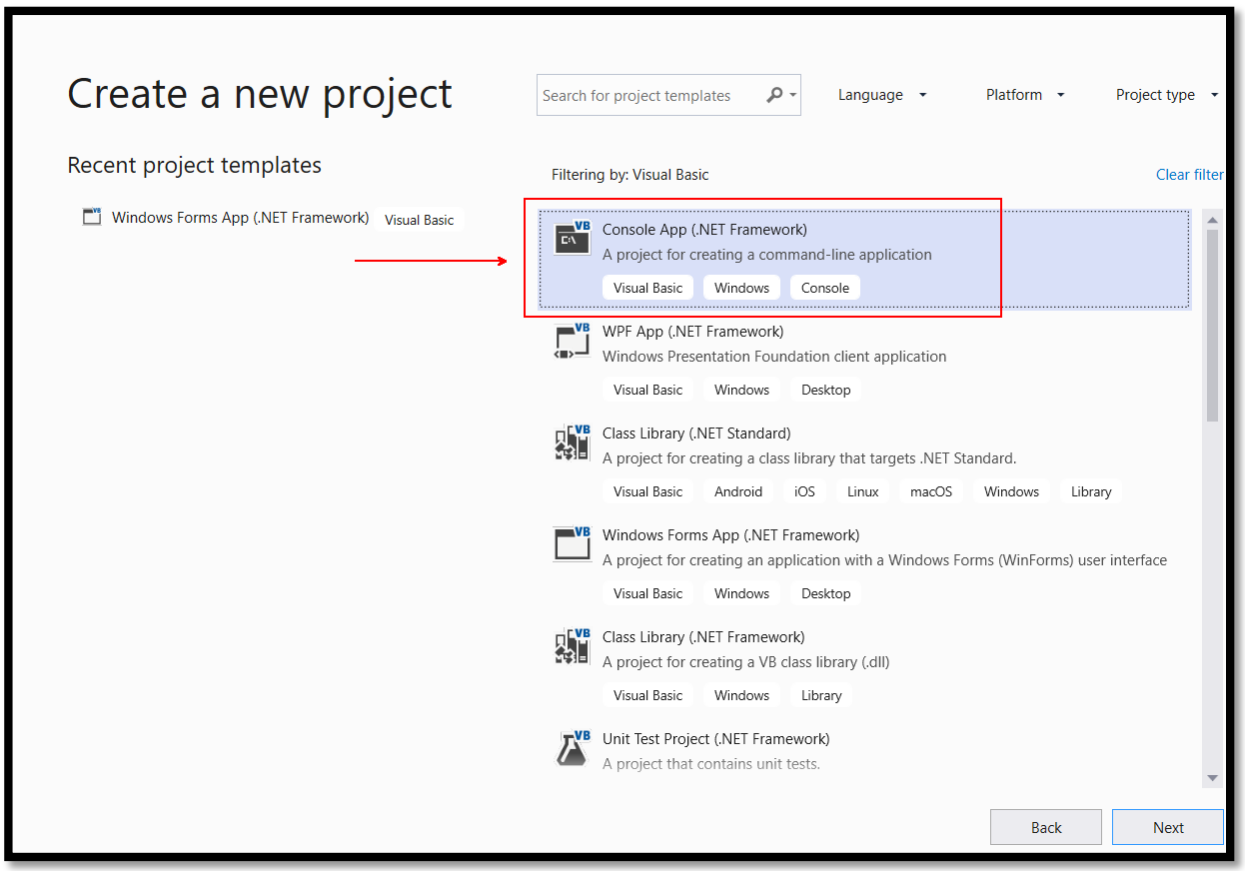
Chapter 3

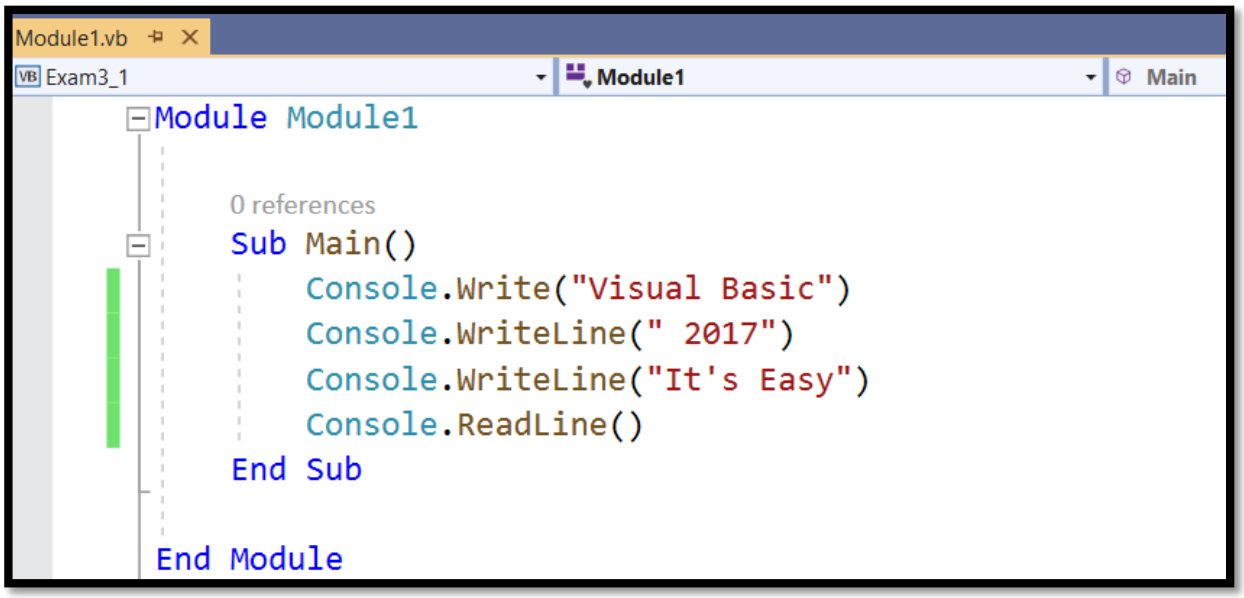
พื้นฐานการเขียนโปรแกรม

3.1 การเขียนโปรแกรมแบบ Console App

การเริ่มต้นเขียนโปรแกรมอาจจะมีโค้ด 4-5 บรรทัดเท่านั้น ดังนั้นเราอาจเขียนโปรแกรมขนาดเล็กแบบเท็กซ์โหมด หรือ DOS Command line จะทำงานโดยการพิมพ์คำสั่งทีละบรรทัด เพื่อให้ง่ายต่อการศึกษาคำสั่งพื้นฐาน หรือเรียกโปรแกรมชนิดนี้ว่า Console App โดยทดลองเขียนโปรแกรมเบื้องต้นดังนี้



ภาพที่ 3.1



ภาพที่ 3.2

เมื่อทดลองเขียนโปรแกรมแล้วสั่งรัน จะได้ผลลัพธ์ดังภาพที่ 3.3

ภาพที่ 3.3

- เมธอด Write จะแสดงข้อความภายในเครื่องหมาย “ ” และแสดงเคอร์เซอร์อยู่ในบรรทัดเดิม
- เมธอด Writeline จะแสดงข้อความภายในเครื่องหมาย “ ” ตรงตำแหน่งที่เคอร์เซอร์กะพริบอยู่ จากนั้นจะเลื่อนเคอร์เซอร์ขึ้นบรรทัดใหม่
- เมธอด Readline สั่งให้โปรแกรมหยุดรอผู้ใช้กดปุ่ม <Enter> ซึ่งเราจะใส่เมธอดนี้ไว้เพื่อเปิดหน้าจอ Console แสดงผลลัพธ์ค้างไว้ หากไม่ใส่ Readline เมื่อรันโปรแกรมเสร็จหน้าจอ Console จะปิดทันที ทำให้ไม่สามารถดูผลลัพธ์ของโปรแกรมได้

3.2 การเขียนคำสั่งในรูปแบบต่างๆ

3.2.1 การแบ่งคำสั่งบรรทัดหนึ่งออกเป็นหลายบรรทัด

เราสามารถแยกบรรทัด ด้วยอักขระช่องว่างแล้วตามด้วยอักขระ _

```
Private Sub Button1_Click(sender As Object, e As EventArgs) Handles Button1.Click
```

```
Private Sub Button1_Click(sender As Object, e As EventArgs) _
