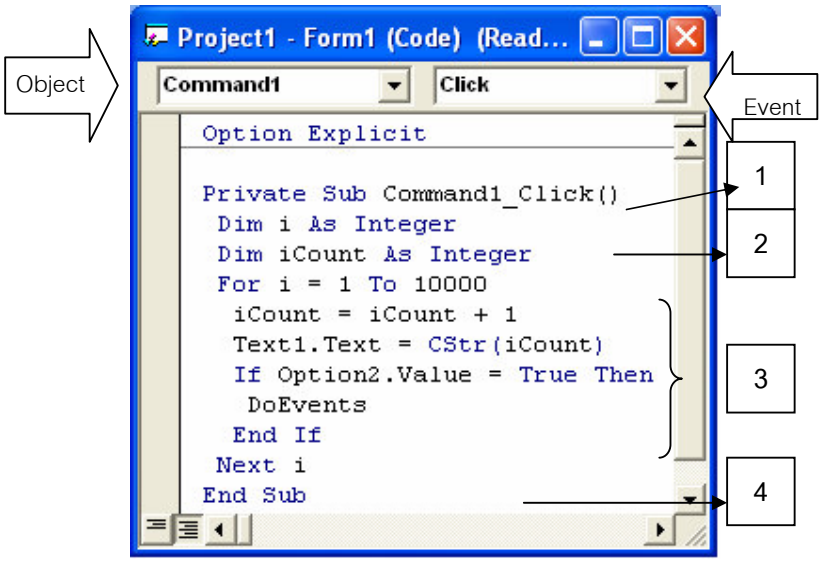
โครงสร้างในการเขียนโปรแกรม

การเขียนโปรแกรมใน Visual Basic ใช้หลักการเขียนโปรแกรมแบบ Event-Driven คือเขียนโปรแกรมเพื่อควบคุมการทำงานตามเหตุการณ์ต่างๆ เช่น on mouse click เป็นต้น การเขียนโปรแกรมควบคุม object ต่างๆที่ปรากฏอยู่บนฟอร์ม จำเป็นต้องมีการใช้โปรแกรมย่อยเข้ามาช่วยควบคุมการทำงาน โปรแกรมย่อยเหล่านี้เรียกว่า Procedure ซึ่งมี

3 รูปแบบคือ

1. Event Procedure คือโปรแกรมย่อยที่ทำงานต่อเมื่อมีเหตุการณ์บางอย่างเกิดขึ้นกับ Object นั้น
2. Sub Procedure คือโปรแกรมย่อยที่ผู้เขียนโปรแกรมพัฒนาขึ้นเมื่อต้องการแบ่งโปรแกรมออกเป็นส่วนเล็กๆเพื่อให้สามารถเรียกใช้งานได้อย่างสะดวก และสามารถปรับปรุงแก้ไขได้ง่าย
3. Function คือโปรแกรมย่อยที่มีลักษณะเดียวกับ Sub Procedure แต่มีข้อแตกต่างกันคือฟังก์ชันต้องมีการส่งค่ากลับให้กับโปรแกรมที่เรียกใช้

ตัวอย่าง



1. Program Name หมายถึงส่วนกำหนดชื่อโปรแกรมย่อย
2. Declaration หมายถึงส่วนประกาศตัวแปร
3. Statement หมายถึงส่วนคำสั่งควบคุมการทำงาน
4. End Statement หมายถึงส่วนที่บอกการจบโปรแกรม

ค่าคงที่และ ตัวแปร (Constant and Variable)

ค่าคงที่ (Constant) คือค่าที่กำหนดไว้ให้กับชื่อใดชื่อหนึ่งเพื่อความสะดวกในการใช้งานเมื่อกำหนดแล้วจะไม่มีการเปลี่ยนแปลงตลอดไป ค่าคงที่มี 2 ชนิดคือ

1. ค่าคงที่แบบตัวเลข
2. ค่าคงที่แบบชุดข้อความ

ตัวอย่าง

```
Const A = 1000
Const B = "Library"
Const C = 3.50
```

กฎเกณฑ์ในการตั้งชื่อตัวแปรมีดังนี้

- ต้องเริ่มต้นด้วยตัวอักษร มีความยาวไม่เกิน 40 ตัวอักษร
- ต้องไม่ใช้คำเฉพาะเช่น index Sub Endsub ฯลฯ
- ประกอบด้วยตัวอักษร ตัวเลข

ตัวแปร (Variable)

คือชื่อที่ตั้งขึ้นมาเพื่อใช้เก็บข้อมูลและนำมาใช้ในการทำงานของโปรแกรม ตัวอย่างรูปแบบของการกำหนด(ประกาศค่า)ตัวแปรเช่น

```
Dim salary as integer
Dim FristName, LastName as string
```

Dim *ชื่อตัวแปร* **As** *ประเภทของข้อมูล*

กฎเกณฑ์ในการตั้งชื่อตัวแปรมีดังนี้

- ต้องเริ่มต้นด้วยตัวอักษร มีความยาวไม่เกิน 255 ตัวอักษร
- ต้องไม่ใช้คำเฉพาะเช่น index Sub Endsub ฯลฯ
- ประกอบด้วยตัวอักษร ตัวเลข และ ใช้ ขีดเส้นใต้ _ ระหว่างคำ เช่น First_Name Last_Name

การประกาศตัวแปรใน Visual Basic สามารถแยกได้ 2 ประเภท คือ

1. **การประกาศตัวแปรแบบ Implicit Declaration** หมายถึง Visual Basic ยอมให้ใช้งานตัวแปรได้โดยไม่ต้องมีการประกาศตัวแปร ชนิดของข้อมูลที่ได้จะเป็นแบบ Variant ซึ่งไม่ควรนำมาใช้ เพราะใช้ทรัพยากรระบบมากและประมวลผลได้ช้า ถึงแม้ว่าผู้ขียนโปรแกรมสามารถใช้แทนข้อมูลได้ทุกชนิด
2. **การประการตัวแปรแบบ Explicit Declaration** หมายถึง ตัวแปรที่ต้องประกาศตัวแปรก่อนการใช้งานทุกครั้ง แล้วจึงสามารถนำตัวแปรนั้นๆ ไปใช้งานได้

