกติ (ไม่ซ้อนกัน)

```
Dim DayNum As Byte, DayName As String
If DayNum = 1 Then
    DayName = "Mon"
End If
If DayNum = 2 Then
    DayName = "Tue"
End If

If DayNum = 7 Then
    DayName = "Sun"
End If
```

การใช้ If...Then...Else ซ้อนกัน

```
Dim DayNum As Byte, DayName As String
If DayNum = 1 Then
    DayName = "Mon"
Else
    If DayNum = 2 Then
        DayName = "Tue"
    Else

        If DayNum = 7 Then
            DayName = "Sun"
        End If

    End If
End If
```

If…Then…ElseIf

- ในกรณีมีหลายเงื่อนไขที่ต้องการตรวจสอบนอกจากใช้คำสั่ง If ซ้อนกันแล้วยังสามารถใช้ ElseIf ซึ่งเป็นคำสั่งสำหรับเลือกทำงานในกรณีใดกรณีหนึ่งจากหลายๆ กรณี โดยแต่ละกรณีจะมีเงื่อนไขเฉพาะกำหนดไว้ซึ่งมีรูปแบบการใช้งานดังนี้

```
If <เงื่อนไขที่ 1> Then
   <ชุดคำสั่งเมื่อเงื่อนไขที่ 1 เป็นจริง>
ElseIf <เงื่อนไขที่ 2> Then
   <ชุดคำสั่งเมื่อเงื่อนไขที่ 2 เป็นจริง>
...
ElseIf <เงื่อนไขที่ n> Then
   <ชุดคำสั่งเมื่อเงื่อนไขที่ n เป็นจริง>
Else
   <ชุดคำสั่งเมื่อทุกเงื่อนที่ผ่านมาเป็นเท็จ>
End If
```

ต.ย. ต้องการเก็บชื่อวันภาษาอังกฤษทั้ง 7 วันไว้ในตัวแปรประเภทสตริงสามารถเขียนเป็นคำสั่งได้ดังนี้

การใช้ If...Then...ElseIf ซ้อนกัน

```
Dim DayNum As Byte, DayName As String
If DayNum = 1 Then
    DayName = "Mon"
ElseIf DayNum = 2 Then
    DayName = "Tue"

ElseIf DayNum = 6 Then
    DayName = "Fri"
Else
    DayName = "Sun"
End If
```

ต.ย.ที่ 2.3 การแสดงผลการสอบจากคะแนน

- มีเกณฑ์การคำนวณผลการสอบ ดังตาราง

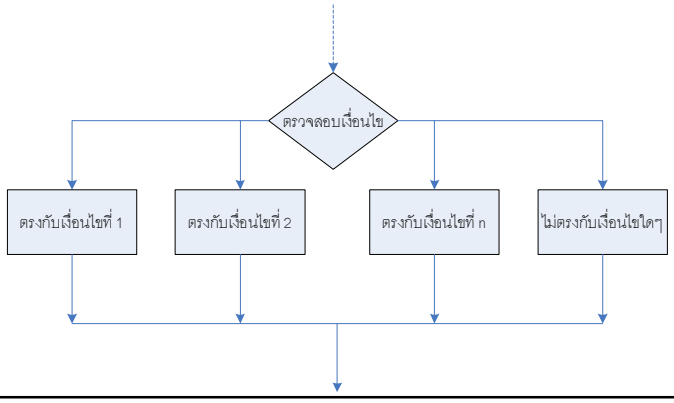
ช่วงคะแนน (0-100)	ผลการสอบ
80-100	EXCELLENT (ยอดเยี่ยม)
70-79	GOOD (ดี)
60-69	FAIR (พอใช้)
0-59	FAIL (ไม่ผ่าน)

Coding ที่ form_load

```
Dim Score As Byte, ConvertScore As String
'  รับค่าคะแนนสอบ
Score = CInt(InputBox("Enter  student  score  [0-100]  :  "))
If Score >= 80 Then
    ConvertScore = "EXCELLENT"
ElseIf Score >= 70 Then
    ConvertScore = "GOOD"
ElseIf Score >= 60 Then
    ConvertScore = "FAIR"
Else
    ConvertScore = "FAIL"
End If
'แสดงผลสอบ
MessageBox.Show(ConvertScore)
```

การตัดสินใจเลือกมากกว่า 2 ทางเลือก

- Select…Case เป็นคำสั่งที่เหมาะกับการตัดสินใจเลือกทางเลือกที่มีมากกว่า 2 ทาง ซึ่งจะมีการทดสอบเงื่อนไขก่อนว่าตรงกับตัวเลือกใดแล้วจึงทำงานตามคำสั่งที่ต่อท้ายทางเลือกนั้น



รูปแบบคำสั่งของ Select…Case

Select Case ตรวจสอบเงื่อนไข
  Case เงื่อนไขที่ 1 : <คำสั่งที่ให้ทำเมื่อเงื่อนไขที่ 1 เป็นจริง>
  Case เงื่อนไขที่ 2 : <คำสั่งที่ให้ทำเมื่อเงื่อนไขที่ 2 เป็นจริง>

  …

  Case เงื่อนไขที่ n : <คำสั่งที่ให้ทำเมื่อเงื่อนไขที่ n เป็นจริง>
  Case Else <เมื่อไม่ตรงกับเงื่อนไขใด จะทำงานหลังคำสั่ง Else>
End Case

ต.ย.ที่ 2.4 การใช้งานคำสั่ง Select...Case

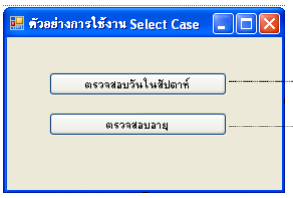
- ต้องการตรวจสอบค่าตัวแปร Number ว่าตรงกับเงื่อนไขของ Case ใด โดยมีรายละเอียดดังนี้

  1) ถ้ามีค่าระหว่าง 1 ถึง 5 ให้ปรากฏไดอะล็อกแสดงข้อความว่า
     Number between 1 to 5
  2) ถ้ามีค่าเป็น 6, 7, 8 หรือ 9 10 ให้ปรากฏไดอะล็อกแสดงข้อความว่า
     Number between 6 to 9
  3) ถ้ามีค่าเท่ากับ 10 ให้ปรากฏไดอะล็อกแสดงข้อความว่า
     Number equal 10
  4) ถ้าไม่ตรงกับเงื่อนไขใดๆ ข้างต้นให้ปรากฏไดอะล็อกแสดงข้อความว่า
     Number NOT between 1 to 10

Coding ที่ form_load

```
Dim Number As Short
Number = CShort(InputBox("Enter  Number  [1-10]  :  "))
Select Case Number
    Case 1 To 5
        MessageBox.Show("Number  between  1  to  5")
    Case 6, 7, 8, 9
        MessageBox.Show("Number  between  6  to  9")
    Case Is = 10
        MessageBox.Show("Number equal 10")
    Case Else
        MessageBox.Show("Number NOT between 1 to 10")
End Select
```