    Handles Button1.Click
```

ภาพที่ 3.4

3.2.2 การรวมคำสั่งหลายบรรทัดในบรรทัดเดียว

เราสามารถรวมหลายคำสั่งให้อยู่ในบรรทัดเดียวกัน โดยใช้เครื่องหมาย : เพื่อแยกแต่ละคำสั่งออกจากกัน ตัวอย่างเช่น

```
X = 1
Y = 2
Z = X + Y
```

```
X = 1 : Y = 2 : Z = X + Y
```

ภาพที่ 3.5

3.3 ตัวแปร ค่าคงที่ และชนิดของข้อมูล

3.3.1 ตัวแปร (Variable)

ตัวแปรใน VB มีหน้าที่เก็บข้อมูลในการทำงานของโปรแกรมไว้เป็นการชั่วคราว ตัวแปรที่เรากำหนดขึ้นมาจะต้องประกอบด้วยชื่อตัวแปรและชนิดของข้อมูล (Data Type) ที่ตัวแปรเก็บได้ เราสามารถประกาศตัวแปรได้โดยมีรูปแบบดังนี้

```
Dim <ชื่อตัวแปร> [As Type]
```

```
Dim Name As String
Dim x As Integer
```

ภาพที่ 3.6

การตั้งชื่อตัวแปร มีกฎดังนี้

- จะต้องเริ่มต้นด้วยตัวอักขระ A-Z, a-z หรือ _
- จะต้องประกอบด้วยเฉพาะตัวอักขระ A-Z, a-z, 0-9 หรือ _
- ถ้าเริ่มต้นตัวแรกด้วย _ จะต้องตามด้วยตัวอักขระหรือตัวเลขอย่างน้อยหนึ่งตัว
- มีความยาวไม่เกิน 1023 ตัวอักษร
- ชื่อตัวแปรต้องไม่ซ้ำกันในขอบเขตเดียวกัน
- ชื่อต้องไม่ซ้ำกับคำสงวน (Reserved Word) ของ VB เช่น คำว่า Dim, If, Set ไม่ได้
- ไม่ว่าใช้ตัวอักษรพิมพ์เล็กหรือพิมพ์ใหญ่ให้ความหายเหมือนกัน

ตัวอย่างชื่อตัวแปรที่ถูกต้อง	ตัวอย่างชื่อตัวแปรที่ผิด
Salary_12m	Salary$12
_30	12Salary
_wks	_
	Event

3.3.2 ขอบเขตการประกาศตัวแปร (Scope of variable)