ขอบเขตในการการกำหนด(ประกาศค่า) ตัวแปร ใน Private Public และ Static เช่น

- การกำหนด(ประกาศค่า) ตัวแปร ในโปรแกรมย่อย จะสามารถเรียกใช้ตัวแปรได้เฉพาะในโปรแกรมย่อยนั้นๆเท่านั้น เช่น Dim Salary As Integer
  Dim FirstName LastName As String
- การกำหนด(ประกาศค่า) ตัวแปรในโมดูลจะสามารถใช้ตัวแปรนั้นได้ทุกที่ในโมดูลนั้นเท่านั้นเช่น
  Private Salary As Integer
- การกำหนด(ประกาศค่า) ตัวแปรที่สามารถเรียกใช้ได้ทุกๆโมดูลที่ตัวแปรประกาศค่าใน public เช่น
  Public Salary As Integer
- การกำหนด(ประกาศค่า) ตัวแปรที่มีการเก็บค่านั้นอยู่ถึงจะเลิกใช้ตัวแปรแล้ว และกลับมาใช้อีกครั้งค่านั้นก็ยังคงอยู่ เช่น Static Salary as Integer

ประเภทของข้อมูล (Data Type)

โปรแกรมใน Visual Basic มีข้อมูลประเภทต่างๆให้ใช้ได้ตามความต้องการ เช่นจำนวนเต็ม (Integer) เศษส่วน (Single, Double) ข้อความ (string) ตัวเลขทางการเงิน (Currency) ค่าทางตรรก (boolean) เป็นต้น ข้อมูลแต่ละชนิดที่กล่าวมา จะใช้พื้นที่ในการเก็บไม่เท่ากัน รวมถึงความเร็วในการประมวลผลก็แตกต่างกันด้วย เช่น ถ้าต้องการเก็บข้อมูลเป็นตัวเลข ให้ใช้ข้อมูลชนิด Integer หรือ Long เนื่องจากใช้ทรัพยากรน้อย และประมวลผลได้เร็ว แต่ทั้งนี้ต้องขึ้นอยู่กับข้อมูลที่จะเก็บ

โปรแกรม Visual Basic แบ่งชนิดของข้อมูลดังนี้

ประเภทของข้อมูล	สัญลักษณ์พิเศษ	ใช้หน่วยความจำ	รายละเอียด
Boolean	ไม่มี	2 Bytes	เก็บค่าทางตรรกะ มีได้ 2 ค่า คือ true (จริง) , false (เท็จ) โดยที่ VB กำหนดไว้ว่า 0 มีค่าเท่ากับ false และตัวเลขจำนวนเต็มใดที่ไม่เท่ากับ 0 มีค่าเท่ากับ true

Byte	ไม่มี	1 Byte	เก็บค่าเลขจำนวนเต็มตั้งแต่ 0-255 ซึ่งเป็นรหัสแอสกี ASCII
Currency	@	8 Bytes	ใช้เก็บตัวเลขจำนวนจริง มีค่าระหว่าง -922,337,203,685,477.5808 ถึง 922,337,203,685,477.5807 ใช้สำหรับเก็บตัวเลขทางการเงินโดยเฉพาะ เพราะมีความละเอียดสูง มีทศนิยม 4 ตำแหน่ง โปรแกรมจะกำหนดค่าเริ่มต้นเป็น 0 ตัวอย่าง **Dim** x **As** currency Dim x@ หมายถึง ให้ตัวแปร x เก็บข้อมูลชนิดเลขทางการเงิน Currency
Variant	ไม่มี	ขึ้นอยู่กับชนิดข้อมูลที่เก็บ	สามารถเก็บข้อมูลได้ทุกชนิด
Integer	%	2 Bytes	ใช้เก็บค่าเลขจำนวนเต็มที่มีค่าระหว่าง -32768 ถึง 32767 โปรแกรมจะกำหนดค่าเริ่มต้นเป็น 0 ตัวอย่าง Dim x As Integer Dim x% หมายถึง ให้ตัวแปร x เก็บข้อมูลชนิดเลขจำนวนเต็ม
Long	&	4 Bytes	ใช้เก็บเลขจำนวนเต็มที่มีค่าใหญ่กว่าตัวแปรประเภท Integer มีค่าระหว่าง -2,147,483,648 ถึง 2,147,483,647 โปรแกรมจะกำหนดค่าเริ่มต้นเป็น 0 ตัวอย่าง **Dim** x **As** Long Dim x& หมายถึง ให้ตัวแปร x เก็บข้อมูลชนิดเลขจำนวนเต็ม Long
Single	!	4 Bytes	ใช้เก็บตัวเลขจำนวนจริงมีจุดทศนิยม แบ่งเป็น 2 กรณี คือ ค่าบวกอยู่ระหว่าง 1.401298E-45 ถึง 3.402823E38 และค่าลบอยู่ระหว่าง -3.402823E38 ถึง -1.401298E-45 โปรแกรมจะกำหนดค่าเริ่มต้นเป็น 0 ตัวอย่าง

