CHAPTER 6

การสร้างและใช้งานโปรแกรมย่อย

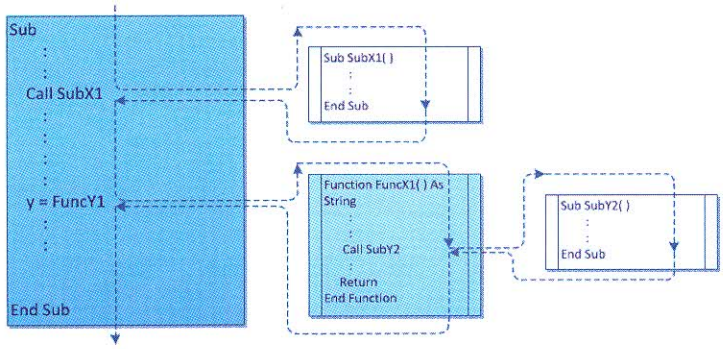
ลองหลับตานึกถึงรถยนต์สักคัน เราจะพบว่ามันเป็นสิ่งที่มีความซับซ้อนมากทั้งเครื่องยนต์กลไก, ระบบไฟฟ้า และระบบต่างๆ มากมาย แต่รถยนต์กลับเป็นสิ่งที่ใช้งานได้ไม่ยาก ขณะที่เราขับรถยนต์ เราคงไม่ไปสนใจว่ารถยนต์คันนี้เริ่มต้นทำงานอย่างไร เวลาจะเบรกกลไกตัวใดต้องทำงานบ้าง เหตุที่เป็นเช่นนี้เพราะรถยนต์คันหนึ่งๆ ถูกแบ่งเป็นระบบต่างๆ มากมาย เช่น ระบบขับเคลื่อน, ระบบปรับอากาศ, ระบบไฟฟ้า เป็นต้น ระบบต่างๆ เหล่านั้นทำงานเป็นอิสระแต่ก็สอดประสานกันดี เราสามารถถอดเปลี่ยนแก้ไขแยกส่วนกันได้ และผู้ใช้งานก็ไม่ต้องสนใจว่าระบบเหล่านั้นทำงานกันอย่างไร

ในการสร้างแอพพลิเคชันที่มีขนาดใหญ่ก็เช่นกัน จำเป็นอย่างยิ่งที่เราต้องมีการแบ่งงานออกเป็นส่วนๆ ซึ่งแต่ละส่วนก็จะมีหน้าที่เฉพาะของตนเอง ซึ่งเราเรียกแต่ละส่วนที่มีหน้าที่เป็นอิสระจากกันนั้นว่า โปรแกรมย่อย ซึ่งบทนี้เราจะได้เรียนรู้การสร้าง และใช้งานฟังก์ชันในทุกรูปแบบด้วย Visual Basic 2010

ประเภทของโปรแกรมย่อย

โปรแกรมย่อย (Sub Program) เป็นส่วนหนึ่งของโปรแกรมหลัก (Main Program) ซึ่งมันถูกแบ่งออกมาให้มีหน้าที่เฉพาะ และแยกการทำงานออกจากกันอย่างเป็นอิสระ

สำหรับคำจำกัดความง่ายของโปรแกรมย่อยก็คือ บล็อกของโค้ดที่สามารถเรียกใช้งานได้จากจุดใดๆ ก็ได้ในแอพพลิเคชัน



การเขียนโปรแกรมที่มีโปรแกรมย่อยอยู่ภายใน จะช่วยโค้ดโปรแกรมนั้นดูเรียบง่ายขึ้น และยังช่วยให้การแก้ไขปรับปรุงทำได้ง่ายขึ้น (คือเลือกแก้ไขปรับปรุงเฉพาะโปรแกรมย่อยที่ต้องการ โดยไม่กระทบกับโปรแกรมย่ออื่นๆ)

นอกจากโปรแกรมย่อยจะช่วยให้เราแบ่งงานออกเป็นงานย่อยๆ แล้ว ข้อดีของโปรแกรมย่อยอีกอย่างก็คือ Reuse ซึ่งหมายถึง โปรแกรมย่อยสามารถถูกเรียกใช้ซ้ำๆ กันได้ไม่จำกัดจำนวน ทำให้ไม่ต้องเขียนโค้ดยาวๆ และเอาเวลาไปคิดงานส่วนอื่นๆ ได้โดยไม่ต้องกังวล เพราะเคยมีฟังก์ชันไว้ให้ใช้งานแล้ว

สำหรับเนื้อหาในบทที่ผ่านมานั้น เราอาจจะเคยเรียกใช้ความสามารถของออบเจ็กต์ และคอนโทรลชนิดต่างๆ ในรูปของการเรียกใช้เมธอด (Method) ซึ่งก็มีความหมายคล้ายกับโปรแกรมย่อย โดยเมธอดที่เราเรียกใช้งานนั้นเราแทบไม่ต้องไปสนใจเลยว่าข้างในมันคืออะไร

ในเบื้องต้นของ Visual Basic 2010 เราแบ่งโปรแกรมย่อยได้ 2 ประเภท ตามสิ่งที่ได้รับหลังจากการทำงานของโปรแกรมย่อยเสร็จสิ้นคือ

- **Sub** มาจากคำเต็มว่า ซับรูทีน (Sub Routine) เป็นโปรแกรมย่อยที่เมื่อทำงานเสร็จแล้ว จะไม่มีการส่งผลการทำงานกลับไปยังโปรแกรมที่เรียกซับรูทีนนี้ใช้งาน
- **Function** ฟังก์ชันเป็นโปรแกรมย่อยเมื่อทำงานเสร็จแล้ว จะต้องมีการส่งผลการทำงานกลับไปยังโปรแกรมที่เรียกฟังก์ชันนี้ใช้งาน

และในโปรแกรมย่อยไม่ว่าจะเป็น Sub หรือ Function นั้น จะมีหรือไม่มีการส่งข้อมูลผ่านตัวพารามิเตอร์ก็ได้ ขึ้นอยู่กับว่าโปรแกรมย่อยนั้นระบุให้มีการกำหนดค่าให้พารามิเตอร์หรือไม่

ในมุมมองของผู้เรียกใช้โปรแกรมย่อยจะมองการเรียกโปรแกรมย่อยนั้นเป็น 1 คำสั่ง ฉะนั้นเมื่อโปรแกรมย่อยทำงานสิ้นสุดลง การทำงานจะกลับไปสู่การทำงานคำสั่งถัดไปของผู้เรียกใช้งานโปรแกรมย่อยนั้นๆ

ในบางครั้งโปรแกรมที่ใช้โปรแกรมย่อย ก็คือตัวโปรแกรมย่อยเอง (เรียกว่า Recursive) ซึ่งถือเป็นเรื่องปกติ แต่เราต้องระมัดระวังในการเรียกใช้งาน มิฉะนั้นจะเกิดการเรียกซ้ำตัวเองไม่สิ้นสุด

โปรแกรมย่อยชนิด Sub

Sub เป็นโปรแกรมย่อยที่เมื่อการทำงานของมันเสร็จสิ้นแล้ว จะไม่มีการส่งผลการทำงานกลับไปยังผู้เรียกใช้งานโปรแกรมย่อย

สำหรับการสร้าง Sub นั้นอาจมีพารามิเตอร์ หรือไม่มีพารามิเตอร์ก็ได้ (พารามิเตอร์ก็จะเป็นชื่อตัวแปรที่เก็บข้อมูลที่ถูกนำไปประกอบการทำงานภายในของ Sub นั่นเอง)

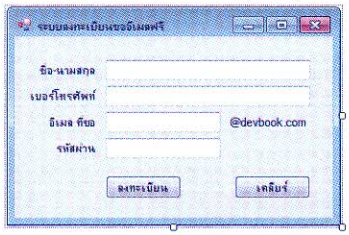
```
Sub ชื่อโปรแกรมย่อย (รายการพารามิเตอร์)
    ชุดคำสั่ง
End Sub
```

Note

ในการเขียนโปรแกรมย่อยขึ้นมาใช้งานนั้น จำเป็นจะต้องประกาศขอบเขตการทำงานของโปรแกรม โดยใช้คำสงวนกำกับไว้หน้าชื่อ Sub เช่น Private, Public เป็นต้น ซึ่งรายละเอียดการใช้คำสงวนนี้ จะกล่าวถึงในบทที่ 10