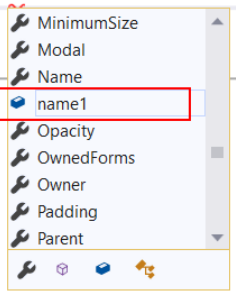
ในการประกาศตัวแปร เราสามารถกำหนดว่าจะให้ตัวแปรนั้นมองเห็นในส่วนใดของโปรแกรมได้บ้าง เช่น ต้องการให้เข้าถึงตัวแปรได้เฉพาะในโปรแกรมย่อยเท่านั้น หรือให้ทุกโปรแกรมย่อยให้คลาสนั้นเห็นทั้งหมด เป็นต้น

เมื่อประกาศตัวแปรโดยใช้คีย์เวิร์ด Public นำหน้าชื่อตัวแปร ความหมายคือ ทุกส่วนของโปรแกรมสามารถมองเห็นและเข้าถึงใช้งานตัวแปรนี้ได้ ส่วนการใช้คีย์เวิร์ด Private นำหน้าชื่อตัวแปร ความหมาย คือ มองเห็นและเข้าถึงตัวแปรได้เฉพาะในโปรแกรมย่อยส่วนนั้น ดังภาพที่ 3.7

```
Public Class Form1
    Public name1 As String
    Private name2 As String
    0 references
    Private Sub Form1_Load(sender As Object, e As EventArgs) Handles MyBase.Load
        Dim name3 As String = "Visual 2019"
        TextBox1.Text = name1
        TextBox2.Text = name2
        TextBox3.Text = name3
    End Sub
End Class
```

Form1

```
Public Class Form2
    Dim name5 As String
    0 references
    Private Sub Form2_Load(sender As Object, e As EventArgs) Handles MyBase.Load
        Dim f As Form1
        Dim name As String
        name = f.
    End Sub
End Class
```



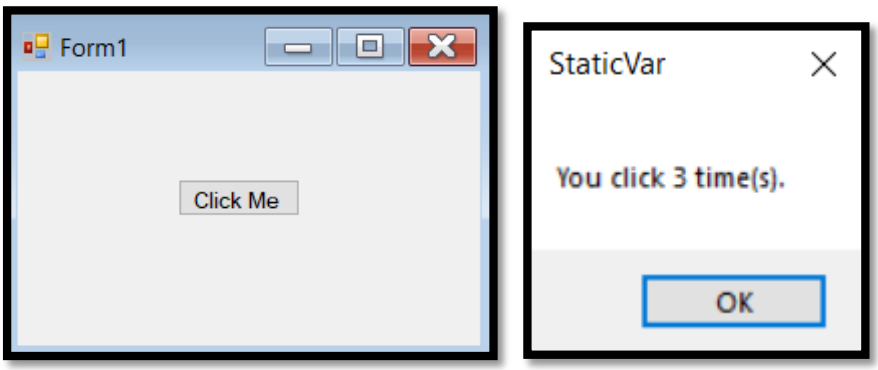
Form2

ภาพที่ 3.7

3.3.3 ตัวแปรแบบสเตติก (Static Variables)

ตัวแปรระดับโปรแกรมย่อย (ตัวแปรที่ถูกประกาศในโปรแกรมย่อย) จะใช้ได้เฉพาะในขณะที่การทำงานอยู่ในโปรแกรมย่อยเท่านั้น ซึ่งหลังจากโปรแกรมย่อยทำงานเสร็จแล้ว ตัวแปรนั้นก็จะถูกทำลาย และเมื่อมีการเรียกใช้งานโปรแกรมย่อยนั้นอีกครั้ง ตัวแปรระดับโปรแกรมย่อยก็จะถูกสร้างขึ้นมาใหม่ ซึ่งอาจทำให้ค่าของตัวแปรไม่ใช่ค่าเก่าอีกต่อไป แต่เราสามารถใช้คำสั่ง Static แทน Dim ในการประกาศตัวแปรเพื่อรักษาค่าล่าสุดของตัวแปร หลังจากเราออกจากโปรแกรมย่อยไว้ได้ โดยเมื่อเราเรียกใช้โปรแกรมย่อยอีกครั้ง ตัวแปรนี้ก็ยังคงเก็บค่าล่าสุดที่กำหนดไว้

ตัวอย่างโปรแกรม การนับจำนวนครั้งของการคลิกเมาส์ว่าปุ่มคำสั่ง จะแสดงกล่องข้อความว่าคลิกไปแล้วกี่ครั้ง




```
Public Class Form1
    0 references
    Private Sub Button1_Click(sender As Object, e As EventArgs) Handles Button1.Click
        Static ClickCount As Integer
        ClickCount = ClickCount + 1
        MsgBox("You click " & ClickCount & " time(s).", MsgBoxStyle.OkOnly)
    End Sub
End Class
```

ภาพที่ 3.8

3.3.4 ค่าคงที่ (Constants)

ค่าคงที่มีไว้สำหรับเก็บค่าที่ไม่เปลี่ยนแปลงตลอดช่วงเวลาที่โปรแกรมรันอยู่ ซึ่งถ้าในโปรแกรมมีการใช้คำสั่งที่เปลี่ยนค่าคงที่ VB ก็จะแสดงข้อผิดพลาดขึ้นมา เพราะเราเปลี่ยนค่าที่ประกาศเป็นค่าคงที่ไม่ได้