ต.ย.ที่ 2.5 การใช้งานคำสั่ง Select...Case



Name=btnCheckDay
Text=ตรวจสอบวันในสัปดาห์

Name=btnCheckAge
Text=ตรวจสอบอายุ

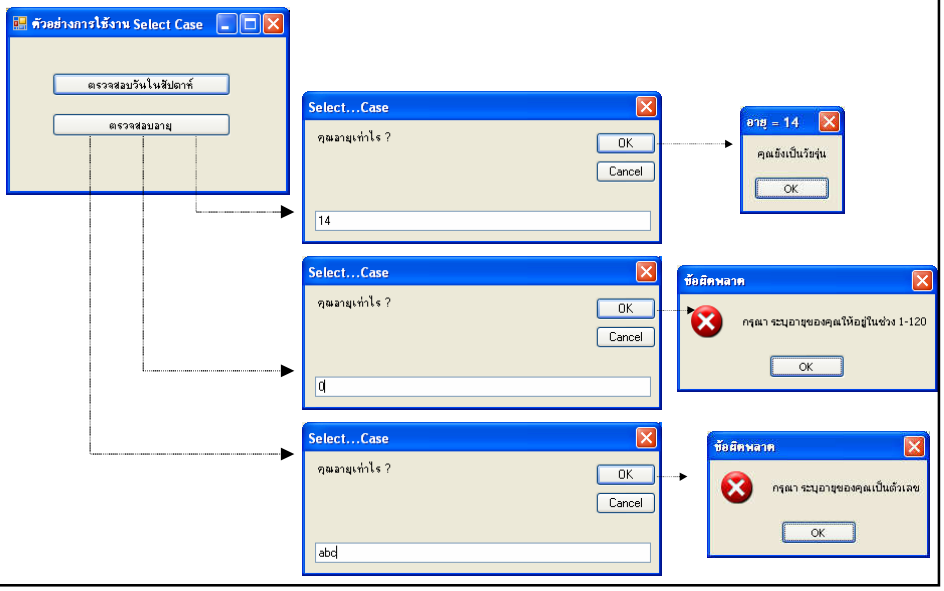
Coding ที่ปุ่ม btnCheckDay

```
Dim day As Integer = Now.DayOfWeek
Select Case day
    Case 0
        btnCheckDay.Text = "วันนี้เป็นวันอาทิตย์ ที่ " & Now.Day.ToString
        Me.BackColor = Color.Red
    Case 1
        btnCheckDay.Text = "วันนี้เป็นวันจันทร์ ที่ " & Now.Day.ToString
        Me.BackColor = Color.Yellow
    Case 2
        btnCheckDay.Text = "วันนี้เป็นวันอังคาร ที่ " & Now.Day.ToString
        Me.BackColor = Color.Pink
End Select
```

Coding ที่ปุ่ม btnCheckAge

```
Dim Age As Integer
Dim Resp As String = InputBox("คุณอายุเท่าไร ?", "Select...Case")
If IsNumeric(Resp) Then
    Age = CInt(Resp)
    Select Case Age
        Case 1 To 12
            MessageBox.Show("คุณยังเด็กมาก", "อายุ = " & Age.ToString)
        Case 13 To 24
            MessageBox.Show("คุณยังเป็นวัยรุ่น", "อายุ = " & Age.ToString)
        Case 25
            MessageBox.Show("คุณอยู่ในวัยเบญจเพศ", "อายุ = " & Age.ToString)
        Case 26 To 60
            MessageBox.Show("คุณอยู่ในวัยทำงาน", "อายุ = " & Age.ToString)
        Case 61 To 120
            MessageBox.Show("คุณอยู่ในวัยชรา", "อายุ = " & Age.ToString)
        Case Else
            MessageBox.Show("กรุณา ระบุอายุของคุณให้อยู่ในช่วง 1-120", "ข้อผิดพลาด", _
            MessageBoxButtons.OK, MessageBoxIcon.Error)
    End Select
Else
    MessageBox.Show("กรุณา ระบุอายุของคุณเป็นตัวเลข", "ข้อผิดพลาด", _
    MessageBoxButtons.OK, MessageBoxIcon.Error)
End If
```

ผลการทำงานของปุ่ม btnCheckAge



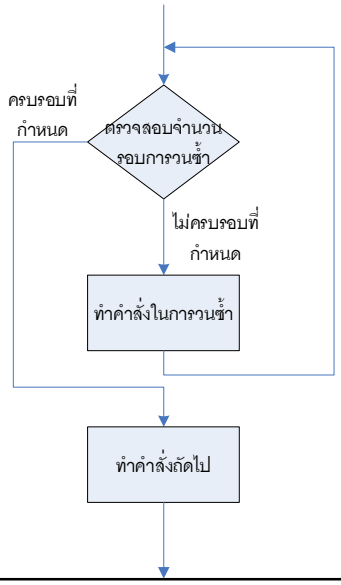
การวนซ้ำ

เป็นการทำงานเดิมซ้ำๆ ซึ่งมี 2 รูปแบบ คือ การวนซ้ำด้วยจำนวนรอบที่แน่นอน และการวนซ้ำด้วยจำนวนรอบที่ไม่แน่นอน

การวนซ้ำด้วยจำนวนรอบที่แน่นอน

- For…Next ใช้สำรับการวนซ้ำด้วยจำนวนรอบที่แน่นอน โดยมีการใช้งานตัวแปร 1 ตัว สำหรับนับจำนวนรอบว่าวนซ้ำตามรอบที่กำหนดหรือไม่
- ตัวแปรที่ทำหน้าที่เป็นตัวนับรอบในการวนซ้ำ สามารถนับได้ทั้งแบบเดินหน้า (เพิ่มขึ้น) และแบบถอยหลัง (ลดลง) นอกจากนี้ยังสามารถกำหนดขั้นการนับได้ว่าจะให้เพิ่มขึ้น หรือลดลงทีละเท่าไร (ถ้าไม่กำหนดจะเพิ่มขึ้นหรือลดลงทีละ 1)

การวนซ้ำด้วยจำนวนรอบที่แน่นอน



รูปแบบคำสั่ง For…Next

For ตัวแปรนับรอบ = จำนวนรอบเริ่มต้น To จำนวนรอบสุดท้าย [Step]
<คำสั่งที่ต้องการให้ทำซ้ำ>
Next ตัวแปรนับรอบ

- Step หมายถึง ขั้นที่ต้องการให้ตัวแปรเพิ่มขึ้น

การวนซ้ำแบบถอยหลัง มีรูปแบบดังนี้

For ตัวแปรนับรอบ = จำนวนรอบสุดท้าย To จำนวนรอบเริ่มต้น [Step]
<คำสั่งที่ต้องการให้ทำซ้ำ>
Next ตัวแปรนับรอบ

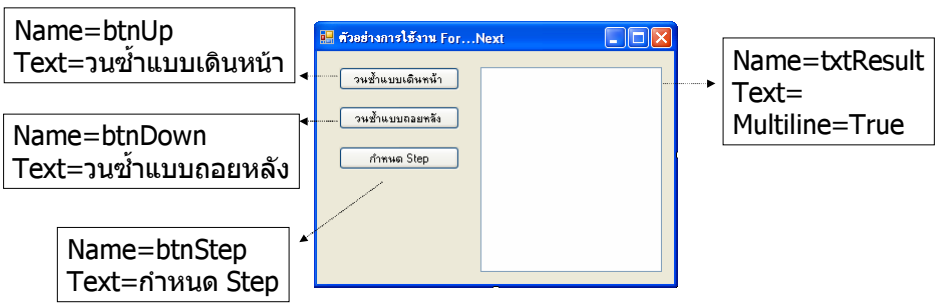
- Step หมายถึง ขั้นที่ต้องการให้ตัวแปรลดลง