			Dim x **As** Single Dim x! หมายถึง ให้ตัวแปร x เก็บข้อมูลชนิดเลขทศนิยมแบบ Single (ความละเอียดต่ำ)
Double	#	8 Bytes	ใช้เก็บตัวเลขจำนวนจริงมีจุดทศนิยมที่มีค่าใหญ่กว่าตัวแปรประเภท Single แบ่งเป็น 2 กรณี คือ ค่าบวกอยู่ระหว่าง 4.94065645841247E-324 ถึง 1.79769313486232E308 ค่าลบอยู่ระหว่าง -1.79769313486232E308 ถึง -4.94065645841247E-324 ตัวอย่าง **Dim** x **As** Double Dim x# หมายถึง ให้ตัวแปร x เก็บข้อมูลชนิดเลขทศนิยมแบบ Double (ความละเอียดสูง)
String	$	ความยาวของ string + 10 Bytes	ใช้เก็บตัวอักษร ข้อความ และตัวเลข ซึ่งต้องอยู่ภายใต้เครื่องหมาย “ ” เก็บข้อมูลได้สูงสุด 65,535 ตัวอักษร โปรแกรมจะกำหนดค่าเริ่มต้นเป็น “” ตัวอย่าง **Dim** x **As** String Dim x$ หมายถึง ให้ตัวแปร x เก็บข้อมูลชนิดข้อความ String
Date	ไม่มี	8 Bytes	ใช้สำหรับเก็บวันเดือนปี
Decimal	ไม่มี	12 Bytes	เป็นข้อมูลชนิดตัวเลขที่สามารถเก็บข้อมูลได้สูงมาก กรณีเลขจำนวนเต็ม มีค่าตั้งแต่ - 79,228,162,514,264,337,593,543,950,335 ถึง +79,228,162,514,264,337,593,543,950,335 กรณีเลขทศนิยม มีค่าตั้งแต่ - 7.9228162514264337593543950335 ถึง +- 7.9228162514264337593543950335 ค่าที่น้อยที่สุดที่ไม่เท่ากับศูนย์ที่สามารถเก็บได้คือ

			0.00000000000000000000000000001
Object	ไม่มี	4 Bytes	เป็นตัวแปรที่อ้างถึง Object ต่างๆของ Application

ตัวดำเนินการ (Operator)

ตัวดำเนินการทางคณิตศาสตร์ (Arithmetic Operator) เป็นตัวดำเนินการที่ใช้ในการคำนวณ

ชื่อตัวดำเนินการ	ลักษณะตัวดำเนินการ	ตัวอย่าง	ผลลัพธ์
การบวก	+	15+20	35
การลบ	-	20-15	5
การคูณ	*	5*4	20
การหาร	/	9/3	3
การหารจำนวนเต็ม	\	10\3	3
การ Modulo	Mod	10 Mod 3	1
การยกกำลัง	^	2^2	4
การเปลี่ยนเครื่องหมาย	-	5	-5

รายละเอียดของตัวดำเนินการทางคณิตศาสตร์

การบวก, ลบ, คูณ, หาร เป็นไปตามกฎทางคณิตศาสตร์ ที่น่าสนใจคือ

- การหารจำนวนเต็ม หมายถึง จะเอาแต่เฉพาะเลขที่เป็นจำนวนเต็มเท่านั้น ส่วนเศษ จะปัดทิ้ง เช่น 10\6=1.6666 แต่จะได้ผลลัพธ์เท่ากับ 1 แม้ว่า .6666 จะเกินครึ่งก็ตาม ไม่มีการปัดเศษแต่อย่างใด
- การ Mod หมายถึง จะเอาแต่เศษที่เหลืออยู่จากการ Mod เช่น 10 Mod 3 =1 เนื่องจาก 10 หาร 3 ไม่ลงตัว เหลือเศษ 1 จึงได้ผลลัพธ์ 1, 20 Mod 6 =2 (เหลือเศษ 2 นั่นเอง)

ตัวดำเนินการทางด้านการเปรียบเทียบ (Comparison Operators)

หมายถึง การนำพจน์ตั้งแต่ 2 พจน์ขึ้นไป มาเปรียบเทียบกัน เพื่อทดสอบเงื่อนไข หรือสร้างเงื่อนไข ผลลัพธ์ที่ได้จะมีค่าเป็นจริง หรือเป็นเท็จเท่านั้น ตัวดำเนินการทางด้านการเปรียบเทียบ มีรายละเอียดดังนี้

ตัวดำเนินการ	ชื่อตัวดำเนินการ	ตัวอย่าง
<	น้อยกว่า	2 < 4
>	มากกว่า	4 > 3
<=	น้อยกว่าหรือเท่ากับ	4 <= 1
>=	มากกว่าหรือเท่ากับ	7>= 2
=	เท่ากับ	3 = 2+1
<>	ไม่เท่ากับ	8 <> 5+2

ตัวดำเนินการทางด้านตรรกะ (Logical Operator)

เป็นตัวดำเนินการที่ใช้ในการเชื่อนิพจน์ (Expression) หลายนิพจน์เข้าด้วยกัน ซึ่งผลลัพธ์ที่ได้มี 2 ค่า เท่านั้น คือ True และ False

A	B	And	Or	Xor	Eqv	Imp	Not A
True	True	True	True	False	True	True	False
True	False	False	True	True	False	False	False
False	True	False	True	True	False	True	True
False	False	False	False	False	True	True	True

รายละเอียดของตัวดำเนินการทางตรรกะ

- **ตัวดำเนินการ And** จะมีกรณีเดียวที่เป็นจริงก็คือ ทั้ง 2 นิพจน์ต้องเป็นจริง
- **ตัวดำเนินการ Or** จะมีกรณีเดียวที่เป็นเท็จก็คือ ทั้ง 2 นิพจน์ต้องเป็นเท็จ
- **ตัวดำเนินการ Xor** ถ้า 2 นิพจน์มีค่าต่างกัน จะได้ผลลัพธ์เป็นจริง
- **ตัวดำเนินการ Eqv** ถ้า 2 นิพจน์มีค่าเหมือนกัน จะได้ผลลัพธ์เป็นจริง
- **ตัวดำเนินการ Imp** จะมีกรณีเดียวที่มีค่าเป็นเท็จคือ นิพจน์หน้าเป็นจริง นิพจน์หลังเป็นเท็จ
- **ตัวดำเนินการ Not** เป็นการสลับค่าของนิพจน์ เช่น ถ้า A เป็นจริง จะได้ผลลัพธ์เป็นเท็จ