การประกาศค่าคงที่มีลักษณะเดียวกับการประกาศตัวแปร โดยใช้คำสั่งที่มีรูปแบบดังนี้

```
[Private | Public] Const <ชื่อค่าคงที่> [As Type] = <ค่าที่กำหนดให้>
```

ในการประกาศค่าคงที่ เราสามารถกำหนดชนิดข้อมูลของค่าคงที่และกำหนดค่าเป็นตัวเลข ข้อความ ค่าในตัวแปร หรือใช้ตัวดำเนินการร่วมในการกำหนดค่าได้

ตัวอย่างการประกาศค่าคงที่

```
Const PI = 3.1415926                          'ประกาศค่าคงที่ PI
Const RPI = 1 / PI                            'ประกาศค่าคงที่ส่วนกลับของ PI
Public Const CVersion As String = "1.0.1"     'ประกาศค่าคงที่ชนิดข้อความ
Private Const OneHundred As Integer = 100     'ประกาศค่าคงที่ชนิดตัวเลข
```

และเราสามารถนำค่าที่ประกาศมาใช้ในโปรแกรมได้ดังนี้

```
AreaCircle = PI * 7 * 7
TextBox2.Text = AreaCircle
```

จากคำสั่งข้างต้น จะเป็นการหาพื้นที่วงกลมจากสูตร PI* (ความยาวรัศมียกกำลังสอง) สำหรับของเขตการประกาศค่าคงที่ จะเหมือนกับของเขตการประกาศตัวแปรดังที่ได้อธิบายไว้แล้ว

3.3.5 ชนิดของข้อมูล (Data Types)

ชนิดของข้อมูลที่เรากำหนดให้กับตัวแปรจะทำให้การเก็บข้อมูลของเรามีประสิทธิภาพ เพราะใช้พื้นที่ขนาดเหมาะสมในการเก็บข้อมูล โดยใน VB จะมีชนิดข้อมูลต่างๆ ดังนี้

ชนิด	ขนาด (ไบต์)	คำอธิบาย
ชนิดข้อมูลแบบตัวเลขจำนวนเต็ม จะใช้เก็บเฉพาะเลขจำนวนเต็มที่ใช้บ่อยในการเขียนโปรแกรม เช่น 101, 0, -30 เป็นต้น		
Byte	1	มีค่าตั้งแต่ 0-255 แต่ละบิตใช้แทนเลข 0 และ 1 ซึ่งเป็นเลขฐานสอง
Char	2	มีค่าตั้งแต่ 0-65535
Short	2	มีค่าตั้งแต่ –32,768 ถึง 36,768
Integer	4	มีค่าตั้งแต่ -2,147,483,648 ถึง 2,147,483,648

ชนิด	ขนาด (ไบต์)	คำอธิบาย
Long	8	มีค่าตั้งแต่ –9,223,372,036,854,775,808 ถึง 9,223,372,036,854,775,808
ชนิดข้อมูลแบบเลขทศนิยม จะใช้เก็บเฉพาะเลขจำนวนทศนิยมสำหรับใช้ในการคำนวณทางคณิตศาสตร์และวิทยาศาสตร์ เช่น 101.34, 0.22, -55.36 เป็นต้น (ตัวอักษร E จะหมายถึงยกกำลังเช่น 1.7E35 เท่ากับ 1.7×10^{35} เป็นต้น		
Single	4	เก็บเลขทศนิยมที่มีความละเอียดต่ำ มีค่าตั้งแต่ ค่าลบ -3.4028235E+38 ถึง -1.401298E-45 ค่าบวก 1.401298E-45 ถึง 3.4028235E+38
Double	8	เก็บเลขทศนิยมความละเอียดสูง มีค่าตั้งแต่ ค่าบวก 4.94065645841246544E-324 ถึง 1.79769313486231570E+308 ค่าลบ -1.79769313486231570E+308 ถึง -4.94065645841246544E-324 และ 0
Decimal	16	เก็บตัวเลขขนาดใหญ่มาก
ชนิดข้อมูลทั่วไป ชนิดข้อมูลทั่วไปนี้จะใช้เก็บข้อความ เก็บค่า True (จริง) หรือ False (เท็จ) อย่างในอย่างหนึ่ง และเก็บค่าวันที่		
Boolean	-	มีค่า True กับ False (ใช้ขนาดพื้นที่หน่วยความจำต่างกันขึ้นกับแพลตฟอร์ม)
String	-	เก็บตัวอักษรได้ 0 ถึง สองพันล้านอักขระ (2^31 และขนาดขึ้นอยู่กับแพลตฟอร์ม)

Date	8	เก็บข้อมูลที่ใช้แทนวันที่และเวลา ตัวอย่างการใช้งาน เช่น
		`Dim MyDate As Date`
		`MyDate = #9/05/2019#`
		`MyDate = #2019/5/9 13:45#`
		`MyDate = #9/05/2019 6:60AM#`

Object	4/8	4 ไบต์บนแพลตฟอร์ม 32 บิตหรือ 8 ไบต์บนแพลตฟอร์ม 64 บิต เก็บข้อมูลที่ใช้ในการอ้างอิงถึงออบเจ็กต์ต่างๆ ถ้าเราประกาศตัวแปรโดยที่ไม่ได้ระบุชนิดข้อมูล ชนิดข้อมูลของตัวแปรตังนั้นก็จะถูกกำหนดเป็น Object ทันที เช่น Dim Test ตัวแปร Test จะเป็นชนิด Object ทันที