ตัวอย่างการใช้งาน Sub Routine : ตัวอย่างนี้จะสร้างแอพพลิเคชันเลียนแบบระบบลงทะเบียนอีเมล ซึ่งจะมีการใช้งาน SubRoutine ที่เราเขียนขึ้นมา

1. เข้าสู่การทำงานของ Visual Basic 2010 แล้วเลือกสร้างโปรเจ็กต์แบบ Windows Application โดยออกแบบหน้าตาของแอพพลิเคชัน ดังนี้



2. กำหนดค่าให้กับคอนโทรลต่างๆ ดังตารางต่อไปนี้

ออบเจ็กต์/คอนโทรล	พร็อพเพอร์ตี้	ค่าที่กำหนด
ฟอร์ม	Name	Form1
	Text	ระบบลงทะเบียนขออีเมลฟรี
TextBox	Name	txtName
TextBox	Name	txtPhone
TextBox	Name	txtEmail
TextBox	Name	txtPassword
Button	Name	btnRegister
	Text	ลงทะเบียน
Button	Name	btnClear
	Text	เคลียร์

3. ดับเบิลคลิกที่ฟอร์มเพื่อเขียนโค้ดใน Event Load โดยจะเริ่มแนะนำให้ผู้ใช้งานทราบวิธีการทำงาน โดยจะเรียกใช้งาน Sub ที่ชื่อว่า InformUser

```
Form1.vb* × Form1.vb [Design]*
(Form1 Events)                                  Load
Public Class Form1

    Private Sub Form1_Load(ByVal sender As System.Object, ByVal e As System.
        'แนะนำการใช้งาน โดยเรียกซับรูทีน InformUser
        InformUser()
    End Sub
End Class
100 %
```

หากเรายังเขียน Sub ไม่สมบูรณ์ Code Editor จะฟ้องข้อผิดพลาดให้ทราบ

4. ในหน้าต่าง Code Editor ให้เขียนโค้ดของ Sub InformUser โดยเขียนต่อท้ายไปได้เลย

```
Form1.vb* × Form1.vb [Design]*
Form1                                           InformUser
Public Class Form1

    Private Sub Form1_Load(ByVal sender As System.Object, ByVal e As System.
        'แนะนำการใช้งาน โดยเรียกซับรูทีน InformUser
        InformUser()
    End Sub

    Sub InformUser()
        Dim strOut As String = ""

        strOut &= "ขอแนะนำท่านเข้าสู่ระบบลงทะเบียนอีเมลฟรี" & vbCrLf
        strOut &= "กรุณากรอกชื่อ, เบอร์โทรศัพท์ ให้ครบถ้วน" & vbCrLf
        strOut &= "ชื่ออีเมลเน็ต เลือกได้ตามชอบใจ, ส่วนรหัสผ่านเน็ต ขอแนะนำดังนี้" & vbCrLf
        strOut &= "1) ตั้งให้ยาวกว่า 4 ตัวอักษร" & vbCrLf
        strOut &= "2) ไม่ควรเป็นรหัสที่เดาได้ง่ายเกินไป"
        MessageBox.Show(strOut, "ระบบ Regiter free e-mail", _
                        MessageBoxButtons.OK, _
                        MessageBoxIcon.Information)
    End Sub
End Class
100 %
```

5. ดับเบิลคลิกปุ่ม ลงทะเบียน แล้วเขียนโค้ดเพื่อตรวจสอบความครบถ้วนถูกต้องของข้อมูลที่ป้อนเข้ามา (โดยเรียกใช้งานผ่าน Sub CheckPassword) ถ้าเรียบร้อยดีก็ถือว่าลงทะเบียนได้ แต่ถ้าไม่เรียบร้อยก็แจ้งให้ผู้ใช้ทราบ

```
Private Sub btnRegister_Click(ByVal sender As System.Object, ByVal e As System.EventArgs) Handles btn-
Register.Click
    'ตรวจสอบว่าผู้ใช้งานกรอกข้อมูลครบหรือไม่
    If txtName.Text = "" Or txtPhone.Text = "" Or txtPassword.Text = "" Then
        InformUser()    'แนะนำผู้ใช้งานกรอกให้ครบ
        Exit Sub
    End If
    'กรณีที่กรอกครบ เราจะตรวจสอบรหัสผ่านที่กำหนดให้ว่า ใช้งานได้หรือไม่
    '  เราจะสร้างซับรูทีน CheckPassword ขึ้นมาตรวจสอบ
    CheckPassword(txtName.Text, txtPhone.Text, txtPassword.Text)
End Sub
```

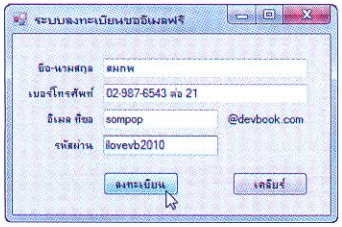
6. สำหรับ Sub CheckPassword นั้นมีหลักการตรวจสอบความครบถ้วนถูกต้องอยู่ 3 ข้อ ดังรายละเอียดที่แสดงในโค้ดต่อไปนี้

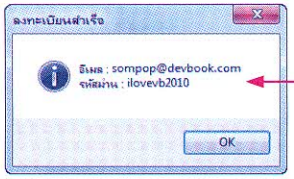
```
Sub CheckPassword(ByVal Name As String, ByVal Phone As String, ByVal Password As String)
    ' ในการตรวจสอบความเหมาะสมของรหัสผ่าน เรามีเกณฑ์ 3 ข้อ
    '1) ตรวจสอบว่ารหัสผ่านมีความยาวต่ำกว่า 4 ตัวอักษรหรือไม่
    If Password.Length < 4 Then
        MessageBox.Show("คุณกำหนดรหัสผ่านสั้นเกินไป (4 ตัวอักษรขึ้นไป)", "ผิดพลาด", MessageBoxButtons.OK,
MessageBoxIcon.Error)
        Exit Sub
    End If

    '2) ตรวจสอบว่ารหัสผ่านตรงกับชื่อ หรือตรงกับเบอร์โทรศัพท์หรือไม่
    If Password = Name Or Password = Phone Then
        MessageBox.Show("คุณกำหนดรหัสผ่านตรงกับชื่อ หรือเบอร์โทรศัพท์", "ผิดพลาด", MessageBoxButtons.OK,
MessageBoxIcon.Error)
        Exit Sub
    End If

    '3) รหัสผ่านนั้นเดาได้ง่ายหรือไม่
    If Password Like "1234" Or Password Like "abcd" Or Password Like "1111" Then
        MessageBox.Show("รหัสผ่านที่กำหนดนั้น เดาง่ายเกินไป", "ผิดพลาด", MessageBoxButtons.OK, MessageBox
Icon.Error)
        Exit Sub
    End If

    'กรณีผ่านทั้ง 3 หัวข้อ
    Dim strOut As String = ""
    strOut &= "อีเมล : " & txtEmail.Text & "@devbook.com" & vbCrLf
    strOut &= "รหัสผ่าน : " & Password
    MessageBox.Show(strOut, "ลงทะเบียนสำเร็จ", MessageBoxButtons.OK, MessageBoxIcon.Information)
End Sub
```

7. ดับเบิลคลิกปุ่ม [เคลียร์] แล้วเขียนโค้ดเพื่อเคลียร์ค่าข้อมูลใน TextBox ต่างๆ ดังนี้

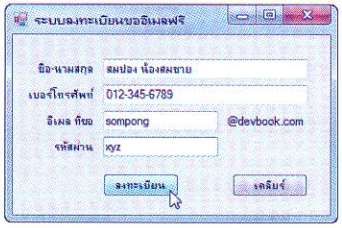
```
Private Sub btnClear_Click(ByVal sender As System.Object, ByVal e As System.EventArgs) Handles btnClear.
Click
    txtName.Clear()
    txtPhone.Clear()
    txtEmail.Clear()
    txtPassword.Clear()
End Sub
```