ตัวดำเนินการทางด้านการเชื่อมข้อความ (Concentration Operators)

โดยหน้าที่หลักแล้ว จะเป็นการเชื่อมข้อความ 2 ข้อความเข้าด้วยกัน แต่ยังมีกรณียกเว้นที่จะเป็นการบวกกันของนิพจน์ 2 นิพจน์ เข้าด้วยกัน ซึ่งขึ้นอยู่กับชนิดของนิพจน์ที่จะมากระทำกัน สามารถแยกออกเป็นกรณี ได้ดังนี้

ตัวดำเนินการ	กรณี	ตัวอย่าง	ผลลัพธ์
+	String + String	"Visual"+"Basic 6.0"	"Visual Basic 6.0"
&	String & String	"Visual" & "Basic 6.0"	"Visual Basic 6.0"
+	String(numeric)+numeric	"20"+6	26
&	String(numeric)&numeric	"20"&6	206

การใช้คำสั่งและฟังก์ชั่นมาตรฐานของ Visual Basic

คำสั่ง (statements) และฟังก์ชัน (functions) เป็นสิ่งหนึ่งที่ต้องทำความเข้าใจกับโครงสร้าง และการนำไปใช้งาน เนื่องจากว่า ถ้าทราบความหมาย รูปแบบการใช้งาน และทราบหน้าที่มากเท่าใด จะสามารถปรับแต่งโค้ด ให้ประมวลผลได้รวดเร็วมากขึ้น

ข้อแตกต่างของคำสั่งและฟังก์ชันก็คือ ถ้าใช้คำสั่ง จะหมายถึง สั่งให้ Visual Basic ทำงานตามขั้นตอนและไม่มีการคืนผลลัพธ์ หรือคืนค่าออกมาจากคำสั่งนั้นๆ เช่น คำสั่งในการวนลูป Do While-Loop

สำหรับฟังก์ชัน หมายถึง สั่งให้ Visual Basic ประมวลผลไปตามหน้าที่ของแต่ละฟังก์ชัน เช่นเดียวกับคำสั่ง แต่จะคืนค่าออกมา 1 ค่า ซึ่งโปรแกรมเมอร์จะเอาค่าดังกล่าวนี้ ไปเป็นเงื่อนไข ในการประมวลผลต่อไป ดังนั้นเมื่อใช้งานฟังก์ชัน จะต้องสร้างตัวแปร 1 ตัวต่อ 1 ฟังก์ชัน เพื่อรับค่าจากฟังก์ชัน และตัวแปรดังกล่าว จะต้องเป็นข้อมูลชนิดเดียวกันกับ ค่าที่ฟังก์ชันนั้นๆ ส่งคืนมาด้วย

กลุ่มคำสั่ง Statement

กลุ่มคำสั่งที่ต้องใช้มากที่สุด ไม่ว่าจะเขียนโปรแกรมด้วยภาษาอะไรก็ตาม และก็ถือได้ว่า เป็นกลุ่มคำสั่งที่มีความสำคัญ มากที่สุดเช่นกัน ซึ่งประกอบไปด้วยกลุ่มคำสั่ง 2 ชุดดังนี้

- ***กลุ่มคำสั่งที่สร้างเงื่อนไขในการตัดสินใจ***

คำสั่งที่สร้างเงื่อนไขในการตัดสินใจมี 2 คำสั่งคือ 1. If Then Else และ 2. Select Case

1. คำสั่ง If-Then Else

เป็นการสร้างเงื่อนไข ตรวจสอบเงื่อนไข หรือเป็นทางเลือก เพื่อตัดสินใจแบบ ธรรมดาและง่ายที่สุด นั่นคือ ถ้าเงื่อนไขเป็นจริง ก็จะทำตามเงื่อนไขที่ได้สร้างไว้ มีรูปแบบการใช้งานดังนี้

```
If condition Then statements
```

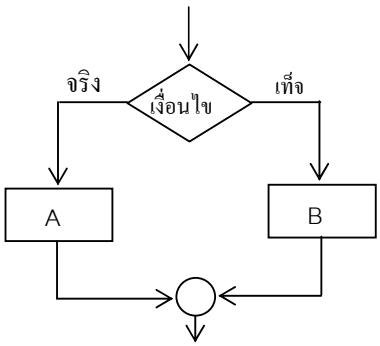
- **ตัวแปร condition** หมายถึง เงื่อนไขที่กำหนดขึ้นมา

- **ตัวแปร statements** หมายถึง เมื่อเงื่อนไขเป็นจริงแล้ว จะให้ทำอะไร เช่น

If cnt < 5 **Then** a = b + cnt

จากตัวอย่าง ถ้าตัวแปร cnt น้อยกว่า 5 ให้เอาตัวแปร b+cnt แล้วเก็บไว้ในตัวแปร a จะเห็นได้ว่าเฉพาะกรณีตัวแปร cnt น้อยกว่า 5 เท่านั้น จึงจะมีการบวกกัน แต่ถ้าตัวแปร cnt มากกว่า 5 ก็จะไม่มีผลอะไร แต่ยังสามารถสร้างเงื่อนไข เพิ่มขึ้น เผื่อไว้ในกรณีที่ เงื่อนไขไม่เป็นจริง โดยมีรูปแบบการใช้งานดังนี้