3.3.6 ตัวดำเนินการในการคำนวณทางคณิตศาสตร์

การกระทำ	**สัญลักษณ์**
การบวก	+
การลบ	-
การคูณ	*
การหาร	/
การหารแบบจำนวนเต็ม	\
การหารเอาเศษ	Mod
การยกกำลัง	^

3.3.7 ตัวดำเนินการในทางตรรกะ จะให้ผลลัพธ์เป็นค่า True และ False

A	B	A And B
True	True	True
True	False	False
False	True	False
False	False	False

A	B	A OR B
True	True	True
True	False	True
False	True	True
False	False	False

A	B	A XOR B
True	True	False
True	False	True
False	True	True
False	False	False

A	B	A AndAlso B
True	True	True
True	False	False
False	ไม่มีการหาค่า	False

ตัวดำเนินการ **AndAlso** จะคล้ายกับตัวดำเนินการ And ยกเว้นแต่ว่าถ้านิพจน์แรกมีค่าเป็น False ก็จะไม่หาค่านิพจน์ที่สองเลย และจะให้ผลลัพธ์กับมาเป็น False

A	B	A ORElse B
True	ไม่มีการหาค่า	True
True	False	True
False	False	False

ตัวดำเนินการ **ORElse** จะคล้ายกับตัวดำเนินการ Or ยกเว้นแต่ว่านิพจน์แรกมีค่าเป็น True ก็จะไม่หาค่านิพจน์ที่สองเลย และจะให้ผลลัพธ์กลับมาเป็น True

A	Not A
True	False
False	True

3.3.8 ตัวดำเนินการในการทำงานเกี่ยวกับข้อมูลชนิด String

ตัวดำเนินการกลุ่มนี้จะใช้เชื่อม String กับ String เข้าด้วยกัน หรือ String กับข้อมูลตัวเลขก็ได้

- ใช้ **+** เชื่อม String กับ String เช่น

```
Console.WriteLine("Your Friend is " + YF)
```

- ใช้ **&** เชื่อม String กับข้อมูลแบบ Numeric หรือ String ก็ได้ เช่น

```
Console.WriteLine("Area is " & AreaCircle)
```

3.3.9 ตัวดำเนินการเปรียบเทียบ

ตัวดำเนินการประเภทนี้จะใช้เปรียบเทียบระหว่างค่า 2 ค่า โดยมีผลลัพธ์เป็น True หรือ False อย่างใดอย่างหนึ่งเท่านั้น ดังนี้

สัญลักษณ์ของตัวดำเนินการ	ความหมาย
=	เท่ากับ
<>	ไม่เท่ากับ
<	น้อยกว่า
>	มากกว่า
<=	น้อยกว่าหรือเท่ากับ
>=	มากกว่าหรือเท่ากับ

3.3.10 ลำดับในการทำงานของตัวดำเนินการ

3.4 รับข้อมูลจากผู้ใช้ด้วยเมธอด InputBox

เป็นช่องรับข้อมูลบนไดอะล็อกบ็อกซ์ เหมาะสำหรับการรับข้อมูลจากผู้ใช้เพียงค่าเดียว โดยกรอกข้อความหรือตัวเลขลงในช่องรับข้อมูล จากนั้นให้คลิกปุ่ม OK ข้อมูลก็จะถูกรับเข้ามาดำเนินการในโปรแกรม InputBox เป็นเมธอดของคลาส Interaction ที่สามารถอ้างอิงเรียกใช้งานได้ทั้งในโปรเจ็กต์ชนิด Windows Forms App และ Console App แต่ไม่สามารถนำไปใช้กับภาษาอื่นใน .NET ได้ เพราะเป็นเมธอดสมาชิกของเนมสเปซ Microsoft.VisualBasic ที่ไม่ใช่ .NET

<u>รูปแบบ</u>

```
'InputBox(Message,title, defaultValue, XPos,YPos)
a = InputBox("กรุณาห้อนจำนวนชั่วโมงการทำงานนอกเวลา", "คำนวณค่าแรงนอกเวลา", 0, 100, 50)
```

Message สตริงที่ต้องการให้แสดงเป็นข้อความในไดอะล็อกบ็อกซ์ เพื่อแจ้งให้ผู้ใช้ทราบว่าต้องการให้ป้อนข้อมูลอะไร มีความยาวได้ 1,024 ตัวอักษร

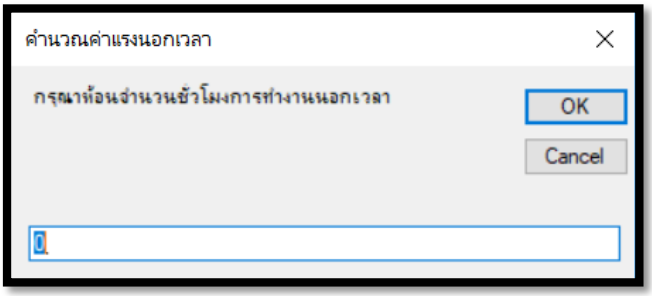
Title สตริงที่ต้องการให้แสดงเป็นข้อความในไตเติลบาร์ของไดอะล็อกซ์บ็อกซ์

defaultValue สตริงที่ต้องการให้แสดงเป็นข้อความในช่องรับข้อมูล หากผู้ใช้ไม่ป้อนข้อมูลใดๆ ลงไป ข้อความที่เป็นค่าเริ่มต้นนี้จะถูกรับเข้ามาในโปรแกรม เพื่อป้องกันไม่ให้โปรแกรมทำงานผิดพลาด