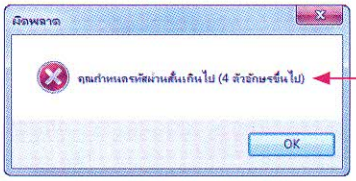
8. ทดสอบการทำงานของแอพพลิเคชันได้ผลดังนี้



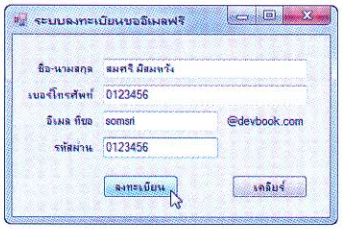


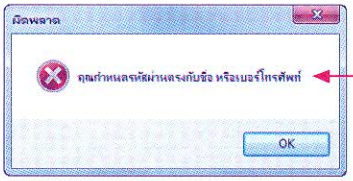
ผลลัพธ์เมื่อทุกอย่างเรียบร้อยดี



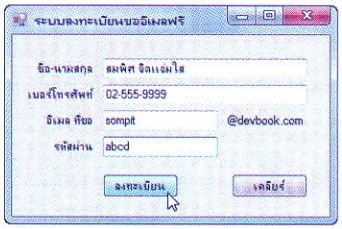


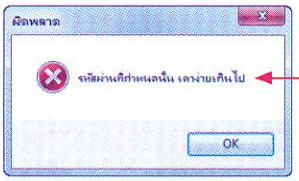
ผลลัพธ์เมื่อกำหนดรหัสผ่านสั้นเกินไป





ผลลัพธ์เมื่อกำหนดรหัสผ่านตรงกับชื่อหรือเบอร์โทรศัพท์





ผลลัพธ์เมื่อกำหนดรหัสผ่านนั้นเดาได้ง่ายเกินไป

โปรแกรมย่อยชนิด Function

Function เป็นโปรแกรมย่อยที่เมื่อการทำงานของมันเสร็จสิ้นแล้ว จะต้องส่งผลการทำงานกลับไปยังผู้เรียกใช้งานโปรแกรมย่อยเสมอ ดังนั้น เมื่อเรียกฟังก์ชันก็แสดงว่าต้องการทราบว่าผลการทำงานเป็นอย่างไร

สำหรับการสร้าง Function นั้นอาจมีพารามิเตอร์ หรือไม่มีพารามิเตอร์ก็ได้ (พารามิเตอร์ก็จะเป็นชื่อตัวแปรที่เก็บข้อมูลที่ถูกนำไปประกอบการทำงานภายในของ Function นั่นเอง)

สำหรับ Function นั้นมีรูปแบบในการสร้าง ดังนี้

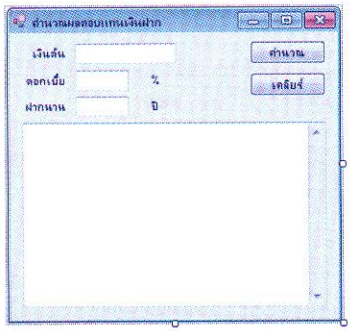
```
Function ชื่อโปรแกรมย่อย (รายการพารามิเตอร์) As ชนิดข้อมูลที่มีการคืนค่า
    ชุดคำสั่ง
    Return ค่าที่คืนกลับ
End Sub
```

จากรูปแบบการใช้งานจะเห็นว่า ก่อนจะจบการทำงานของ Function นั้น เราต้องส่งค่าคืนให้ผู้เรียกใช้งาน Function โดยค่านั้นต้องอยู่หลังคำสั่ง Return

Function หนึ่งๆ สามารถให้ผลการทำงานได้หลากหลายรูปแบบ แต่ค่าที่คืนนั้นก็ต้องมีชนิดข้อมูลสอดคล้องกับชนิดที่ระบุไว้ตอนประกาศชื่อ Function

ตัวอย่างการใช้งาน Function : ตัวอย่างนี้จะสร้างแอพพลิเคชันที่คำนวณผลตอบแทนจากการฝากเงิน ซึ่งผู้ใช้จะต้องกรอกเงินต้น, อัตราดอกเบี้ย และจำนวนปีที่ฝาก

1. เข้าสู่การทำงานของ Visual Basic 2010 แล้วเลือกสร้างโปรเจ็กต์แบบ Windows Application โดยออกแบบหน้าตาของแอพพลิเคชัน ดังนี้



2. กำหนดค่าให้กับคอนโทรลต่างๆ ดังตารางต่อไปนี้

ออบเจ็กต์/คอนโทรล	พร็อพเพอร์ตี้	ค่าที่กำหนด
ฟอร์ม	Name	Form1
	Text	คำนวณผลตอบแทนเงินฝาก
TextBox	Name	txtPrincipal
TextBox	Name	txtInterest
TextBox	Name	txtYear
TextBox	Name	txtOut
	Multiline	True
	ScrollBars	Vertical

ออบเจ็กต์/คอนโทรล	พร็อพเพอร์ตี้	ค่าที่กำหนด
Button	Name	btnCalc
	Text	คำนวณ
Button	Name	btnClear
	Text	เคลียร์

3. ดับเบิลคลิกปุ่ม [คำนวณ] เพื่อคำนวณผลตอบแทนซึ่งจะมีการเรียกใช้งานฟังก์ชัน CheckInput เพื่อตรวจสอบว่าข้อมูลที่กรอกเข้ามาเหมาะสมหรือไม่ ถ้าเหมาะสมดีก็จะนำมาคำนวณโดยเรียกใช้ฟังก์ชัน CalcSaving

```
    Private Sub btnCalc_Click(ByVal sender As System.Object, ByVal e As System.EventArgs) Handles btnCalc.
Click
        'เรียกใช้ฟังก์ชัน CheckInput เพื่อตรวจสอบความเรียบร้อยของข้อมูลก่อนนำมาคำนวณ
        If CheckInput(txtPrincipal.Text, txtInterest.Text, txtYear.Text) Then
            Dim principal As Single = Val(txtPrincipal.Text)
            Dim interest As Single = Val(txtInterest.Text)
            Dim year As Single = Val(txtYear.Text)

            'เรียกใช้งานฟังก์ชัน CalcSaving เพื่อคำนวณผลตอบแทนจากการฝากเงิน
            txtOut.Text = CalcSaving(principal, interest, year)
        Else
            MessageBox.Show("กรุณากรอกข้อมูลเป็นตัวเลข ให้ครบถ้วน และถูกต้อง", "ผิดพลาด", MessageBoxBut-
tons.OK, MessageBoxIcon.Error)
        End If
    End Sub
```

4. สำหรับฟังก์ชัน CheckInput จะทำหน้าที่ตรวจสอบว่าข้อมูลที่ผู้ใช้งานกรอกมาเหมาะสมหรือไม่ โดยข้อมูลทุกตัว (ทั้งเงินต้น, ดอกเบี้ย และจำนวนปีที่ฝากเงิน) ต้องเป็นตัวเลขที่มากกว่า 0 และถ้าเป็นดอกเบี้ยให้มีค่าระหว่าง 0 ถึง 100 ส่วนจำนวนปีที่ฝากก็ต้องฝากตั้งแต่ 1 ปีขึ้นไป ซึ่งถ้าตรวจสอบแล้วไม่เป็นไปตามเงื่อนไขให้ฟังก์ชันนี้รีเทิร์นค่า False กลับไปให้

```
    Function CheckInput(ByVal prin As String, ByVal intr As String, ByVal yr As String) As Boolean
        'ตรวจสอบว่าข้อมูลเงินต้น : ต้องเป็นตัวเลข และมากกว่า 0 บาท
        If IsNumeric(prin) And (Val(prin) > 0) Then
            Return True
        End If

        'ตรวจสอบอัตราดอกเบี้ย : ต้องมากกว่า 0% และไม่เกิน 100%
        If IsNumeric(intr) And (Val(intr) > 0) And (Val(intr) < 100) Then
            Return True
        End If

        'ตรวจสอบว่าจำนวปีที่ฝาก : ตั้งแต่ 1 ปีขึ้นไป
        If IsNumeric(yr) And (Val(yr) >= 1) Then
```

```
        Return True
    End If

    Return False
End Function
```