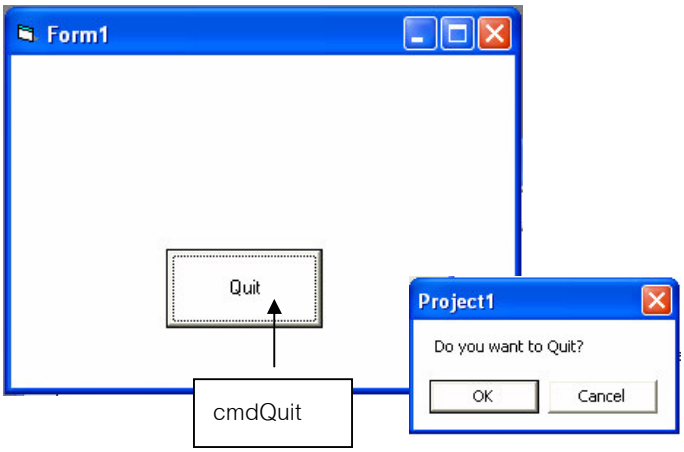
If condition **Then**
    statements
Else
    elsestatements
End If



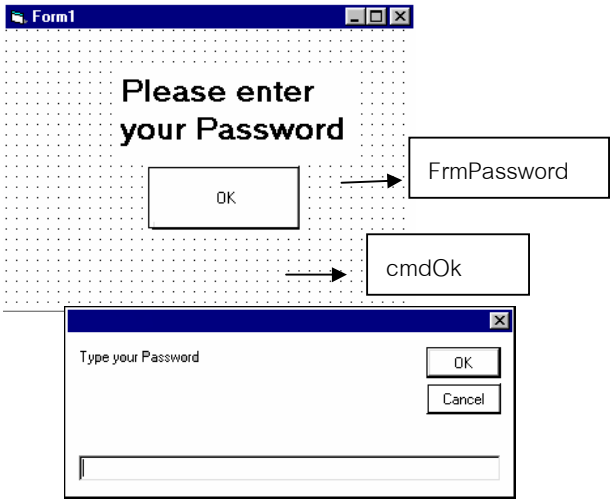
- **คำสั่ง Else** หมายถึง กรณีที่เงื่อนไขดังกล่าวเป็นเท็จ
- **คำสั่ง End If** หมายถึง จบเงื่อนไข
- **ตัวแปร elsestatements** หมายถึง ชุดคำสั่งที่จะทำ เมื่อเงื่อนไขเป็นเท็จ

ตัวอย่าง Demo11 สร้างปุ่ม Quit เมื่อคลิกที่ปุ่มให้มี MsgBox เพื่อยืนยันการออกจากโปรแกรม

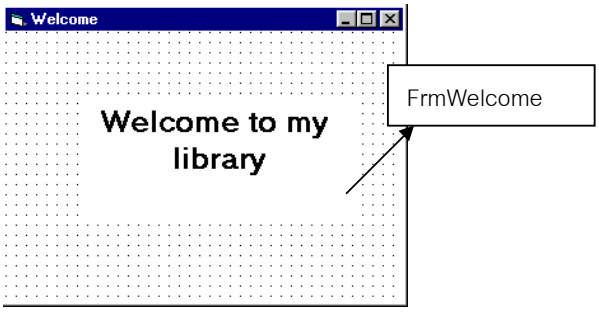
```
Private Sub cmdquit_Click()

Dim ans As Integer
ans = MsgBox("Do you want to Quit?", vbOKCancel)
If ans = vbCancel Then
   Cancel = True
Else
   End
End If

End Sub
```



ตัวอย่าง **Demo12**



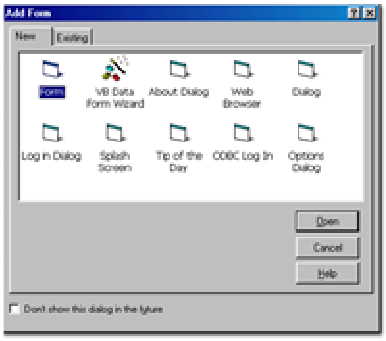

```
Private Sub cmdOk_Click()
    Dim Password, pword
    Password = "CU"
    pword = InputBox("Type your Password")

If   pword <> Password Then
    MsgBox "Sorry, incorrect Password"
Else
    frmPassword.Hide
    frmWelcome.Show
End If
End Sub
```



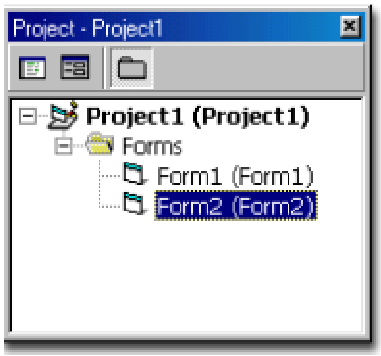
บทเสริม

การเพิ่ม Form หรือโมดูลอื่นๆ เข้ามาในสภาพแวดล้อม

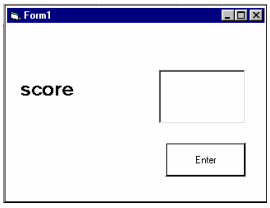
1. เลือกเมนู Project -> Add Form หรือคลิกที่ปุ่ม  บนทูลบาร์



2. ที่หน้าต่าง Project Explorer จะมีรายการแสดงเพิ่มขึ้นมา ซึ่งอยู่ภายใต้ Project เดียวกัน จากรูปจะเห็นว่า Project1 ประกอบด้วย Form 2 ฟอร์ม



แบบฝึกหัดที่3 จงสร้างจอภาพเพื่อคิดคะแนนของนักเรียนโดยใช้คำสั่งเงื่อนไข If- Then-Else โดยมีเงื่อนไขดังนี้