Xpos ตัวเลขระยะห่างระหว่างขอบด้านซ้ายของไดอะล็อกบ็อกซ์จากขอบด้านซ้ายของจอภาพ

Ypos ตัวเลขระยะห่างระหว่างขอบด้านบนของไดอะล็อกบ็อกซ์จากขอบด้านบนของจอภาพ

จากตัวอย่างเมื่อรันโค้ดจะได้ผลลัพธ์ดังภาพ



ภาพที่ 3.9

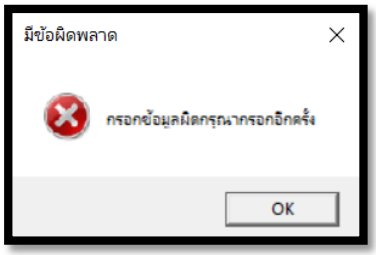
3.5 แสดงกล่องข้อความ MsgBox

เป็นกล่องแสดงข้อความเพื่อแจ้งผลลัพธ์จากการคำนวณ วิเคราะห์ข้อมูล แสดงสถานะการทำงาน หรือการแจ้งเตือนต่างๆ แก่ผู้ใช้ ซึ่งจะรอจนผู้ใช้รับทราบข้อมูลแล้วคลิกปุ่ม OK เพื่อยืนยันและปิด MsgBox เป็นเมธอดของคลาส Interaction ที่สามารถอ้างอิงเรียกใช้งานได้ทั้งในโปรเจ็กต์ชนิด Windows Forms App และ Console App รวมทั้งอยู่ในเนมสเปซ Microsoft.VisualBasic ที่ไม่ใช่ .NET จึงไม่สามารถรองรับการใช้งานกับภาษาอื่นใน .NET ได้

<u>รูปแบบ</u>

```
'MsgBox(Message, Styles, title)
MsgBox("กรอกข้อมูลผิดกรุณากรอกอีกครั้ง", MsgBoxStyle.Critical, "มีข้อผิดพลาด")
```

<u>ผลลัพธ์</u>



- Message สตริงที่ต้องการให้แสดงเป็นข้อความในไดอะล็อกบ็อกซ์ยาวไม่เกิน 1,024 ตัวอักษร
- Style รูปแบบของปุ่มและไอคอนที่ต้องการให้แสดงในไดอะล็อกบ็อกซ์
- Title สตริงที่ต้องการให้แสดงเป็นข้อความในไตเติลบาร์ของไดอะล็อกบ็อกซ์

รูปแบบของ Style

รูปแบบสไตล์	คำอธิบาย
Okonly	แสดงเฉพาะปุ่ม OK
OKCancel	แสดงปุ่ม OK และ Cancel
AbortRetryIgnore	แสดงปุ่ม Abort, Retry และ Ignore
YesNoCancel	แสดงปุ่ม Yes, No และ Cancel
YesNo	แสดงปุ่ม Yes และ No
RetryCancel	แสดงปุ่ม Retry และ Cancel
Critical	แสดงไอคอนข้อความสำคัญ
Question	แสดงไอคอนคำถาม
Exclamation	แสดงไอคอนข้อความแจ้งเตือน
Information	แสดงไอคอนข้อความแจ้งข่าวหรือประชาสัมพันธ์
DefaultButton1	กำหนดปุ่มแรกเป็นดีฟอลต์
DefaultButton2	กำหนดปุ่มที่สองเป็นดีฟอลต์
DefaultButton3	กำหนดปุ่มที่สามเป็นดีฟอลต์
ApplicationModal	กำหนดให้ผู้ใช้ตอบสนองก่อนจะกลับไปทำงานในแอพพลิเคชันต่อไป
SystemModal	หยุดการทำงานของแอพพลิเคชันทั้งหมดจนกว่าผู้ใช้จะตอบสนองต่อกล่องข้อความ
MsgBoxSetForeground	กำหนดให้กล่องข้อความปรากฏขึ้นมาซ้อนอยู่บนหน้าต่างการทำงานอื่นๆ
MsgBoxRight	กำหนดให้ข้อความชิดขวา
MsgBoxRtlReading	แสดงข้อความสำหรับการอ่านจากขวาไปซ้าย เช่น ภาษาฮิบรู และอารบิก

ตัวอย่างการใช้เมธอด InputBox และ MsgBox

เป็นโปรแกรมคำนวณค่าแรงนอกเวลาของคนงาน โดยจะแสดงกล่องรับข้อมูลให้ผู้ใช้กรอกจำนวนชั่วโมงการทำงานนอกเวลา จากนั้นโปรแกรมจะทำการคำนวณค่าแรงด้วยชั่วโมงละ 70 บาท และแสดงค่าบนกล่องข้อความ

1) สร้างโปรเจ็กต์ใหม่ขึ้นมาชนิด Console App

2) ปรากฏหน้าต่างสำหรับเขียนโค้ด ให้เขียนดังนี้

```
Module Module1
    0 references
    Sub Main()
        Dim time, ans As Integer  'ประกาศตัวแปร time รับข้อมูล ans เก็บผลลัพธ์
        time = InputBox("กรุณาป้อนจำนวนชั่วโมงการทำงานนอกเวลา", "คำนวณค่าแรงนอกเวลา", 0, 100, 50)
        'รับข้อมูลจากกล่องรับข้อมูลเก็บในตัวแปร time
        ans = Val(time) * 70     'แปลงข้อมูลที่รับมาเป็นตัวเลขและคูรกับอัตราค่าแรง 70 บาท
        MsgBox("เวลาทำงานนอกเวลา : " & time & " ชั่วโมง ได้ค่าตอบแทน : " & ans,
               MsgBoxStyle.OkOnly, "ค่าจ้างนอกเวลา")

    End Sub
End Module
```