ต.ย. การใช้งานคำสั่ง For…Next

- เป็นการหาผลบวกของตัวเลขจำนวนเต็มตั้งแต่ 1 ถึง 10 โดยใช้คำสั่ง For... Next แล้วแสดงผลลัพธ์ที่ได้ทางกล่องข้อความ

```
Dim Number, Sum As Integer
Number = 0 : Sum = 0
For Number = 1 To 10
    Sum += Number
Next
MessageBox.Show("Summary of 1 to 10 =" & Sum)
```

ต.ย.ที่ 2.6 การใช้งานคำสั่ง For…Next

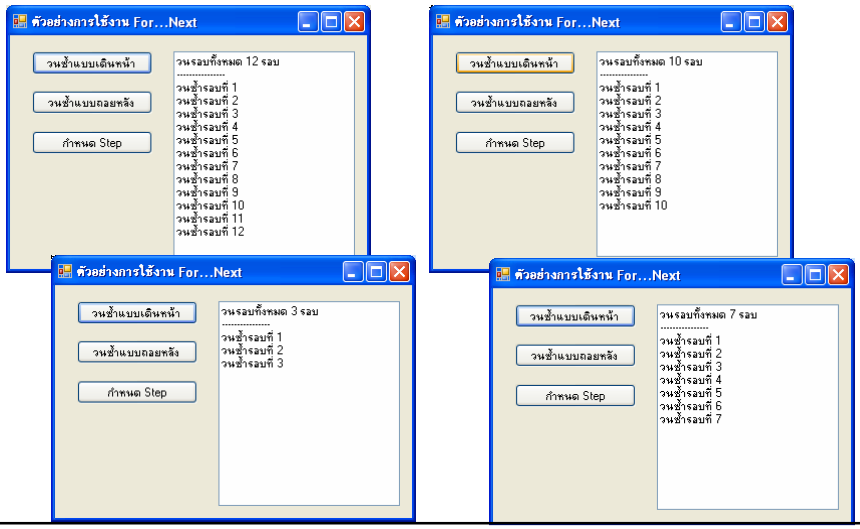


Coding ที่ปุ่ม btnUp

```
Dim Num, Count As Integer
Count = Int((10 * Rnd()) + 3)    'สุ่มเลขระหว่าง 3-12 มาเก็บไว้ในตัวแปร count
txtResult.Text = ""  'เคลียร์ข้อความใน textbox
txtResult.Text += "วนรอบทั้งหมด " & CStr(Count) & " รอบ" & vbCrLf
txtResult.Text += "----------------" & vbCrLf
For Num = 1 To Count
    txtResult.Text += "วนซ้ำรอบที่ " & Num.ToString & vbCrLf
Next
```

ผลลัพธ์การทำงานของปุ่ม btnUp

จำนวนรอบจะสุ่มมาจากเลข 3 ถึง 12 (ตัวแปร count)

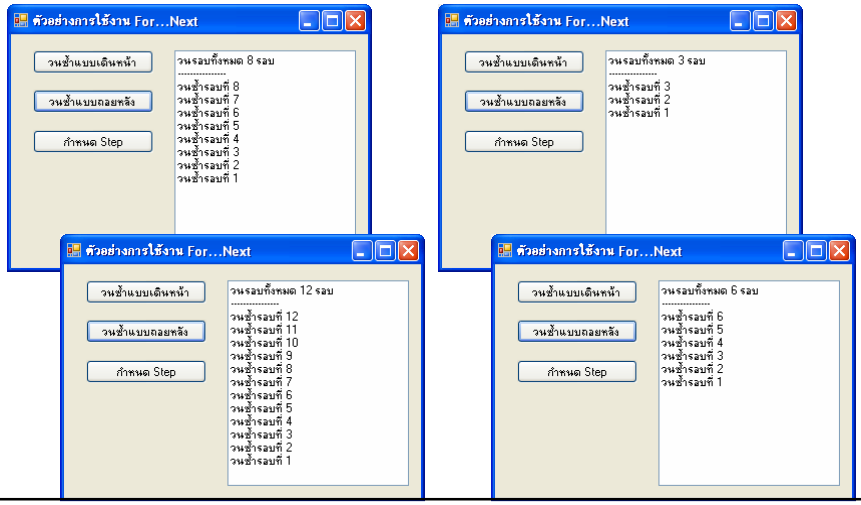


Coding ที่ปุ่ม btnDown

```
Dim Num, Count As Integer
Count = Int((10 * Rnd()) + 3)    'สุ่มเลขระหว่าง 3-12 มาเก็บไว้ในตัวแปร count
txtResult.Text = ""  'เคลียร์ข้อความใน textbox
txtResult.Text += "วนรอบทั้งหมด " & CStr(Count) & " รอบ" & vbCrLf
txtResult.Text += "----------------" & vbCrLf
For Num = Count To 1 Step -1
    txtResult.Text += "วนซ้ำรอบที่ " & Num.ToString & vbCrLf
Next
```

ผลลัพธ์การทำงานของปุ่ม btnDown

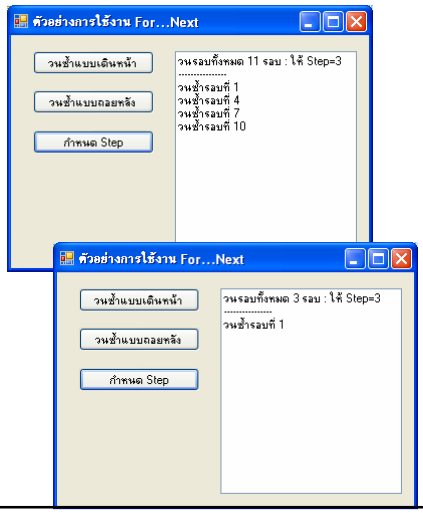
จำนวนรอบจะสุ่มมาจากเลข 3 ถึง 12 (ตัวแปร count)

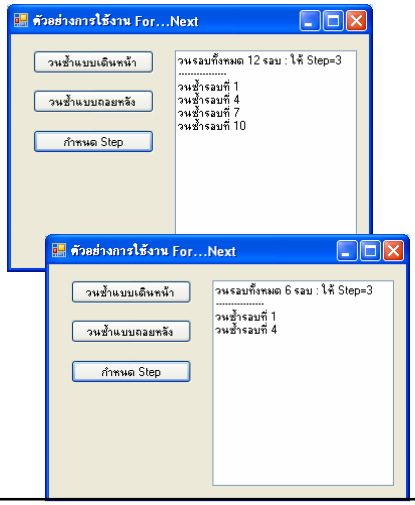


Coding ที่ปุ่ม btnStep

```
Dim Num, Count As Integer
Count = Int((10 * Rnd()) + 3)    'สุ่มเลขระหว่าง 3-12 มาเก็บไว้ในตัวแปร count
txtResult.Text = ""  'เคลียร์ข้อความใน textbox
txtResult.Text += "วนรอบทั้งหมด " & CStr(Count) & " รอบ : ให้ Step=3" & vbCrLf
txtResult.Text += "----------------" & vbCrLf
For Num = 1 To Count Step 3
    txtResult.Text += "วนซ้ำรอบที่ " & Num.ToString & vbCrLf
Next
```