0-49 ได้ เกรด F

50-59 ได้ เกรด D

60-69 ได้ เกรด C

70-79 ได้ เกรด B

80-100 ได้ เกรด A

ตัวเลขสูงกว่า 100 มี ข้อความ บอกให้รู้ว่าคะแนนจะสูงกว่า 100 ไม่ได้

2. Slect-Case

เป็นคำสั่งที่ใช้เลือกในการตัดสินใจของโปรแกรมที่มีหลายๆเงื่อนไขเพื่อแก้ปัญหาการใช้คำสั่ง If-Then-Else ต่อกันยาวๆ (nested if)

```
Select Case   testexpression
 Case expressionlist1
       statements1
 Case [Is]     expressionlist2
   statements2
 [Case [Is]     expressionlist-n ]
   [statements-n ]
 Case Else
   elsestatements
End Select
```

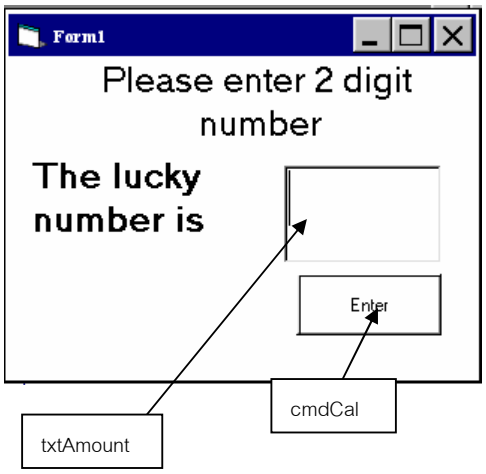
- **คำสั่ง Select Case** หมายถึง เริ่มกรณีตรวจสอบแบบ Select Case
- **ตัวแปร testexpression** หมายถึง ตัวแปรที่จะนำมาตรวจสอบ
- **ตัวแปร expressionlist1** หมายถึงค่าของตัวแปรในกรณีที่ 1
- **ตัวแปร statement1** หมายถึง ชุดคำสั่งที่ต้องกระทำ เมื่อค่าของตัวแปรตรงกับกรณีที่ 1
- **ตัวแปร expressionlist2** หมายถึงค่าของตัวแปรในกรณีที่ 2

- **ตัวแปร statement2** หมายถึง ชุดคำสั่งที่ต้องกระทำ เมื่อค่าของตัวแปรตรงกับกรณีที่ 2
- **คำสงวน Is** จะใช้ในกรณีที่คุณใส่เฉพาะเงื่อนไข ซึ่ง VB จะใส่ให้คุณโดยอัตโนมัติ
- **คำสั่ง Case Else** หมายถึง เมื่อค่าของตัวแปร ไม่เท่ากับกรณีที่ผ่านมาทั้งหมด ให้มาที่กรณีนี้
- **ตัวแปร elsestatements** หมายถึง ชุดคำสั่งสำหรับกรณีที่ค่าของตัวแปรไม่ตรงกับกรณีใดเลย
- **คำสั่ง End Select** หมายถึง จบการตรวจสอบแบบ Select Case

ตัวอย่าง Demo13 เกมทายตัวเลข

กำหนดตัวเลขในโปรแกรม เท่ากับ 20 ให้ผู้ใช้ทายตัวเลขที่กำหนดไว้

```
Private Sub cmdCal_Click()
    Select Case txtAmount.Text
        Case Is < 20
                MsgBox "TOO  LOW"
        Case Is = 20
                MsgBox "You are the winner"
        Case Else
                MsgBox "TOO  HIGH"
    End Select
End Sub
```



แบบฝึกหัดที่4 จากแบบฝึกหัดที่3 จงเขียนโปรแกรมด้วยคำสั่ง Select case

- ***กลุ่มคำสั่งในการวนลูป หรือทำซ้ำ***

คำสั่งในกลุ่มนี้ เป็นอีกกลุ่มหนึ่งที่ต้องใช้งานบ่อยครั้งที่สุด ไม่ว่าจะพัฒนาแอพพลิเคชันด้วยภาษาอะไรก็ตาม คำสั่งนี้มีหน้าที่สำหรับสั่งให้ VB ประมวลผลซ้ำกลุ่มคำสั่งเดิม วนไปเรื่อยๆ จนกว่าจะตรงกับเงื่อนไขที่ต้องการ และยังมีหน้าที่สำหรับลดขั้นตอนในการเขียนโค้ดที่มีลักษณะเหมือนๆ กันได้อีกด้วย กลุ่มคำสั่งทำซ้ำมีดังนี้ 1. For-Next 2. Do While-Loop 3. Do-Loop While 4. Do Until-Loop 5.Do-Loop Until

1. คำสั่ง For-Next

คำสั่งนี้ มีหน้าที่สำหรับสั่งให้ VB วนลูป เหมาะสำหรับการทำงาน ที่ทราบจำนวนรอบ หรือทราบขอบเขตการวนที่แน่นอน มีรูปแบบการใช้งานดังนี้

```
For counter = start To end   [Step step]
  statements
[Exit For]
  [statements ]
Next [counter ]
```

- **ตัวแปร counter** หมายถึง ตัวแปรที่กำหนดขึ้นมาเพื่อ เป็นตัวนับรอบที่ใช้ในการวนลูป
- **ตัวแปร start** หมายถึง ค่าเริ่มต้นของตัวนับ เป็นเลขจำนวนจริง ควรที่จะใช้เลขจำนวนเต็มเท่านั้น เพื่อง่ายต่อการตรวจสอบค่าตัวนับ
- **คำสงวน To** หมายถึง ขอบเขตการนับ
- **ตัวแปร end** หมายถึง ค่าสิ้นสุดของตัวนับ ไม่จำเป็นต้องเป็นตัวเลข ในบางครั้งอาจสร้างเป็นเงื่อนไขที่ไม่ใช่ตัวเลขก็ได้ แต่ในการใช้งานโดยทั่วๆ ไป จะเป็นตัวเลข และควรจะเป็นเลขจำนวนเต็มด้วย
- **คำสงวน Step** หมายถึง เป็นการกำหนดค่าที่จะเพิ่มขึ้นของตัวนับในแต่ละรอบ ถ้าไม่กำหนด จะเพิ่มรอบละ 1
- **ตัวแปร step** หมายถึง ค่าของตัวนับที่จะเพิ่มขึ้นในแต่ละรอบ จะต้องเป็นเลขจำนวนจริงเท่านั้น ในการใช้งานตามปกติ ควรใช้เลขจำนวนเต็ม เพื่อง่ายต่อการตรวจสอบจำนวนรอบในภายหลัง
- **ตัวแปร statements** หมายถึง ชุดคำสั่งที่คุณต้องการวนลูป
- **คำสั่ง Exit For** หมายถึง คำสั่งที่บังคับให้ออกจากลูปทันที จะใช้ในบางกรณีเท่านั้น เช่นการดักจับข้อผิดพลาด