ภาพที่ 3.10

3.6 แสดงกล่องข้อความ MessageBox

เป็นกล่องข้อความในลักษณะเดียวกันกับ MsgBox ต่างกันที่ MessageBox เป็นคลาสใน .NET ซึ่งสามารถรองรับการใช้งานกับภาษาอื่นๆใน .NET ได้ มีเมธอดที่นิยมใช้งานได้แก่

เมธอด	คำอธิบาย
Show(message)	แสดงเฉพาะข้อความ
Show(message, title)	แสดงข้อความและชื่อไตเติล
Show(message, title, button)	แสดงข้อความ ชื่อไตเติล และปุ่มแบบต่างๆ
Show(message, title, button, icon)	แสดงข้อความ ชื่อไตเติล ปุ่มแบบต่างๆ และ ไอคอนต่างๆ

- message สตริงที่ต้องการให้แสดงเป็นข้อความในไดอะล็อกบ็อกซ์
- title สตริงที่ต้องการให้แสดงเป็นข้อความในไตเติลบาร์ของไดอะล็อกบ็อกซ์
- button กำหนดค่าแสดงปุ่มในรูปแบบต่างๆ ในไดอะล็อกบ็อกซ์

รูปแบบปุ่ม	คำอธิบาย
AbortRetryIgnore	แสดงปุ่ม Abort, Retry และ Ignore
OK	แสดงปุ่ม OK
OKCancel	แสดงปุ่ม OK และ Cancel
RetryCancel	แสดงปุ่ม Retry แล Cancel
YesNo	แสดงปุ่ม Yes และ No
YesNoCancel	แสดงปุ่ม Yes, No และ Cancel

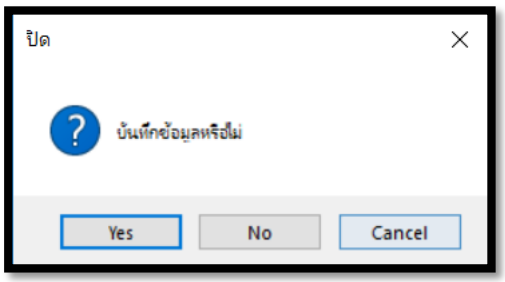
- icon กำหนดค่าแสดงไอคอนในรูปแบบต่างๆ ให้กับไดอะล็อกบ็อกซ์

รูปแบบไอคอน	คำอธิบาย
Asterisk	แสดงไอคอนคำอธิบายประกอบ
Error	แสดงไอคอนแจ้งความผิดพลาด
Exclamation	แสดงไอคอนเครื่องหมายอัศเจรีย์
Hand	แสดงไอคอนการดูแล
Information	แสดงไอคอนแจ้งข้อมูล
None	ไม่มีไอคอนสัญลักษณ์
Question	แสดงไอคอนเครื่องหมายคำถาม
Stop	แสดงไอคอนเครื่องหมายหยุด
Warning	แสดงไอคอนแจ้งเตือนข้อควรระวัง

<u>รูปแบบ</u>

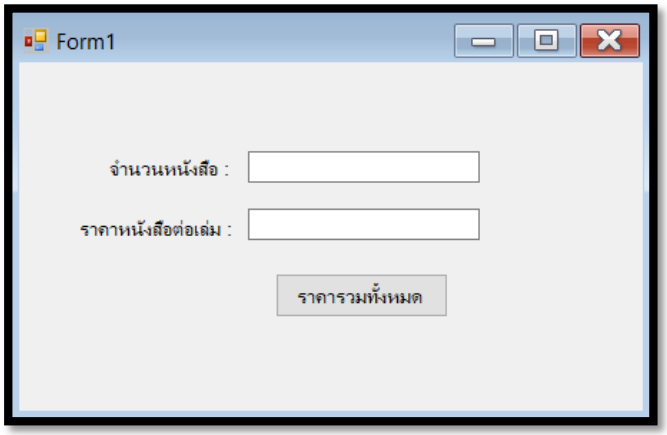
```
MessageBox.Show("บันทึกข้อมูลหรือไม่", "ปิด", MessageBoxButtons.YesNoCancel, MessageBoxIcon.Question)
```