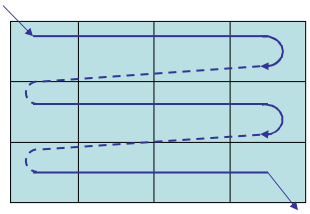
ผลลัพธ์การทำงานของปุ่ม btnStep



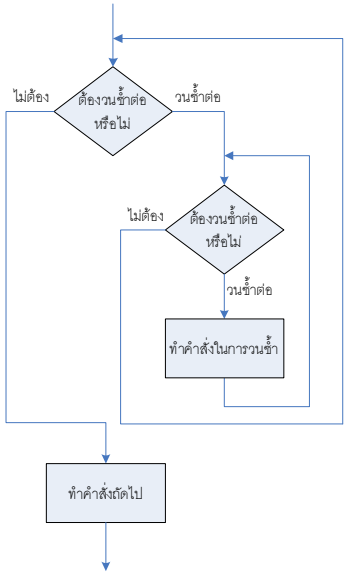


การใช้คำสั่ง For…Next แบบซ้อนกัน

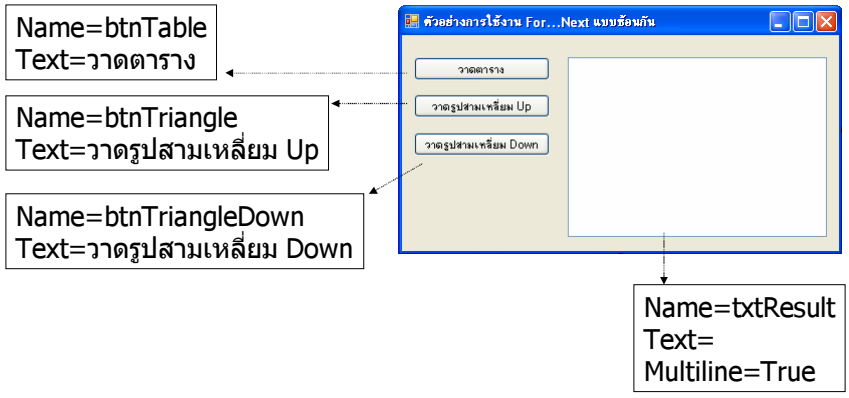
- การวนซ้ำแบบซ้อนกันเป็นการเขียนโปรแกรมที่ซับซ้อน เช่น การเติมข้อมูลในตาราง 2 มิติ ต้องมีการเติมข้อมูลทีละแถว (เป็นการวนซ้ำตามจำนวนรอบของแถว) ซึ่งแต่ละแถวต้องเติมข้อมูลทีละคอลัมน์ (เป็นการวนซ้ำตามจำนวนคอลัมน์) โดยในการวนซ้ำจะเริ่มจากการวนซ้ำทีละคอลัมน์ก่อน เมื่อครบ 1 แถว ก็จะขึ้นแถวใหม่แล้ววนซ้ำจนครบทุกๆ แถว



การใช้คำสั่ง For…Next แบบซ้อนกัน



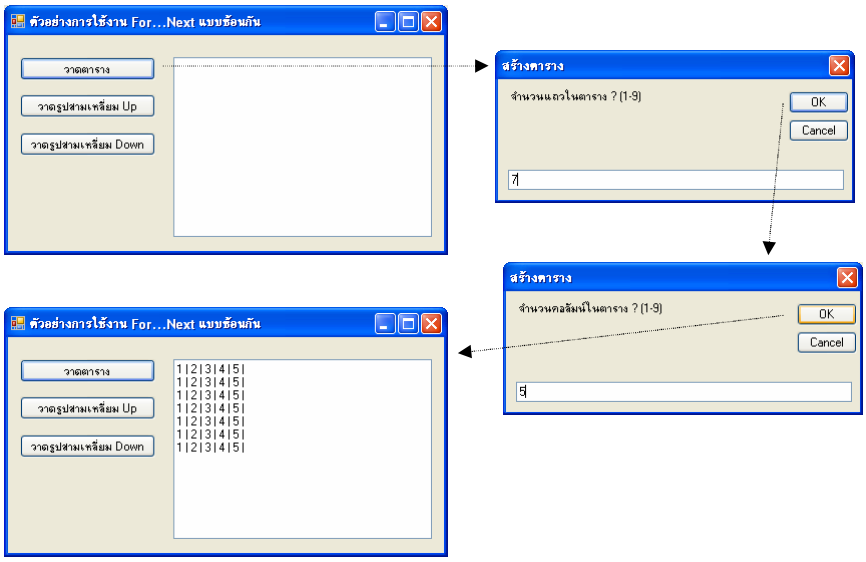
ต.ย.ที่ 2.7 การใช้ For…Next แบบซ้อนกัน



Coding ที่ปุ่ม btnTable

```
Dim Row, Column As Integer
Dim RowInput As Integer = Int(InputBox("จำนวนแถวในตาราง ? (1-9)", "สร้างตาราง"))
Dim ColInput As Integer = Int(InputBox("จำนวนคอลัมน์ในตาราง ? (1-9)", "สร้างตาราง"))
If (RowInput < 1 Or RowInput > 9) Or (ColInput < 1 Or ColInput > 9) Then
    MessageBox.Show("กรุณา เลือกแถวและคอลัมน์เป็นตัวเลขระหว่าง 1-9", "ข้อผิดพลาด", _
    MessageBoxButtons.OK, MessageBoxIcon.Error)
Else
    txtResult.Text = ""
    For Row = 1 To RowInput
        For Column = 1 To ColInput
            txtResult.Text += Column.ToString & " | "
        Next
        txtResult.Text += vbCrLf     'แสดงผลในแถวใหม่ของตาราง
    Next
End If
```