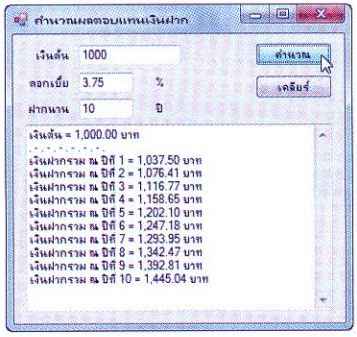
5. สำหรับฟังก์ชัน CalcSaving จะทำหน้าที่คำนวณผลตอบแทนเงินฝาก โดยจะคำนวณให้เห็นว่าในระยะเวลาตั้งแต่ปีที่ 1 ถึงปีที่กำหนดไว้นั้นยอดเงินฝากเป็นเท่าใด (เราจะใช้การคำนวณดอกเบี้ยทบต้น)

```
Function CalcSaving(ByVal sngPri As Single, ByVal sngInt As Single, ByVal intYr As Integer) As String
    Dim i As Integer
    Dim strOut As String = ""

    sngInt *= 0.01
    strOut &= "เงินต้น = " & sngPri.ToString("#,##0.00") & " บาท" & vbCrLf
    strOut &= ". - . - . - . - . - . - ." & vbCrLf

    For i = 1 To intYr
        sngPri = sngPri * (1 + sngInt)
        strOut &= "เงินฝากรวม ณ ปีที่ " & i.ToString & " = " & sngPri.ToString("#,##0.00") & " บาท" &
vbCrLf
    Next
    Return strOut
End Function
```

6. ดับเบิลคลิกปุ่ม [เคลียร์] แล้วเขียนโค้ดเพื่อเคลียร์ค่าต่างๆ ของ TextBox ดังนี้

```
Private Sub btnClear_Click(ByVal sender As System.Object, ByVal e As System.EventArgs) Handles btnClear.
Click
    txtPrincipal.Clear()
    txtInterest.Clear()
    txtYear.Clear()
    txtOut.Clear()
End Sub
```

7. ทดสอบการทำงานของแอพพลิเคชันได้ผลดังนี้



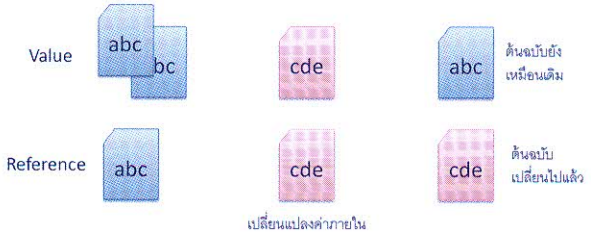
การส่งค่าตัวแปรให้โปรแกรมย่อย

จากหัวข้อที่ผ่านมาเราอาจจะต้องส่งข้อมูลบางอย่างไปประกอบในการทำงานของโปรแกรมย่อย โดยเราจะเรียกชื่อของตัวแปรที่ส่งไปทำงานว่า พารามิเตอร์ (Parameter) และเรียกค่าของข้อมูลที่ถูกส่งไปในพารามิเตอร์ว่า อาร์กิวเมนต์ (Argument)

ใน Visual Basic นั้นมีรูปแบบการส่งอาร์กิวเมนต์เบื้องต้น 2 รูปแบบ ได้แก่

- **Pass by Value** เป็นการผ่านค่าโดยจะก็อปปี้ค่าที่จะผ่าน แล้วส่งไปให้โปรแกรมย่อยที่ต้องการแม้ว่าภายในโปรแกรมย่อยจะมีการเปลี่ยนแปลงค่าที่ผ่านไปให้นั้นก็จะไม่มีผลใดๆ ต่อค่าที่ผ่านซึ่งอยู่ภายนอกโปรแกรมย่อยนั้น โดยกำหนดรูปแบบการรับค่าที่โปรแกรมย่อยเป็น ByVal
- **Pass by Reference** เป็นการผ่านค่าโดยการบอกตำแหน่งในหน่วยความจำที่เก็บค่าที่จะผ่านนั้นไว้ เพราะฉะนั้นถ้าภายในโปรแกรมย่อยมีการเปลี่ยนแปลงค่าที่ผ่านไปให้ย่อมส่งผลโดยตรงของค่าที่ผ่านซึ่งอยู่ภายนอกโปรแกรมย่อยนั้น โดยกำหนดรูปแบบการรับค่าที่โปรแกรมย่อยเป็น ByRef

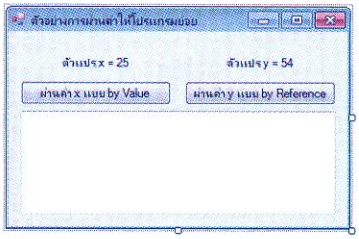
หากจะอธิบายง่ายๆ ว่าการผ่านค่าทั้งสองแบบนี้แตกต่างกันอย่างไร ให้นึกถึงการส่งเอกสารหลักฐานทางราชการ ถ้าตัวจริงก็คือส่งแบบ Pass by Reference แต่ถ้าส่งสำเนาก็คือ Pass by Value นั่นเอง



โดยปกติถ้าเราไม่ได้ระบุชนิดของการส่งผ่านอาร์กิวเมนต์ Visual Basic 2010 จะเลือกรูปแบบ Pass by Value

ตัวอย่างการส่งค่าตัวแปรให้โปรแกรมย่อย : เราจะสร้างแอพพลิเคชันที่ส่งค่าตัวแปรให้โปรแกรมย่อยทั้ง 2 แบบ แล้วสังเกตความแตกต่างจากผลลัพธ์ที่เกิดขึ้น

1. เข้าสู่การทำงานของ Visual Basic 2010 แล้วเลือกสร้างโปรเจ็กต์แบบ Windows Application โดยออกแบบหน้าตาของแอพพลิเคชัน ดังนี้



2. กำหนดค่าให้กับคอนโทรลต่างๆ ดังตารางต่อไปนี้

ออบเจ็กต์/คอนโทรล	พร็อพเพอร์ตี้	ค่าที่กำหนด
ฟอร์ม	Name	Form1
	Text	ตัวอย่างการผ่านค่าให้โปรแกรมย่อย
Button	Name	btnByVal
	Text	ผ่านค่าตัวแปร x แบบ by Value
Button	Name	btnByRef
	Text	ผ่านค่าตัวแปร y แบบ by Reference
TextBox	Name	txtOut
	Multiline	True

3. ดับเบิลคลิกปุ่ม [ผ่านค่า x แบบ by Value] แล้วเขียนโค้ดให้มีการแสดงค่าตัวแปรทั้งก่อน และหลังการส่งค่าตัวแปรแบบ Pass by Value

```
Private Sub btnByVal_Click(ByVal sender As System.Object, ByVal e As System.EventArgs) Handles
btnByVal.Click
    txtOut.Text = ""
    Dim x As Integer = 25
    txtOut.Text &= "ส่งค่าตัวแปรแบบ Pass by Value" & vbCrLf
    txtOut.Text &= "ก่อนส่งตัวแปร แล้ว x = " & x.ToString & vbCrLf
    PassByValue(x)
    txtOut.Text &= "หลังจากส่งตัวแปร แล้ว x = " & x.ToString
End Sub
```

4. สำหรับโปรแกรมย่อยที่มีการส่งค่าตัวแปรแบบ Pass by Value จะมีการทำงาน โดยเปลี่ยนแปลงค่าที่ส่งเข้าไป (บวกเพิ่มอีก 10)

```
Sub PassByValue(ByVal varX As Integer)
    varX += 10
    txtOut.Text &= "- - - - - - - - - -" & vbCrLf
    txtOut.Text &= "  ในการทำงานของโปรแกรมย่อย x = " & varX.ToString & vbCrLf
    txtOut.Text &= "- - - - - - - - - -" & vbCrLf
End Sub
```

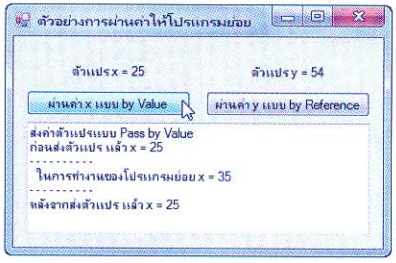
5. ดับเบิลคลิกปุ่ม ผ่านค่า y แบบ by Reference แล้วเขียนโค้ดให้มีการแสดงค่าตัวแปรทั้งก่อน และหลังการส่งค่าตัวแปรแบบ Pass by Reference