<u>ผลลัพธ์</u>



ตัวอย่าง การใช้กล่องข้อความแสดงผลลัพธ์การคำนวณราคาหนังสือทั้งหมด

1) สร้างโปรเจคชนิด Windows Forms App ขึ้นมาใหม่ แล้วจัดวางคอนโทรลดังภาพ



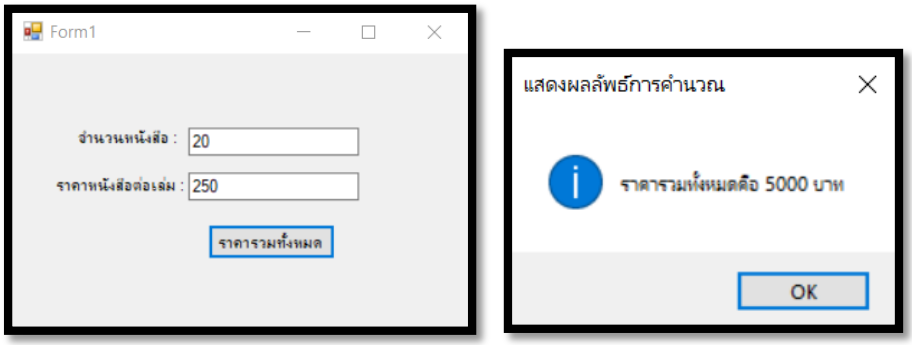
ภาพที่ 3.11

2) ดับเบิลคลิกปุ่มเพื่อเข้าสู่หน้าต่าง Code Editor เขียนโค้ดดังนี้

```
Private Sub Button1_Click(sender As Object, e As EventArgs) Handles Button1.Click
    Dim TotalPrice As Integer
    TotalPrice = Val(TextBox1.Text) * Val(TextBox2.Text)
    MessageBox.Show("ราคารวมทั้งหมดคือ " & TotalPrice & " บาท", "แสดงผลลัพธ์การคำนวณ",
                    MessageBoxButtons.OK, MessageBoxIcon.Information)
End Sub
```

ภาพที่ 3.12

3) ผลลัพธ์ที่ได้จากการรันโปรแกรม



ภาพที่ 3.13

3.7 การแปลงข้อมูลขั้นพื้นฐาน

3.7.1 ฟังก์ชั่นการแปลงข้อมูลแบบ In Line เป็นการแปลงข้อมูลโดยคอมไพล์แบบอินไลน์ที่ทำงานอย่างรวดเร็ว มีฟังก์ชันที่นิยมใช้งาน ดังนี้

ฟังก์ชัน	ไปเป็นข้อมูลชนิด	ตัวอย่างการใช้งาน	ผลลัพธ์
CByte	Byte	CByte("234.123")	234
CInt	Integer	CInt("3000.457")	3000
CLng	Long	CLng("10000000")	10000000
CCur	Currency	CCur("4598.8976")	4598.8976
Csng	Single	CSng("10000.7117")	10000.71
CDbl	Double	CDbl("10000.7117")	10000.7117
CDate	Date	CDate("7 Jan 2561")	1/7/2561
CStr	String	CStr(975.7773)	"975.7773"

3.7.2 เมธอดในคลาส Convert เป็นกลุ่มของเมธอดสำหรับแปลงข้อมูลจากชนิดหนึ่งไปเป็นชนิดอื่น ซึ่งมีเมธอดที่นิยมใช้งานดังนี้

ชื่อ	รายละเอียด
ToBoolean	เปลี่ยนจากค่าที่กำหนดไปเป็นข้อมูลชนิด Boolean ที่สอดคล้องกัน
ToByte	เปลี่ยนจากค่าที่กำหนดไปเป็นข้อมูลชนิด Byte (8-bit unsigned integer)
ToChar	เปลี่ยนจากค่าที่กำหนดไปเป็นข้อมูลชนิด Char
ToDateTime	เปลี่ยนจากค่าที่กำหนดไปเป็นข้อมูลชนิด DateTime
ToDecimal	เปลี่ยนจากค่าที่กำหนดไปเป็นข้อมูลชนิด Decimal
ToDouble	เปลี่ยนจากค่าที่กำหนดไปเป็นข้อมูลชนิด Double
ToInt16	เปลี่ยนจากค่าที่กำหนดไปเป็นข้อมูลชนิด Short (16-bit signed integer)
ToInt32	เปลี่ยนจากค่าที่กำหนดไปเป็นข้อมูลชนิด Integer (32-bit signed integer)
ToInt64	เปลี่ยนจากค่าที่กำหนดไปเป็นข้อมูลชนิด Long (64-bit signed integer)
ToSingle	เปลี่ยนจากค่าที่กำหนดไปเป็นข้อมูลชนิด Single

ชื่อ	รายละเอียด
ToString	เปลี่ยนจากค่าที่กำหนดให้อยู่ในรูปแบบสตริง

3.7.3 เมธอด TryParse เป็นการแปลงข้อมูลชนิดข้อความให้เป็นชนิดตัวเลข โดยเมื่อแปลงข้อมูลสำเร็จจะได้ค่าตามตัวเลขที่รับข้อมูลมา แต่หากไม่สำเร็จ ผลลัพธ์ที่ได้จะเป็นค่า 0 ซึ่งโปรแกรมยังคงทำงานต่อได้ มีรูปแบบดังนี้

```
Integer.TryParse("150", Price)    'แปลงเป็นตัวเลข Integer โดยตัวแปร Price = 150
Double.TryParse("12.25", Rate)    'แปลงเป็นตัวเลข Double โดยตัวแปร Rate = 12.25
Decimal.TryParse("259", Pay)      'แปลงเป็นตัวเลข Decimal โดยตัวแปร Pay = 259
Integer.TryParse("115%", Sale)    'แปลงเป็นตัวเลข Integer โดยตัวแปร Sale = 0 (อ่านค่าไม่ได้)
```

3.7.4 เมธอด Val แปลงตัวเลขที่ถูกรวมอยู่ในข้อความให้ออกมาเฉพาะค่าตัวเลข มีรูปแบบดังนี้

```
Val("475 Baht")       'คืนค่ากลับมาเป็น 475
Val("Baht 3778")      'คืนค่ากลับมาเป็น 3778
Val(" 452 88 Baht")   'คืนค่ากลับมาเป็น 45288
Val("120$")           'คืนค่ากลับมาเป็น 120
Val("32,000 Baht")    'คืนค่ากลับมาเป็น 32
Val("30%")            'คืนค่ากลับมาเป็น 30
```