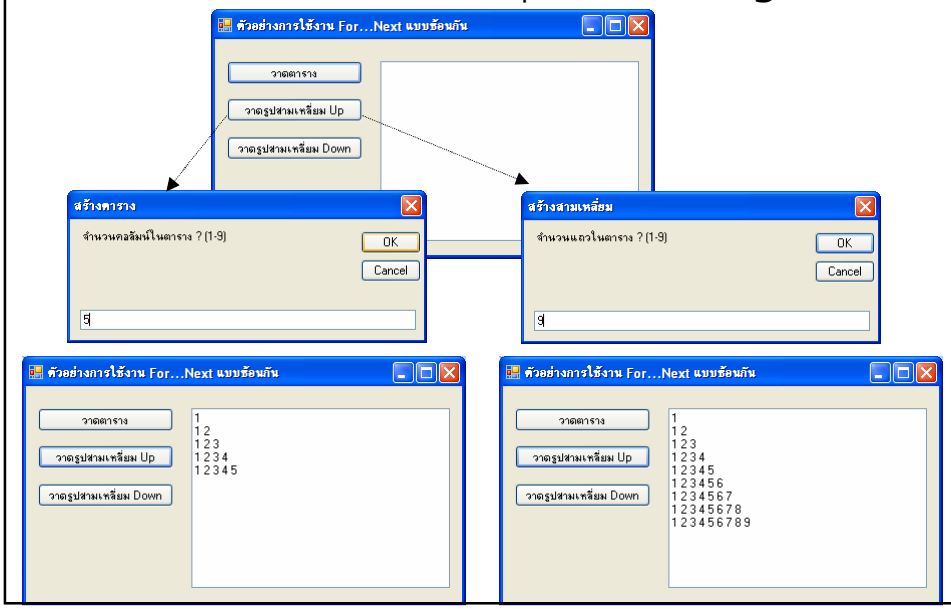
ผลการทำงานของปุ่ม btnTable



Coding ที่ปุ่ม btnTriangle

```
Dim Row, Column As Integer
Dim RowInput As Integer = Int(InputBox("จำนวนแถวในตาราง ? (1-9)", "สร้างสามเหลี่ยม"))
If RowInput < 1 Or RowInput > 9 Then
    MessageBox.Show("กรุณา เลือกแถวเป็นตัวเลขระหว่าง 1-9", "ข้อผิดพลาด", _
    MessageBoxButtons.OK, MessageBoxIcon.Error)
Else
    txtResult.Text = ""
    For Row = 1 To RowInput
        For Column = 1 To Row    'จำนวนการวนซ้ำในรอบใน ไม่ตายตัว
            txtResult.Text += Column.ToString & " "
        Next
        txtResult.Text += vbCrLf     'แสดงผลในแถวใหม่ของรูปสามเหลี่ยม
    Next
End If
```

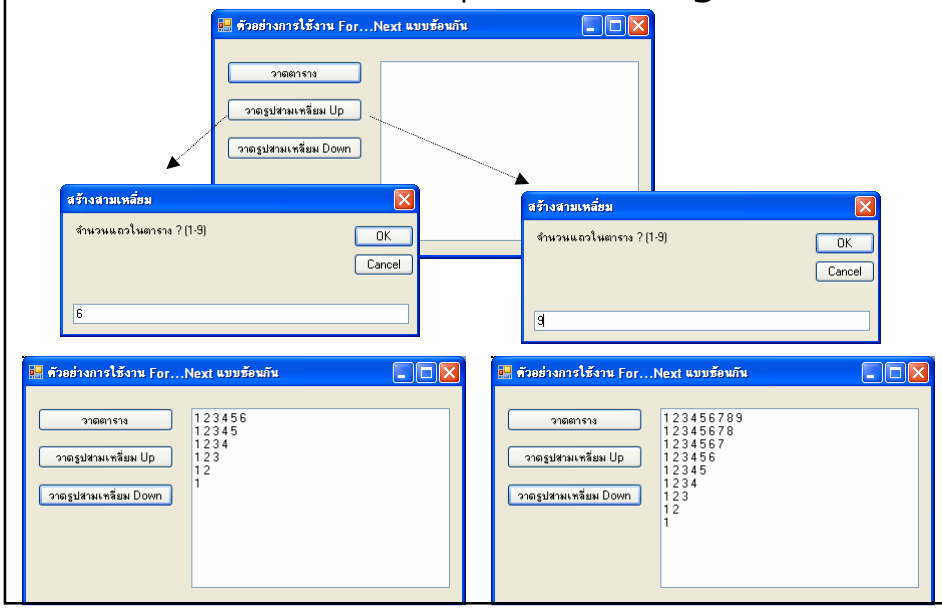
ผลการทำงานของปุ่ม btnTriangle



Coding ที่ปุ่ม btnTriangleDown

```
Dim Row, Column As Integer
Dim RowInput As Integer = Int(InputBox("จำนวนแถวในตาราง ? (1-9)", "สร้างสามเหลี่ยม"))
If RowInput < 1 Or RowInput > 9 Then
    MessageBox.Show("กรุณา เลือกแถวเป็นตัวเลขระหว่าง 1-9", "ข้อผิดพลาด", _
    MessageBoxButtons.OK, MessageBoxIcon.Error)
Else
    txtResult.Text = ""
    For Row = RowInput To 1 Step -1     'ใช้การวนซ้ำแบบย้อนกลับ
        For Column = 1 To Row    'จำนวนการวนซ้ำในรอบใน ไม่ตายตัว
            txtResult.Text += Column.ToString & " "
        Next
        txtResult.Text += vbCrLf     'แสดงผลในแถวใหม่ของรูปสามเหลี่ยม
    Next
End If
```

ผลการทำงานของปุ่ม btnTriangleDown



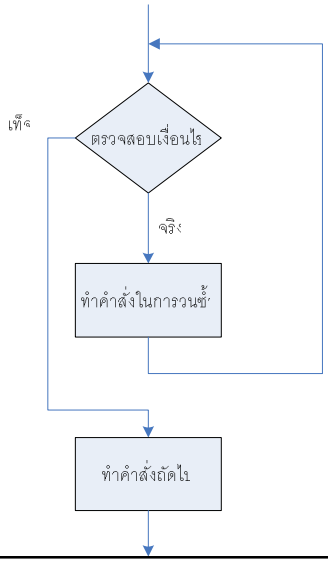
การวนซ้ำด้วยจำนวนรอบที่ไม่แน่นอน

- การวนซ้ำด้วยจำนวนรอบที่ไม่แน่นอน จะใช้วิธีตรวจสอบเงื่อนไขว่าต้องวนซ้ำหรือไม่ ซึ่งมีคำสั่งให้เลือกใช้งานหลายรูปแบบ ดังนี้

1. While…End While
2. Do…Loop
   - Do While…Loop
   - Do Until…Loop
   - Do…Loop While
   - Do…Loop Until

While…End While

- เป็นการวนซ้ำที่ไม่สามารถบอกจำนวนรอบที่แน่นอนได้ ซึ่งจะวนซ้ำถึงเมื่อใดนั้น จะใช้การตรวจสอบเงื่อนไขก่อนการวนซ้ำว่าต้องวนซ้ำอีกหรือไม่ ถ้าเงื่อนไขยังคงเป็นจริงก็จะวนซ้ำต่อ แต่ถ้าเงื่อนไขเป็นเท็จจะออกจากการวนซ้ำทันที



รูปแบบคำสั่ง While…End While

While <ตรวจสอบเงื่อนไขว่าจริงหรือเท็จ>
<ถ้าเงื่อนไขเป็นจริง ให้ทำงานตามคำสั่ง>
End While

Do While…Loop

- การทำงานของคำสั่งนี้เหมือน While…End While ทุกประการ มีรูปแบบคำสั่งดังนี้

Do While <ตรวจสอบเงื่อนไขว่าจริงหรือเท็จ>
    <ถ้าเงื่อนไขเป็นจริง ให้ทำงานตามคำสั่ง>
Loop

Do Until…Loop

- การทำงานของคำสั่งนี้ จะมีการตรวจสอบเงื่อนไขตรงกันข้ามกับ While…End While คือ จะหยุดทำการวนซ้ำเมื่อเงื่อนไขเป็นจริง (ถ้าเป็นเท็จก็วนซ้ำต่อไป) มีรูปแบบคำสั่ง ดังนี้

Do Until <ตรวจสอบเงื่อนไขว่าจริงหรือเท็จ>
    <ถ้าเงื่อนไขเป็นเท็จ ให้ทำงานตามคำสั่ง>
Loop