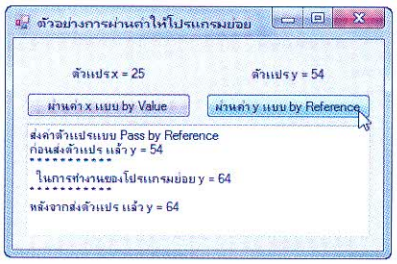
```
Private Sub btnByRef_Click(ByVal sender As System.Object, ByVal e As System.EventArgs) Handles btnBy-
Ref.Click
    txtOut.Text = ""
    Dim y As Integer = 54
    txtOut.Text &= "ส่งค่าตัวแปรแบบ Pass by Reference" & vbCrLf
    txtOut.Text &= "ก่อนส่งตัวแปร แล้ว y = " & y.ToString & vbCrLf
    PassByReference(y)
    txtOut.Text &= "หลังจากส่งตัวแปร แล้ว y = " & y.ToString
End Sub
```

6. สำหรับโปรแกรมย่อยที่มีการส่งค่าตัวแปรแบบ Pass by Referene จะมีการทำงาน โดยเปลี่ยนแปลงค่าที่ส่งเข้าไป (บวกเพิ่มอีก 10)

```
Sub PassByReference(ByRef varY As Integer)
    varY += 10
    txtOut.Text &= "* * * * * * * * * * *" & vbCrLf
    txtOut.Text &= "  ในการทำงานของโปรแกรมย่อย y = " & varY.ToString & vbCrLf
    txtOut.Text &= "* * * * * * * * * * *" & vbCrLf
End Sub
```

7. ทดสอบการทำงานของแอพพลิเคชันได้ผลดังนี้





Tips

สำหรับการส่งค่าตัวแปรแบบ Pass by Reference นั้นมักจะถูกนำมาใช้ก็ต่อเมื่อเราต้องการให้ผลการเปลี่ยนแปลงเกิดกับตัวแปรนั้นทันที หรืออีกกรณีคือ การใช้งานกับฟังก์ชันชนิดที่ต้องการรีเทิร์นค่ามากกว่า 1 ค่า

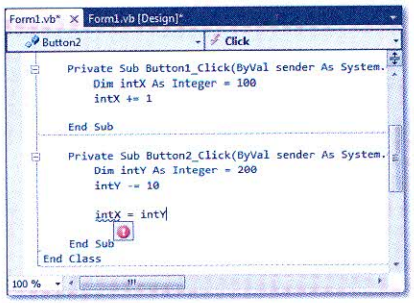
การรีเทิร์นค่ามากกว่า 1 ค่านั้นมักจะเป็นการรีเทิร์นในรูปของ อาร์เรย์ หรือ Structure (เราจะได้เรียนรู้ในบทที่ 7) โดยเราต้องส่งค่าตัวแปรแบบ Reference ในตัวแปรอาร์เรย์ หรือ Structure แล้วก็รีเทิร์นค่ากลับมาผ่านตัวแปรดังกล่าวนั้นกลับมา

ขอบเขตของตัวแปรและโปรแกรมย่อย

ในการเขียนโปรแกรมด้วยภาษา Visual Basic นั้นตัวแปรที่ถูกประกาศนั้นจะมีขอบเขตการใช้งานอยู่เฉพาะในขอบเขต (Scope) ของโค้ดที่มันประกาศ

คำว่าขอบเขตของโค้ดนั้นชัดเจนที่สุดก็คือ การที่ถูกครอบไว้ด้วยชื่อคำสั่ง และ End เช่น บล็อกคำสั่งของ Sub หรือ Function

หากเรามีการอ้างอิงถึงตัวแปรที่อยู่นอกขอบเขต ก็จะถูก Visual Studio แจ้งข้อผิดพลาดให้ได้รับทราบเพื่อแก้ไข



▲ อ้างอิงตัวแปร intX ซึ่งอยู่นอกขอบเขต

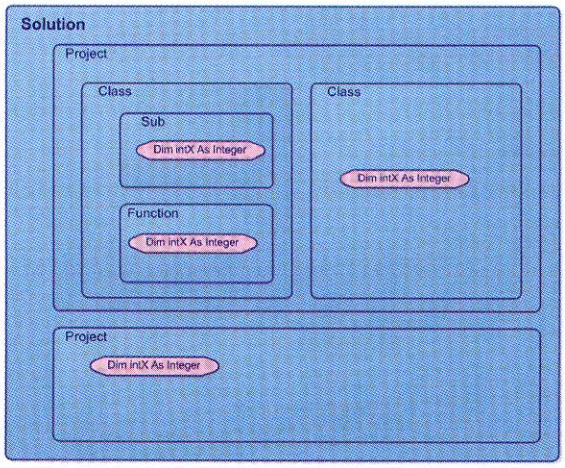
ในแอพพลิเคชันที่ซับซ้อนขึ้นขอบเขตของโค้ดอาจจะอยู่กันคนละไฟล์ เช่น แอพพลิเคชันที่มีหลายๆ ฟอร์ม (เราจะได้ศึกษารายละเอียดในบทที่ 9)

ใน Visual Basic 2010 มีแบ่งขอบเขตของตัวแปรให้เราใช้งานอยู่ 4 ระดับ เรียงลำดับจากขอบเขตของตัวแปรที่ใหญ่ที่สุดไปหาขอบเขตของตัวแปรที่เล็กที่สุด ดังนี้

- **ตัวแปรระดับ Global** มีขอบเขตที่มองเห็นได้ทั้งโปรเจ็กต์ (จะประกาศเป็น Public ใน Module)
- **ตัวแปรระดับ Module** มีขอบเขตที่มองเห็นได้แต่ละไฟล์ในโปรเจ็กต์
- **ตัวแปรระดับ Procedure** มีขอบเขตที่มองเห็นได้ภายในโปรแกรมย่อย (ภายใน Sub และ Function)

ในการประกาศตัวแปรสำหรับการเขียนโปรแกรมนั้น เราควรประกาศตัวแปรในขอบเขตที่เราต้องใช้งาน เพราะการประกาศตัวแปรแบบ Global นั้นจะทำให้สูญเสียเนื้อที่ในหน่วยความจำถาวรไปตลอดเพื่อเก็บตัวแปรนั้น และทำให้ยากในการแก้ไขโปรแกรมในภายหลังด้วย

หลังจากประกาศตัวแปรจนกระทั่งสิ้นสุดการทำงานคำสั่งสุดท้ายในบล็อกของโค้ด ตัวคอมไพเลอร์จะคืนทรัพยากรที่เกิดจากการประกาศตัวแปรในขอบเขตนั้นไป (ในอดีตเราต้องมาใส่ใจตรงนี้เพราะเป็นสาเหตุสำคัญให้แอพพลิเคชันอุ้ยอ้าย และผิดพลาดจากการจัดการการคืนทรัพยากรที่ไม่ดี)

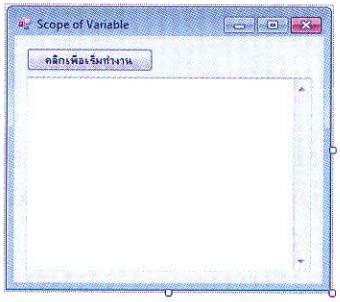


จากรูปแม้จะประกาศตัวแปรชื่อว่า intX เหมือนกัน แต่เมื่ออยู่กันคนละขอบเขตก็ถือว่าเป็นตัวแปรคนละตัวกัน

การประกาศตัวแปรสำหรับการเขียนโปรแกรมที่ดีนั้น ควรจะประกาศ ณ ตำแหน่งที่ต้องการจะใช้ การประกาศไว้ช่วงต้นของบล็อกของโค้ดก็ไม่ใช่เรื่องดีนัก เพราะแอพพลิเคชันจะต้องเสียเนื้อที่หน่วยความจำไปเพื่อเก็บตัวแปรนั้น (ยิ่งเยอะยิ่งเสียพื้นที่มาก)