ตัวอย่าง Code

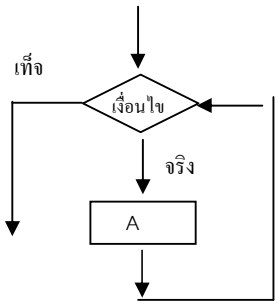
```
Private Sub Form_Click()

Dim A As Integer
    For A = 1 To 10
        Form1.Print "Number :  " & A
    Next A

End Sub
```

```
Private Sub Form_Click()

Dim A As Integer
  For A = 0 To 10 Step 2
    Form1.Print "Number :  " & A
  Next A

End Sub
```

2. Do While-Loop

คือคำสั่งที่ให้ทำงานซ้ำๆกันในขณะที่เงื่อนไขยังเป็นจริง เมื่อเงื่อนไขเป็นเท็จจะออกจาก loop เลย ควรทดสอบค่าตรรกะก่อน ถ้าเป็นจริงจึงจะทำคำสั่งต่อมา มีข้อที่น่าสังเกตคือ การใช้ ลูปแบบนี้ จะมีโอกาสที่ไม่ต้องวนลูปเลยแม้แต่รอบเดียว นั่นคือ ถ้าเงื่อนไขที่เข้ามาเป็นเท็จ ก็ไม่ต้องวนทันที

```
Do While condition
  statements
  [Exit Do]
Loop
```



- **ตัวแปร condition** หมายถึง เงื่อนไขที่กำหนดขึ้นมา
- **ตัวแปร statements** หมายถึง ชุดคำสั่งที่ต้องการทำซ้ำ
- **คำสั่ง Exit Do** หมายถึง ออกจาก Do While-Loop ทันที
- **คำสั่ง Loop** หมายถึง ขอบเขตสิ้นสุดการวนลูป

ลูปแบบ Do While นี้ จะวนลูปก็ต่อเมื่อเงื่อนไข (ตัวแปร condition) ยังเป็นจริงอยู่ ถ้าเงื่อนไขดังกล่าวเป็นเท็จเมื่อใด ก็จะออกจากลูปทันที

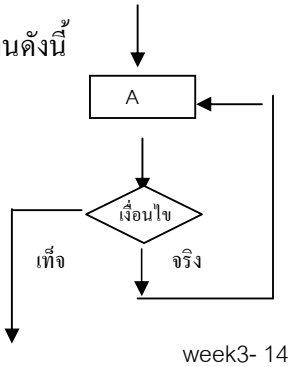
ตัวอย่าง Code

Private Sub Form_Click() Dim A As Integer A = 0 Do While (A < 10) A = A + 1 Form1.Print "Number : " & A Loop Form1.Print "The End" **End Sub**	**Private Sub Form_Click()** Dim A As Integer A = 10 Do While (A < 10) A = A + 1 Form1.Print "Number : " & A Loop Form1.Print "The End" **End Sub**

3. คำสั่ง Do-Loop While

คำสั่งนี้เป็นการสั่งวนลูป มีจำนวนรอบขึ้นอยู่กับเงื่อนไข มีรูปแบบการใช้งานดังนี้

```
Do
  statements
  [Exit Do]
Loop While condition
```



- **ตัวแปร condition** หมายถึง เงื่อนไขที่คุณกำหนดขึ้นมา
- **ตัวแปร statements** หมายถึง ชุดคำสั่งที่คุณต้องการทำซ้ำ
- **คำสั่ง Exit Do** หมายถึง ออกจาก Do- Loop While ทันที
- **คำสั่ง Loop** หมายถึง ขอบเขตสิ้นสุดการวนลูป

จะเห็นได้ว่า มีลักษณะคล้ายกับแบบที่ 1 เพียงแต่ย้ายเงื่อนไขมาไว้ด้านล่าง ซึ่งหมายความว่า ลูปแบบนี้จะวนอย่างน้อยที่สุด 1 รอบ เพื่อตรวจสอบเงื่อนไข (ตัวแปร condition) ในรอบแรกที่เข้ามาก่อนว่าเป็นจริงหรือเท็จ ถ้าจริงก็จะวนลูปต่อไปตามปกติ แต่ถ้าเป็นเท็จ จะออกจากลูปทันที *เช่น*

ตัวอย่าง Code

Private Sub Form_Click()
 	
Dim A As Integer
  A = 0
  Do
  A = A + 1
  Form1.Print "Number : " & A
  Loop While (A < 10)
  Form1.Print "The End"
 	
End Sub
	Private Sub Form_Click()
	
	Dim A As Integer
	  A = 10
	  Do
	  A = A + 1
	  Form1.Print "Number : " & A
	  Loop While (A < 10)
	  Form1.Print "The End"
	