Do…Loop While

- คำสั่งนี้ จะเปลี่ยนตำแหน่งการตรวจสอบเงื่อนไขการวนซ้ำไปอยู่ส่วนท้ายของการทำงาน นั่นคือ จะต้องมีการวนซ้ำอย่างน้อย 1 ครั้งแล้วจึงตรวจสอบเงื่อนไข และจะหยุดการวนซ้ำเมื่อเงื่อนไขเป็นเท็จ มีรูปแบบคำสั่ง ดังนี้

Do
    <ทำงานตามคำสั่ง>
Loop While **<ตรวจสอบเงื่อนไขว่าจริงหรือเท็จ ถ้าจริงให้ทำซ้ำ>**

Do…Loop Until

- คำสั่งนี้ จะเปลี่ยนตำแหน่งการตรวจสอบเงื่อนไขการวนซ้ำไปอยู่ส่วนท้ายของการทำงาน นั่นคือ จะต้องมีการวนซ้ำอย่างน้อย 1 ครั้งแล้วจึงตรวจสอบเงื่อนไข และจะหยุดการวนซ้ำเมื่อเงื่อนไขเป็นจริง มีรูปแบบคำสั่ง ดังนี้

Do
    <ทำงานตามคำสั่ง>
Loop Until **<ตรวจสอบเงื่อนไขว่าจริงหรือเท็จ ถ้าเท็จให้ทำซ้ำ>**

ข้อสังเกต

- จากการทำงานของคำสั่ง While และ Until สรุปได้ว่า

1. While จะหยุดทำการวนซ้ำเมื่อตรวจสอบเงื่อนไขแล้วเป็นเท็จ
2. Until จะหยุดทำการวนซ้ำเมื่อตรวจสอบเงื่อนไขแล้วเป็นจริง

- การเขียนโปรแกรมที่มีการวนซ้ำ อาจเกิดข้อผิดพลาดที่เกิดขึ้นได้บ่อย คือ **การวนซ้ำแบบไม่มีที่สิ้นสุด** นั่นคือ เงื่อนไขที่ไม่มีทางเป็นจริงได้ ซึ่งโปรแกรมจะแสดงข้อผิดพลาด (Error) เช่น OverflowException was unhandled. ขึ้นมาให้ทราบ

การกระโดดออกจากการวนซ้ำ

- บางครั้งจำเป็นต้องให้โปรแกรมกระโดดออกจากการวนซ้ำด้วยเงื่อนไขบางประการ โดยใช้คำสั่ง Exit แล้วตามด้วยคำสั่งที่จะกระโดดออกมา ดังรูปแบบต่อไปนี้
  - **Exit For** เป็นการกระโดดออกจากการวนซ้ำที่ใช้งานคำสั่ง For…Next โดยไม่สนใจว่าจะมีการวนซ้ำครบตามจำนวนรอบที่กำหนดหรือไม่
  - **Exit Do** เป็นการกระโดดออกจากการวนซ้ำที่ใช้งานคำสั่ง Do…Loop โดยไม่สนใจว่าเงื่อนไขจะเป็นจริงหรือเท็จ

ต.ย. การใช้งานคำสั่ง While...End While

```
Dim Num As Byte, StrNum As String
Num = 1 : StrNum = ""
While (Num < 10)
    StrNum &= CStr(Num) & vbCrLf
    Num += 1
End While
MessageBox.Show(StrNum)
```

- โปรแกรมทำการวนซ้ำ คือกำหนดให้ตัวแปร StrNum เก็บค่าของตัวแปร Num ที่แปลงค่าเป็นข้อความโดยใช้ฟังก์ชัน CStr แล้วจึงขึ้นบรรทัดใหม่ จากนั้นให้เพิ่มค่าตัวแปร Num รอบละหนึ่ง โดยทำงานดังกล่าวนี้ไปเรื่อยๆ จนกว่าเงื่อนไข Num น้อยกว่า 10 เป็นเท็จจึงออกจากลูป เมื่อออกจากลูปแล้วจึงแสดงค่าตัวแปร StrNum ในไดอะล็อกแสดงข้อความซึ่งจะแสดงตัวเลข 1 ถึง 9
  นั่นคือตัวอย่างนี้คำสั่ง While ทำงานทั้งหมด 9 รอบ

ต.ย. การใช้งานคำสั่ง Do While...Loop

```
Dim Num As Byte, StrNum As String
Num = 1 : StrNum = ""
Do While (Num < 10)
    StrNum &= CStr(Num) & vbCrLf
    Num += 1
Loop
MessageBox.Show(StrNum)
```

ต.ย. การใช้งานคำสั่ง Do Until...Loop

```
Dim Num As Byte, StrNum As String
Num = 1 : StrNum = ""
Do Until (Num > 9)
     StrNum &= CStr(Num) & vbCrLf
     Num += 1
Loop
MessageBox.Show(StrNum)
```

ต.ย. การใช้งานคำสั่ง Do...Loop While

```
Dim Num As Byte, StrNum As String
Num = 1 : StrNum = ""
Do
     StrNum &= CStr(Num) & vbCrLf
     Num += 1
Loop While (Num < 10)
MessageBox.Show(StrNum)
```

ต.ย. การใช้งานคำสั่ง Do...Loop Until

```
Dim Num As Byte, StrNum As String
Num = 1 : StrNum = ""
Do
    StrNum &= CStr(Num) & vbCrLf
    Num += 1
Loop Until (Num > 9)
MessageBox.Show(StrNum)
```

ต.ย. การใช้งานคำสั่งการวนซ้ำแบบซ้อนกัน

- โครงสร้างคำสั่งที่ใช้ในการทำงานวนซ้ำคือ While...End, While Do...Loop, For…Next และ For Each…Next สามารถนำมาซ้อนกันได้เหมือนกับคำสั่ง If ซึ่งเรียกว่า **ลูปซ้อนกัน** (nested loop) แสดงตัวอย่างการใช้คำสั่ง Do Until...Loop ซ้อนกันสองชุดเพื่อพิมพ์ตัวเลขเป็นตารางสูตรคูณแม่หนึ่งถึงแม่สาม
- จากตัวอย่างมีการกำหนดตัวแปรประเภทไบท์จำนวนสองตัวเพื่อควบคุมการเปลี่ยนแถวและคอลัมน์ของตารางสูตรคูณซึ่งมีตัวแปร Row ทำหน้าที่ควบคุมแถวคือ 1 ถึง 3 และตัวแปร Column ทำหน้าที่ควบคุมคอลัมน์คือ 1 ถึง 12 ส่วนตัวแปร StrNum ทำหน้าที่เก็บค่าข้อความที่ได้

ต.ย.ที่ การใช้งานคำสั่งการวนซ้ำแบบซ้อนกัน

```
'  Multiplication table 1 to 3
Dim Row, Column As Byte
Dim StrNum As String
Row = 1 : StrNum = ""
Do Until (Row > 12)
    Column = 1
    Do Until (Column > 3)
        StrNum += Column & "  x  " & Row & "  =    " _
         & Row * Column & "       " & vbTab
        Column += 1
    Loop
    StrNum += vbLf
    Row += 1
Loop
MessageBox.Show(StrNum)
```