ตัวอย่างขอบเขตตัวแปร : เราจะสร้างแอพพลิเคชันที่มีการใช้ตัวแปรที่อยู่ในขอบเขตต่างๆ กัน เพื่อชี้ให้เห็นขอบเขตของการประกาศตัวแปรที่ได้อธิบายมาแล้วข้างต้น

1. เข้าสู่การทำงานของ Visual Basic 2010 แล้วเลือกสร้างโปรเจ็กต์แบบ Windows Application โดยออกแบบหน้าตาของแอพพลิเคชัน ดังนี้



2. กำหนดค่าให้กับคอนโทรลต่างๆ ดังตารางต่อไปนี้

ออบเจ็กต์/คอนโทรล	พร็อพเพอร์ตี้	ค่าที่กำหนด
ฟอร์ม	Name	Form1
	Text	Scope of Variable
Button	Name	btnStart
	Text	คลิกเพื่อเริ่มทำงาน
TextBox	Name	txtOut
	Multiline	True
	ScrollBas	Vertical

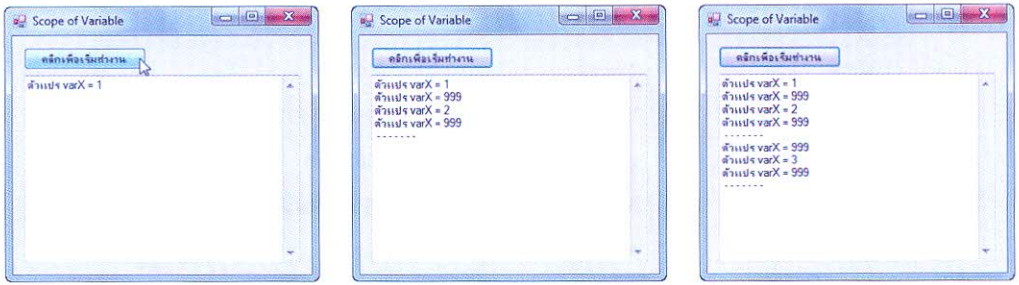
3. ดับเบิลคลิกที่พื้นฟอร์ม แล้วเขียนโค้ดที่มีการประกาศตัวแปร (ระดับ Global) ชื่อว่า varX โดยมีการนำไปกำหนดค่าในอีเวนต์ Form_Load ดังนี้

```
Form1.vb*  Form1.vb [Design]*
Form1                                   (Declarations)
Public Class Form1

    Dim varX As Integer      ' Global Variable

    Private Sub Form1_Load(ByVal sender As System.Object, ByVal e As System.
        varX = 1
        txtOut.Text = "ตัวแปร varX = " & varX.ToString & vbCrLf
    End Sub
End Class
100 %
```

4. ดับเบิลคลิกปุ่ม [คลิกเพื่อเริ่มทำงาน] แล้วเขียนโค้ดให้มีการประกาศตัวแปรชื่อว่า varX อีกเช่นกัน (จะเป็นตัวแปรระดับ Procedure) แล้วมีการกำหนดค่า ขณะเดียวกันเมื่อเรียกใช้ซับรูทีน ChangeVarX ก็จะมีการเปลี่ยนแปลงค่า varX (ซึ่งเป็นตัวระดับ Global)

```
Form1.vb*  Form1.vb [Design]*
btnStart                                Click
Public Class Form1

    Dim varX As Integer      ' Global Variable

    Private Sub Form1_Load(ByVal sender As System.Object, ByVal e As System.
        varX = 1
        txtOut.Text = "ตัวแปร varX = " & varX.ToString & vbCrLf
    End Sub

    Private Sub btnStart_Click(ByVal sender As System.Object, ByVal e As Sys
        Dim varX As Integer = 999   'Local Variable
        txtOut.Text &= "ตัวแปร varX = " & varX.ToString & vbCrLf
        ChangeVarX()
        txtOut.Text &= "ตัวแปร varX = " & varX.ToString & vbCrLf
        txtOut.Text &= " - - - - - - -" & vbCrLf
    End Sub

    Sub ChangeVarX()
        varX += 1          ' เพิ่มค่าให้ตัวแปร Global
        txtOut.Text &= "ตัวแปร varX = " & varX.ToString & vbCrLf
    End Sub
End Class
100 %
```

5. ทดสอบการทำงานของแอพพลิเคชันได้ผลดังนี้



จะเห็นว่าทุกครั้งที่คลิก เราจะมีการแสดงค่าตัวแปร varX ทั้งที่เป็นตัวแปรในระดับ Procedure และระดับ Global ซึ่งตัวแปรทั้งสองมีชื่อเหมือนกันแต่อยู่คนละขอบเขต

แม้ว่าตัวอย่างนี้การทำงานจะถูกต้อง แต่ก็ไม่ใช่ตัวอย่างที่ดีนักในการประกาศค่าตัวแปรที่เหมือนกัน เพราะจะสร้างความสับสนได้ง่าย โดยเฉพาะเมื่อต้องพัฒนาแอพพลิเคชันขนาดใหญ่

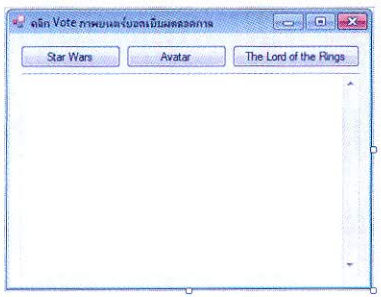
การใช้งานตัวแปรแบบ Static

โดยปกติแล้วอายุของตัวแปรจะสิ้นสุดลงเมื่อจบการทำงานในขอบเขตที่ตัวแปรนั้นได้ประกาศไว้ แต่ถ้าเราต้องการใช้งานตัวแปรนั้นตลอดการใช้งานของแอพพลิเคชัน เราสามารถใช้งานตัวแปรนั้นได้โดยไม่ต้องประกาศเป็นตัวแปรในระดับ Global โดยประกาศตัวแปรแบบ Static

ประกาศตัวแปรแบบ Static จะประกาศตัวแปรโดยใช้คีย์เวิร์ด Static จะทำให้ตัวแปรนี้สามารถเก็บค่าไว้ได้แม้จะสิ้นสุดการทำงานในขอบเขตที่ได้ประกาศไว้ และเมื่อการทำงานของโปรแกรมกลับมาทำในขอบเขตที่ได้ประการตัวแปรนี้อีก ก็จะสามารถเรียกค่าล่าสุดของตัวแปรมาใช้งานได้ต่อไป

ตัวอย่างการใช้งานตัวแปรแบบ Static : ตัวอย่างนี้เราจะสร้างแอพพลิเคชันเลียนแบบการโหวตคะแนนในการประกวด หรือการสำรวจความนิยม โดยเราจะใช้ตัวแปรแบบ Static ทำหน้าที่เก็บคะแนนโหวต

1. เข้าสู่การทำงานของ Visual Basic 2010 แล้วเลือกสร้างโปรเจ็กต์แบบ Windows Application โดยออกแบบหน้าตาของแอพพลิเคชัน ดังนี้