	End Sub

ข้อแตกต่างของลูป Do While - Loop กับ Do-Loop While นั่นคือ ถ้าเป็น Do While-Loop กรณีที่เงื่อนไขเป็นเท็จ บนฟอร์ม จะไม่ปรากฏตัวเลขอะไรเลย เพราะเนื่องจากว่า ไม่ได้มีการวนลูปแต่อย่างใด เพราะเงื่อนไขเป็นเท็จ (ค่า i=11) แต่ลูปแบบ Do-Loop While จะปรากฏค่าของตัวเงื่อนไขออกมาก่อน ถ้าจริง ก็จะวนลูปตามปกติ แต่กรณีนี้เป็นเท็จ จึงแสดงค่าออกมา 1 ค่า ซึ่งเกิดมาจาก การวนรอบแรกนั่นเอง

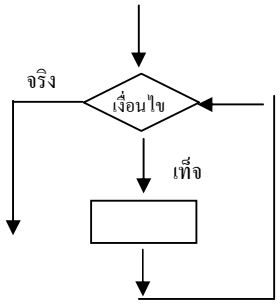
4. คำสั่ง Do Until-Loop

คำสั่งนี้มีหน้าที่สั่งให้วนลูปเช่นกัน มีจำนวนรอบขึ้นอยู่กับเงื่อนไข (condition) ถ้าเงื่อนไขเป็นเท็จจึงจะวนลูป แต่ถ้าเป็นจริงจะไม่มีการวนแต่อย่างใด ซึ่งจะตรงกันข้ามกับลูปชนิด Do While-Loop มีรูปแบบการใช้งานดังนี้

Do Until condition

statements

[Exit Do]



statements

Loop

- **ตัวแปร condition** หมายถึง เงื่อนไขที่กำหนดขึ้นมา
- **ตัวแปร statements** หมายถึง ชุดคำสั่งที่ต้องการทำซ้ำ
- **คำสั่ง Exit Do** หมายถึง ออกจาก Do Until- Loop ทันที
- **คำสั่ง Loop** หมายถึง ขอบเขตสิ้นสุดการวนลูป เช่น

ตัวอย่าง Code

Private Sub Form_Click() Dim A As Integer A = 0 Do Until (A > 10) A = A + 1 Form1.Print "Number : " & A Loop Form1.Print "The End" **End Sub**	**Private Sub Form_Click()** Dim A As Integer A = 10 Do Until (A < 10) A = A + 1 Form1.Print "Number : " & A Loop Form1.Print "The End" **End Sub**
	เกิด Overflow เนื่องจาก

5. การใช้ คำสั่งทำซ้ำ Do-Loop Until

คำสั่งการวนลูปแบบนี้มีลักษณะคล้ายกับลูปแบบ Do-Loop While เพียงแต่ เงื่อนไขที่เข้ามาจะต้องเป็นเท็จ จึงจะวนลูป ถ้าเงื่อนไขเป็นจริง จะออกจากลูปทันที การใช้ลูปแบบนี้จะมีการวนอย่างน้อยที่สุด 1 รอบเช่นกัน เนื่องจากการวนรอบแรก ที่จะต้องมีการตรวจสอบเงื่อนไขนั่นเอง มีรูปแบบการใช้งานดังนี้

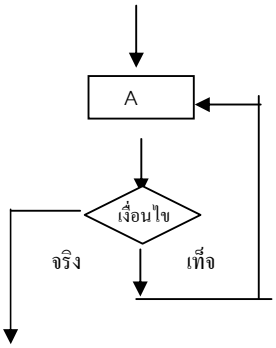
Do

statements

[Exit Do]

[statements]

Loop Until condition



- **ตัวแปร condition** หมายถึง เงื่อนไขที่กำหนดขึ้นมา
- **ตัวแปร statements** หมายถึง ชุดคำสั่งที่ต้องการทำซ้ำ
- **คำสั่ง Exit Do** หมายถึง ออกจาก Do Loop-Until ทันที
- **คำสั่ง Loop** หมายถึง ขอบเขตสิ้นสุดการวนลูป เช่น

ตัวอย่าง Code

Private Sub Form_Click()
 	
Dim A As Integer
  A = 0
  Do
  A = A + 1
  Form1.Print "Number : " & A
  Loop Until (A > 10)
  Form1.Print "The End"
 	
End Sub
	Private Sub Form_Click()
	
	Dim A As Integer
	  A = 10
	  Do
	  A = A + 1
	  Form1.Print "Number : " & A
	  Loop Until (A < 10)
	  Form1.Print "The End"
	
	End Sub

	เกิด Overflow เนื่องจาก

เฉลยแบบฝึกหัดที่ 3 และ 4

```
Private Sub cmdCal_Click()
 If txtScore.Text < 50 Then
    MsgBox "Grade F", vbOKOnly, "คุณได้"
Else
 If txtScore.Text < 60 Then
    MsgBox "Grade D", vbOKOnly, "คุณได้"
  Else
 If  txtScore.Text < 70 Then
    MsgBox "Grade C", vbOKOnly, "คุณได้"
    Else
     If   txtScore.Text < 80 Then
        MsgBox "Grade B", vbOKOnly, "คุณได้"
      Else
          If   txtScore.Text <= 100 Then
             MsgBox "Grade A", vbOKOnly, "คุณได้"
         Else
         MsgBox "MAX = 100", vbOKOnly, "try again"
End If
End If
End If
End If
End If

End Sub
```

```
Private Sub cmdCal_Click()
Dim score As Integer
score = txtScore.Text
Select Case score
Case 0 To 49
   MsgBox "Grade F", vbOKOnly, "คุณได้"
Case 50 To 59
   MsgBox "Grade D", vbOKOnly, "คุณได้"
Case 60 To 69
   MsgBox "Grade C", vbOKOnly, "คุณได้"
Case 70 To 79
    MsgBox "Grade B", vbOKOnly, "คุณได้"
Case 80 To 100
    MsgBox "Grade A", vbOKOnly, "คุณได้"
Case Else
    MsgBox "MAX = 100", vbOKOnly, "try again"

End Select
txtScore.Text = ""
End Sub
```