โปรแกรมย่อย

โปรแกรมย่อย

- เป็นส่วนของโปรแกรมที่ถูกร้างขึ้นมาเพื่อทำงานหนึ่งให้สำเร็จ ซึ่งโปรแกรมย่อยสามารถถูกเรียกใช้งานซ้ำได้หลายครั้ง เช่น กำหนดให้การสร้างบ้านเป็นโปรแกรมหลัก มีการทาสีเป็นโปรแกรมย่อย ดังนั้นสามารถเรียกใช้การทาสีกับห้องต่างๆ ในบ้านได้ รวมถึงนอกบ้าน โดยที่เราไม่ต้องสนใจว่าโปรแกรมย่อยของการทาสีจะต้องประกอบด้วยขั้นตอนย่อยๆ อะไรบ้าง
- ข้อดีของโปรแกรมย่อย คือ ลดความซ้ำซ้อนในการเขียนโปรแกรม โดยจะเก็บชุดคำสั่งไว้ด้วยกัน เพื่อให้สามารถเรียกใช้ได้ตลอดเวลา ทำให้ไม่ต้องเขียนคำสั่งชุดเดิมซ้ำๆ และยังช่วยให้การแก้ไขทำได้ง่าย

ประเภทของโปรแกรมย่อย

- โปรแกรมย่อยใน VB2005 มี 2 ประเภท โดยแบ่งตามรูปแบบการส่งผลการทำงานกลับ ดังนี้
  1. **Sub** มาจากคำเต็มว่า ซับรูทีน (Sub Routine) เป็นโปรแกรมย่อยที่เมื่อจบการทำงานแล้ว จะไม่มีการส่งผลการทำงานกลับมาให้ผู้เรียกใช้โปรแกรมย่อยทราบ
  2. **Function** ฟังก์ชันเป็นโปรแกรมย่อยที่เมื่อจบการทำงานแล้ว จะต้องส่งผลการทำงานกลับมาให้ผู้เรียกใช้ฟังก์ชันทราบ
- ใน VB2005 เมื่อกล่าวถึงโปรแกรมย่อย จะแทนด้วยคำว่า Procedure ซึ่งหมายถึงทั้ง Sub และ Function

การใช้งานโปรแกรมย่อยชนิด Sub

- มีรูปแบบ ดังนี้

```
Sub ชื่อซับรูทีน (รายการพารามิเตอร์)
   <คำสั่งการทำงาน>
   ...
End Sub
```

- พารามิเตอร์ (parameter) มีไว้เพื่อส่งข้อมูลเข้าไปใช้ประกอบการทำงานภายในซับรูทีน ซึ่งอาจจะมีหรือไม่มีก็ได้

การใช้งานโปรแกรมย่อยชนิด Function

- มีรูปแบบ ดังนี้

```
Function ชื่อฟังก์ชัน (รายการพารามิเตอร์) As ชนิดข้อมูล
   <คำสั่งการทำงาน>
   ...
   ชื่อฟังก์ชัน = ค่าที่คืนกลับ
End Function
```

- ชนิดข้อมูล คือ การระบุชนิดข้อมูลของค่าที่จะคืนกลับให้ผู้ใช้
- ฟังก์ชันต้องมีการคืนค่ากลับให้ผู้ใช้เสมอ โดยปกติจะระบุไว้ที่บรรทัดสุดท้ายก่อน End Function หรือใช้คำสั่ง Return ตามด้วยค่าที่คืนกลับแทนก็ได้

การผ่านค่าตัวแปรในโปรแกรมย่อย

- ในการส่งข้อมูลไปประกอบการทำงานของโปรแกรมย่อยทั้งประเภท Sub และ Function ทำได้โดยใช้ตัวแปร ซึ่งเรียกว่า **พารามิเตอร์** (Parameter) และเรียกค่าของข้อมูลที่ถูกส่งไปยังพารามิเตอร์ว่า **อาร์กิวเมนต์** (Argument)

รูปแบบการส่งค่าอาร์กิวเมนต์

- รูปแบบการส่งอาร์กิวเมนต์เบื้องต้นมี 2 แบบ ดังนี้
  1. **Pass by Value** เป็นการผ่านค่าโดยจะคัดลอกค่าที่จะผ่าน แล้วส่งไปให้โปรแกรมย่อยที่ต้องการ แม้ว่าภายในโปรแกรมย่อยจะมีการเปลี่ยนแปลงค่าที่ผ่านไปให้นั้น ก็จะไม่มีผลใดๆ ต่อค่าที่ผ่านซึ่งอยู่ภายนอกโปรแกรมย่อยนั้น
  2. **Pass by Reference** เป็นการผ่านค่าโดยการบอกตำแหน่งในหน่วยความจำที่เก็บค่าที่จะผ่านนั้นไว้ ดังนั้นถ้าภายในโปรแกรมย่อยมีการเปลี่ยนแปลงค่าที่ผ่านไปให้ย่อมส่งผลโดยตรงต่อค่าที่ผ่าน ซึ่งอยู่ภายนอกโปรแกรมย่อยนั้น

รูปแบบการส่งค่าอาร์กิวเมนต์

- โดยปกติถ้าไม่ระบุรูปแบบการส่งค่าอาร์กิวเมนต์ โปรแกรมจะเลือกรูปแบบ Pass by Value โดย Code Editor จะเติมคำว่า **ByVal** ไว้หน้าพารามิเตอร์แต่ละตัว

ขอบเขตของการทำงานในโปรแกรม

- การที่โปรแกรมมีโปรแกรมย่อย แสดงว่ามีการแบ่งโปรแกรมออกเป็นส่วนต่างๆ ที่อิสระจากการ การที่โปรแกรมย่อยเป็นอิสระทำให้เราสามารถสร้างตัวแปรที่มีชื่อเหมือนกันขึ้นมาใช้งานได้ ซึ่ง VB2005 แยกแยะความแตกต่างของตัวแปรที่สร้างขึ้นได้ ด้วยการใช้ ขอบเขตของโปรแกรม (Scope) ซึ่งมี 4 ระดับ เรียงลำดับจากขอบเขตใหญ่ไปยังขอบเขตเล็กดังนี้
  1. **Global** เป็นขอบเขตที่มองเห็นได้ทั้ง Project
  2. **Module** เป็นขอบเขตระดับของแต่ละไฟล์ใน Project
  3. **Procedure** เป็นขอบเขตระดับโปรแกรมย่อย (Sub และ Function)
  4. **Block** เป็นขอบเขตตัวแปรที่เล็กที่สุด มีผลอยู่ระหว่างบล็อกของคำสั่งหนึ่ง

ขอบเขตของการทำงานในโปรแกรม

- การประกาศตัวแปรสำหรับเขียนโปรแกรมที่ดี ควรประกาศในขอบเขตเท่าที่จำเป็นต้องใช้งาน การประกาศตัวแปรระดับ Global มากๆ ไม่ใช่สิ่งที่ดี เพราะ Application จะเสียเนื้อที่หน่วยความจำถาวรตลอดเวลาที่ทำงานเพื่อเก็บค่าตัวแปรนั้น นอกจากนี้ตัวแปรระดับ Global ยังส่งผลให้การตรวจสอบแก้ไขโปรแกรมทำได้ยากขึ้นด้วย
- ตัวแปรแต่ละตัวมีอายุการใช้งานได้เฉพาะในขอบเขตที่ประกาศไว้เท่านั้น หากมีการใช้งานตัวแปรนอกเหนือขอบเขต Code Editor จะแสดงข้อผิดพลาดให้ทราบ คือ ปรากฏเส้นหยักใต้ชื่อตัวแปรนั้น