2. กำหนดค่าให้กับคอนโทรลต่างๆ ดังตารางต่อไปนี้

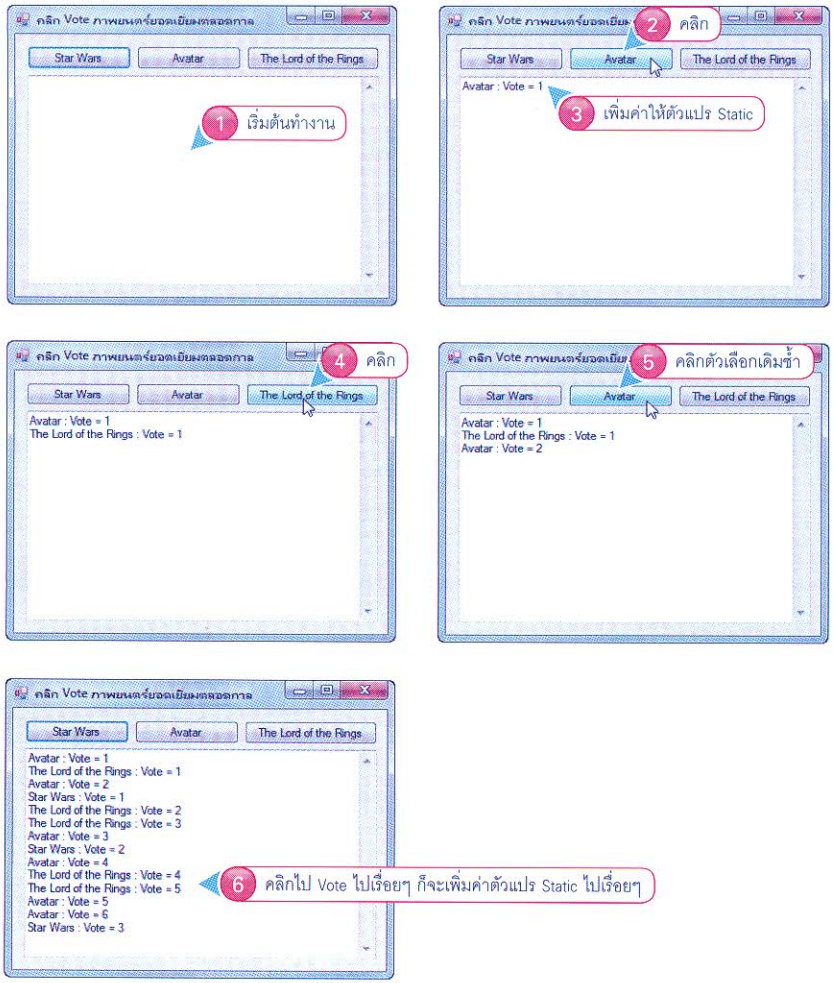
ออบเจ็กต์/คอนโทรล	พร็อพเพอร์ตี้	ค่าที่กำหนด
ฟอร์ม	Name	Form1
	Text	คลิก Vote ภาพยนตร์ยอดเยี่ยมตลอดกาล
Button	Name	btnStarWars
	Text	Star Wars
Button	Name	btnAvartar
	Text	Avartar
Button	Name	btnTheLord
	Text	The Lord of the Rings
TextBox	Name	txtOut
	Multiline	True
	ScrollBas	Vertical

3. ดับเบิลคลิกปุ่มต่างๆ แล้วเขียนโค้ดให้มีการประกาศตัวแปรที่นับการคลิก Vote ซึ่งเป็นตัวแปรแบบ Static ดังนี้

```
    Private Sub btnStarWars_Click(ByVal sender As System.Object, ByVal e As System.EventArgs) Handles
btnStarWars.Click
        Static cnt As Integer
        cnt += 1
        txtOut.Text &= btnStarWars.Text & " : Vote = " & cnt.ToString & vbCrLf
    End Sub

    Private Sub btnAvatar_Click(ByVal sender As System.Object, ByVal e As System.EventArgs) Handles
btnAvatar.Click
        Static cnt As Integer
        cnt += 1
        txtOut.Text &= btnAvatar.Text & " : Vote = " & cnt.ToString & vbCrLf
    End Sub

    Private Sub btnTheLord_Click(ByVal sender As System.Object, ByVal e As System.EventArgs) Handles
btnTheLord.Click
        Static cnt As Integer
        cnt += 1
        txtOut.Text &= btnTheLord.Text & " : Vote = " & cnt.ToString & vbCrLf
    End Sub
```

จะเห็นว่าตัวแปรไม่ว่าจะเป็นแบบธรรมดา หรือ Static หากไม่มีการกำหนดค่าเริ่มต้น Visual Basic จะกำหนดให้ ในกรณีจำนวนเต็มค่าเริ่มต้น = 0

4. ทดสอบการทำงานของแอพพลิเคชันได้ผลดังนี้



Note

หากเราไม่ใช้คีย์เวิร์ด Static กับตัวแปร เราก็ยังมีอีกวิธีก็คือ ประกาศตัวแปรในระดับ Global ซึ่งถ้าตัวแปร Global มีไม่มากนักก็โชคดีไป แต่ถ้ามีจำนวนมากๆ เข้า นอกจากจะสิ้นเปลืองพื้นที่หน่วยความจำแล้ว ยังอาจจะทำให้ผิดพลาดได้ง่าย เช่น อาจมีใครไปแก้ไขค่าตัวแปรนี้เมื่อไรก็ได้ (คิดในกรณีที่เราพัฒนาแอพพลิเคชันกันเป็นทีม)

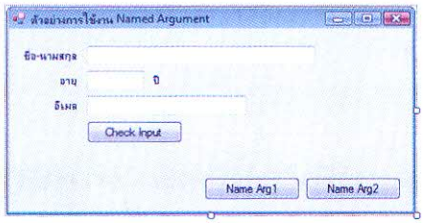
การใช้งานโปรแกรมย่อยโดยสลับลำดับอาร์กิวเมนต์

ใน Visual Basic 2010 มีเทคนิคอีกอย่างหนึ่งที่น่าสนใจคือ การสลับที่ของลำดับอาร์กิวเมนต์ที่จะส่งไปยังโปรแกรมย่อย โดยเราต้องระบุอาร์กิวเมนต์ให้ครบตามจำนวนที่ต้องการด้วย

สำหรับวิธีการสลับลำดับอาร์กิวเมนต์นั้นทำได้โดยต้องระบุชื่อพารามิเตอร์ที่จะรับค่านั้นๆ ด้วย ซึ่งเราเรียกเทคนิคนี้ว่า Named Arguments

ตัวอย่างการใช้งานโปรแกรมย่อยที่สลับอาร์กิวเมนต์ : ตัวอย่างนี้เราจะสร้าง และเรียกใช้งานโปรแกรมย่อยโดยสลับลำดับอาร์กิวเมนต์

1. เข้าสู่การทำงานของ Visual Basic 2010 แล้วเลือกสร้างโปรเจ็กต์แบบ Windows Application โดยออกแบบหน้าตาของแอพพลิเคชัน ดังนี้



2. กำหนดค่าให้กับคอนโทรลต่างๆ ดังตารางต่อไปนี้

ออบเจ็กต์/คอนโทรล	พร็อพเพอร์ตี้	ค่าที่กำหนด
ฟอร์ม	Name	Form1
	Text	ตัวอย่างการใช้งาน Name Argument
TextBox	Name	txtName
TextBox	Name	txtAge
TextBox	Name	txtEmail
Button	Name	btnCheckInput
	Text	Check Input
Button	Name	btnNameArg1
	Text	Name Arg1
Button	Name	btnNameArg2
	Text	Name Arg2

3. ดับเบิลคลิกปุ่ม Check Input แล้วเขียนโค้ดขึ้นมา เพื่อให้ตรวจสอบการกรอกข้อมูลใน TextBox ว่าเหมาะสมหรือไม่ (ชื่อก็ต้องเป็นข้อความห้ามว่างไว้ ส่วนอายุก็ต้องกรอกเป็นตัวเลขระหว่าง 1 ถึง 100 ส่วนอีเมลก็ต้องกรอกห้ามว่างไว้) โดยเราจะเรียกใช้งานฟังก์ชันที่เขียนขึ้นมาคือ CheckInputData ซึ่งจะตรวจสอบให้เรา แล้วรีเทิร์นกลับมาเป็นผลการตรวจสอบดังนี้

```
Private Sub btnCheckInput_Click(ByVal sender As System.Object, ByVal e As System.EventArgs) Handles
btnCheckInput.Click
    Dim strChk As String = CheckInputData(txtName.Text, Val(txtAge.Text), txtEmail.Text)
    MessageBox.Show(strChk, "ผลการตรวจสอบ Input")
End Sub

Function CheckInputData(ByVal Name As String, ByVal Age As Integer, ByVal Email As String) As String
    If Name = "" Or (Age < 0) Or (Age > 100) Or Email = "" Then
        Return "Input ผิดพลาด"
    Else
        Return "Input ถูกต้อง"
    End If
End Function
```

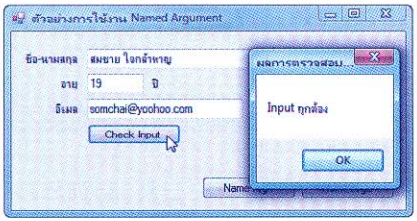
4. สำหรับปุ่มอีก 2 ปุ่มเราจะเขียนโค้ดให้มีการเรียกใช้งานฟังก์ชัน CheckInputData เช่นกัน แต่จะเรียกโดยใช้ความสามารถของ Name Argument ซึ่งสามารถจะสลับอาร์กิวเมนต์ได้ ดังนี้

```
Private Sub btnNameArg1_Click(ByVal sender As System.Object, ByVal e As System.EventArgs) Handles
btnNameArg1.Click
    Dim Name1 As String = txtName.Text
    Dim Age1 As Integer = Val(txtAge.Text)
    Dim Email1 As String = txtEmail.Text

    Dim strChk1 As String = CheckInputData(Age:=Age1, Email:=Email1, Name:=Name1)
    MessageBox.Show(strChk1, "ผลการตรวจสอบ Input")
End Sub

Private Sub btnNameArg2_Click(ByVal sender As System.Object, ByVal e As System.EventArgs) Handles
btnNameArg2.Click
    Dim Name2 As String = txtName.Text
    Dim Age2 As Integer = Val(txtAge.Text)
    Dim Email2 As String = txtEmail.Text

    Dim strChk2 As String = CheckInputData(Email:=Email2, Name:=Name2, Age:=Age2)
    MessageBox.Show(strChk2, "ผลการตรวจสอบ Input")
End Sub
```

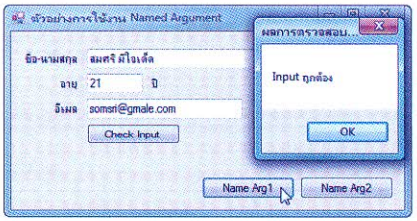
จะสังเกตว่าเวลาที่เราสลับอาร์กิวเมนต์ต้องระบุชื่อตัวแปรให้ชัดเจน แล้วกำหนดค่าให้พารามิเตอร์นั้นโดยใช้เครื่องหมาย := เสมอ ซึ่ง Code Editor ก็จะคอยช่วยให้เราทราบว่ามีพารามิเตอร์ตัวใดบ้าง ขณะเขียนโค้ดแบบ Name Argument

```
btnNameArg1                                   Click
    Private Sub btnNameArg1_Click(ByVal sender As System.Object, ByVal e As System.EventArgs
        Dim Name1 As String = txtName.Text
        Dim Age1 As Integer = Val(txtAge.Text)
        Dim Email1 As String = txtEmail.Text

        Dim strChk1 As String = CheckInputData(Age:= Age1,
    End Sub                    CheckInputData(Name As String, Age As Integer, Email As String) As String
End Class
expression to 'Email:=', type 'Email:= ' <expression>. Otherwise the expression will be   Email:=
e parameter in the current position in the method declaration.                            Name:=
```

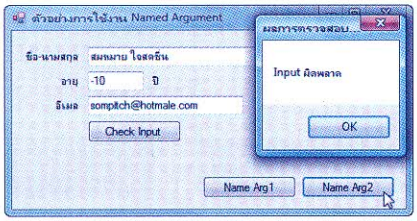
5. ทดสอบการทำงานของแอพพลิเคชัน ได้ผลดังนี้



▲ ส่งค่าตามลำดับ



▲ ส่งค่าสลับตำแหน่ง



▲ ส่งค่าสลับตำแหน่ง

กำหนดส่งผ่านค่าดีฟอลต์ให้กับโปรแกรมย่อย

ที่ผ่านมาไม่ว่าเราจะเรียกใช้โปรแกรมย่อยที่เป็น Sub หรือ Function สิ่งที่เราต้องทำให้ครบถ้วนก็คือการส่งผ่านอาร์กิวเมนต์ให้ครบตามรายการที่โปรแกรมย่อยนั้นประกาศไว้ (อาจมีพิเศษบ้างในเรื่องของการสลับลำดับอาร์กิวเมนต์ แต่อย่างไรก็ต้องส่งให้ครบอยู่ดี)

แต่ในบางครั้งเราอาจจะไม่ต้องผ่านอาร์กิวเมนต์ครบทุกตัวก็ได้ ตัวที่ไม่ได้ผ่านค่าก็ให้ใช้ค่าดีฟอลต์ไปแทน โดยมีหลักการดังนี้

- **ส่วนของการประกาศพารามิเตอร์ของโปรแกรมย่อย** : ให้ใส่คำว่า Optional ข้างหน้าชื่อพารามิเตอร์ตัวที่ต้องการให้มีการกำหนดค่าดีฟอลต์ แล้วกำหนดค่าดีฟอลต์ไว้ด้านหลัง
- **การเรียกใช้งานค่าดีฟอลต์** : ให้ละส่วนที่เป็นดีฟอลต์ไม่ต้องผ่านค่าไปให้โปรแกรมย่อยนั้น

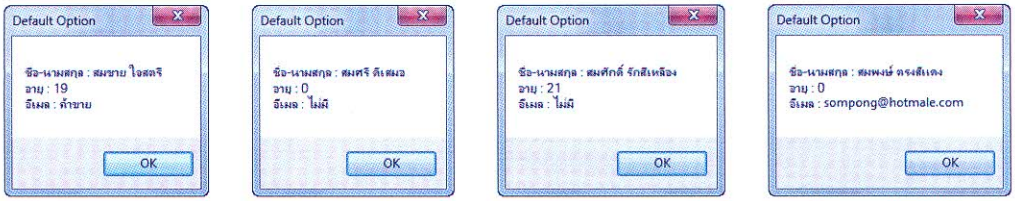
ตัวอย่างการกำหนดค่าดีฟอลต์ให้โปรแกรมย่อย : ตัวอย่างนี้เราจะส่งผ่านค่าดีฟอลต์ให้โปรแกรมย่อย ลองดูโค้ดต่อไปนี้

1. เราจะเรียกใช้โปรแกรมย่อย ShowInfo ซึ่งผู้เรียกใช้จะต้องระบุพารามิเตอร์ที่แสดงชื่อ-นามสกุล กับพารามิเตอร์ที่แสดงอาชีพที่มีการกำหนดค่าดีฟอลต์ว่า "ไม่ระบุอาชีพ"

```
Private Sub btnDefault_Click(ByVal sender As System.Object, ByVal e As System.EventArgs) Handles btnDefault.Cli
    ShowInfo("สมชาย ใจสดชื่", 19, "ค้าขาย")
    ShowInfo("สมศรี ดีเสมอ")
    ShowInfo("สมศักดิ์ รักสิเหลือง", 21)
    ShowInfo("สมพงษ์ ตรงดีเลง", , "sompong@hotmale.com")
End Sub

Sub ShowInfo(ByVal Name As String, Optional ByVal Age As Integer = 0, Optional ByVal Email As String = "ไม่มี")
    Dim strOut As String = ""
    strOut &= "ชื่อ-นามสกุล : " & Name.ToString & vbCrLf
    strOut &= "อายุ : " & Age.ToString & vbCrLf
    strOut &= "อีเมล : " & Email.ToString
    MessageBox.Show(strOut, "Default Option")
End Sub
```

▲ เรียกใช้งานค่าดีฟอลต์

2. ทดสอบการทำงานของแอพพลิเคชันได้ผลดังนี้



ก่อนจบบท

ในบทนี้เราได้เรียนรู้การสร้างโปรแกรมย่อยเพื่อเข้ามาช่วยลดความซับซ้อนของการเขียนโปรแกรมที่เริ่มมีขนาดใหญ่ และซับซ้อน ซึ่งแบ่งได้ 2 ประเภทตามค่าที่คืนกลับผู้เรียกใช้คือ Sub และ Function พร้อมกันนั้นเราได้เรียนรู้วิธีการผ่านค่าไปให้โปรแกรมได้ใช้งานทั้งแบบ Pass by Value และ Pass by Reference ซึ่งจะต้องเข้าใจขอบเขตของการทำงานของตัวแปรไปด้วย ปิดท้ายด้วยการใช้เทคนิคต่างๆ ในการเรียกใช้งานโปรแกรมย่อยทั้งการใช้ตัวแปร Static, การใช้ Name Argument และการกำหนดค่าดีฟอลต์ให้กับข้อมูลที่จะส่งให้โปรแกรมย่อยใช้งาน