การใช้งานตัวแปรแบบ Static

- ตัวแปรแบบ Static คือ การกำหนดให้ตัวแปรนั้นมีอายุตลอดการใช้งานโปรแกรม
- โดยปกติอายุตัวแปรจะสิ้นสุดเมื่อออกจากขอบเขตที่ตัวแปรได้ประกาศไว้ แต่ถ้ากำหนดตัวแปรแบบ Static จะช่วยให้ตัวแปรสามารถเก็บค่าไว้ได้แม้จะสิ้นสุดการทำงานในขอบเขตที่ได้ประกาศไว้ และเมื่อการทำงานถูกสั่งให้มาทำในขอบเขตที่ประกาศไว้ซ้ำอีกครั้ง ก็สามารถเรียกใช้ค่าตัวแปรจากการรันครั้งที่ผ่านๆ มา
- ดังนั้น ตัวแปรแบบ Static จึงเป็นทางเลือกหนึ่งแทนการใช้ตัวแปรแบบ Global

การเรียกใช้โปรแกรมย่อยโดยสลับค่าอาร์กิวเมนต์

- โดยทั่วไปการเรียกใช้โปรแกรมย่อยที่มีอาร์กิวเมนตุจะต้องระบุอาร์กิวเมนต์ตามลำดับที่กำหนดไว้ แต่ใน VB 2005 ไม่จำเป็นต้องเรียกใช้โปรแกรมย่อยแบบดังกล่าว โดยผู้ใช้สามารถสลับตำแหน่งอาร์กิวเมนต์ได้ โดยต้องระบุชื่อตัวแปรพร้อมกับค่าที่ส่งผ่านนั้น เรียกความสามารถนี้ว่า Named Arguments

การเรียกใช้โปรแกรมย่อยโดยสลับค่าอาร์กิวเมนต์

- ตัวอย่างเช่น มีการประกาศ Sub ชื่อ MyData
  Sub MyData(ByVal Name As String, ByVal Age As Integer)
- การเรียกใช้งานโปรแกรมย่อย MyData ทำได้หลายรูปแบบ ดังนี้
  MyData('มาลี รุ่งเรือง', 22)

  MyData(Name := 'มาลี รุ่งเรือง', Age := 22)

  MyData(Age := 22, Name := 'มาลี รุ่งเรือง')

การกำหนดค่าดีฟอลต์ให้กับโปรแกรมย่อย

- โดยปกติการเรียกใช้งาน Sub หรือ Function ต้องส่งค่าอาร์กิวเมนต์ให้ครบตามรายการของพารามิเตอร์ในโปรแกรมย่อยนั้น แต่ในบางครั้งเราอาจมีข้อมูลไม่ครบทุกพารามิเตอร์ ทำให้จำเป็นต้องมีค่าดีฟอลต์ไว้ใช้ในกรณีดังกล่าว สามารถทำได้ดังนี้
  - **ส่วนของการประกาศพารามิเตอร์ของโปรแกรมย่อย** : ให้ใส่คำว่า Optional ไว้หน้าพารามิเตอร์ที่ต้องการให้กำหนดค่าดีฟอลต์ แล้วกำหนดค่าดีฟอลต์ไว้ด้านหลัง
  - **การเรียกใช้งานค่าดีฟอลต์** : ให้ละส่วนที่เป็นดีฟอลต์ ไม่ต้องส่งค่าไปให้โปรแกรมย่อยนั้น

การกำหนดค่าดีฟอลต์ให้กับโปรแกรมย่อย

- ตัวอย่าง การเรียกใช้โปรแกรมย่อยที่ให้ผู้เรียกใช้ระบุพารามิเตอร์ที่แสดงชื่อนามสกุลกับพารามิเตอร์ที่แสดงอาชีพ โดยพารามิเตอร์นี้ถ้าไม่มีการกำหนดไว้เมื่อเรียกใช้โปรแกรมย่อยจะใช้ค่าดีฟอลต์คือ “**ไม่ระบุ**” แทน

```
Sub ShowInfo(ByVal strName As String, Optional ByVal strCarreer As String = "ไม่ระบุ")
    Dim strOut As String = ""
    strOut += "ชื่อนามสกุล : " & strName & vbCrLf
    strOut += "อาชีพ : " & strCarreer
    MessageBox.Show(strOut, "Default Option")
End Sub
```

```
Private Sub btnDefault_Click(ByVal sender As System.Object, ByVal e As System.EventArgs) _
Handles btnDefault.Click
    Call ShowInfo("สมบัติ ดีพร้อม", "ขับรถแท็กซี่")
    Call ShowInfo("สมชาติ กล้าหาญดี", ) 'เรียกใช้งานโดยใช้ค่าดีฟอลต์
    Call ShowInfo("สมศรี", "")          'เรียกใช้งานโดยไม่ใช้ค่าดีฟอลต์
End Sub
```

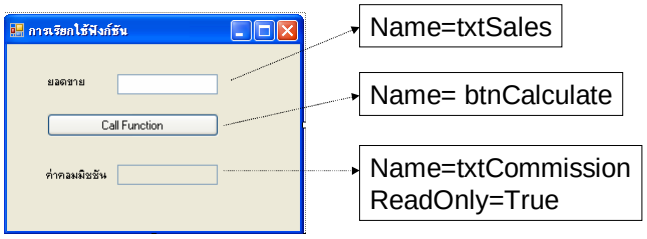
ต.ย.โปรแกรมย่อยแบบ Sub
ซึ่งเป็นโปรแกรมสำหรับคำนวณหาพื้นที่สี่เหลี่ยม

```
'  Sub  procedure  definition.  Sub  procedure  with  two  arguments.
Sub SubComputeArea(ByVal Length As Double, ByVal Width As Double)
    Dim Area As Double      '  Declare  local  variable.
    If Length = 0 Or Width = 0 Then '  If  either  argument  =  0.
        Exit Sub  '  Exit  Sub  immediately.
    End If
    Area = Length * Width       '  Calculate  area  of  rectangle.
    Debug.WriteLine(Area)       '  Print  Area  to  Immediate  window.
End Sub
```

ต.ย.โปรแกรมย่อยแบบ Function
ซึ่งเป็นฟังก์ชันสำหรับหาค่าคอมมิชชัน (commission)

```
Private Function Commission(ByVal saleAmount As Decimal) As Decimal
    ' Calculate  the  sale  commission
    If saleAmount < 100 Then
        Return 0
    ElseIf saleAmount <= 200 Then
        Return 0.15 * saleAmount
    Else
        Return 0.2 * saleAmount
    End If
End Function
```

การออกแบบหน้าจอ



ต.ย.โปรแกรมย่อยแบบ Function ซึ่งเป็นฟังก์ชันสำหรับหาค่าคอมมิชชัน (commission)

- คำสั่งต่อไปนี้เป็นการเรียกใช้ฟังก์ชัน Commission ซึ่งจะทำงานเมื่อผู้ใช้คลิกที่ปุ่ม btnCalculate

```
Private Sub btnCalculate_Click(ByVal sender As System.Obj
    Dim Sales As Decimal
    Sales = Decimal.Parse(txtSales.Text)
    ' Calling  the  Commission  function
    txtCommission.Text = Commission(Sales).ToString("C")
End Sub
```