แผนบริหารการสอนประจำบทที่ 5

คำสั่งควบคุมทิศทางการทำงานของโปรแกรม

หัวข้อประจำบท

1. ทิศทางการทำงานแบบเรียงลำดับ
2. คำสั่งควบคุมทิศทางแบบมีเงื่อนไข
3. คำสั่งควบคุมทิศทางแบบทำซ้ำ

วัตถุประสงค์เชิงพฤติกรรม

1. เพื่อให้ผู้เรียนมีความรู้ความเข้าใจ โครงสร้างการทำงานของโปรแกรม
2. เพื่อให้ผู้เรียนมีความรู้ความเข้าใจเกี่ยวกับโครงสร้างการทำงานแบบมีเงื่อนไข และแบบวนซ้ำ
3. เพื่อให้ผู้เรียนสามารถมีความรู้ความเข้าใจ คำสั่งควบคุมการทำงานของโปรแกรมแบบมีเงื่อนไขและแบบทำซ้ำได้
4. เพื่อให้ผู้เรียนสามารถวิเคราะห์ นำคำสั่งไปใช้ในการแก้โจทย์ปัญหาได้อย่างงเหมาะสม

วิธีการสอนและกิจกรรม

1. ผู้สอนบรรยายในชั้นเรียน ตามหัวข้อเนื้อหาในเอกสารประกอบการสอน วิชาเทคโนโลยีการเขียนโปรแกรมคอมพิวเตอร์
2. ผู้เรียนทำแบบฝึกหัดเพื่อส่งเสริมความเข้าใจ กระบวนการคิด และการแก้ปัญหา
3. ผู้สอนสาธิตการใช้งานโปรแกรมพร้อมทั้งให้ผู้เรียนฝึกปฏิบัติตาม

สื่อการเรียนการสอน

1. เอกสารประกอบการสอน วิชา เทคโนโลยีการเขียนโปรแกรมคอมพิวเตอร์
2. โปรแกรม PowerPoint
3. โปรแกรม Microsoft Visual Studio 2010

การวัดผลและการประเมินผล

1. สังเกตจากการอภิปราย การตอบคำถาม และซักถามระหว่างเรียน
2. การฝึกปฏิบัติการใช้โปรแกรม
3. การทำแบบฝึกหัดท้ายบท

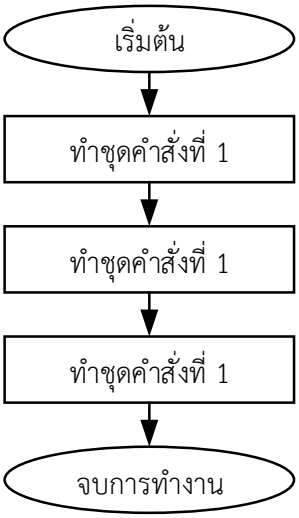
บทที่ 5
คำสั่งควบคุมทิศทางการทำงานของโปรแกรม

ทิศทางการทำงานของโปรแกรมขึ้นอยู่กับขั้นตอนการทำงานของโปรแกรมหรืออัลกอริทึม ซึ่งจะนำมาสร้างเป็นโปรแกรมโดยใช้คำสั่งควบคุมการทำงานของโปรแกรมให้สามารถทำงานได้ตามที่ต้องการ (Newsome, 2011) โดยคำสั่งควบคุมทิศทางการทำงานเป็นมาตรฐานของทุกโปรแกรมภาษา โดยแต่ละโปรแกรมภาษามีคำสั่งควบคุมทิศทางการทำงาน โครงสร้างและรูปแบบการใช้ความแตกต่างกันขึ้นอยู่กับแต่ละโปรแกรมภาษา

คำสั่งควบคุมทิศทาง (Control Flows) หมายถึง คำสั่งหรือกฎเกณฑ์ ที่ใช้สำหรับควบคุมทิศทางการทำงาน เพื่อให้บรรลุเป้าหมาย (สุชาติ คุ้มมณี, 2557) และเป็นการสั่งงานโดยชุดคำสั่งที่จะถูกประมวลผลด้วยโปรแกรม (Dierbach, 2013) ทิศทางการทำงานของโปรแกรม ประกอบไปด้วยทิศทางการทำงาน 3 ประเภท ได้แก่ ทิศทางแบบเรียงลำดับ ทิศทางแบบทางเลือก และทิศทางแบบทำซ้ำ แต่ละประเภทมีลักษณะของการทำงานที่แตกต่างกันออกไป มีรายละเอียดดังต่อไปนี้

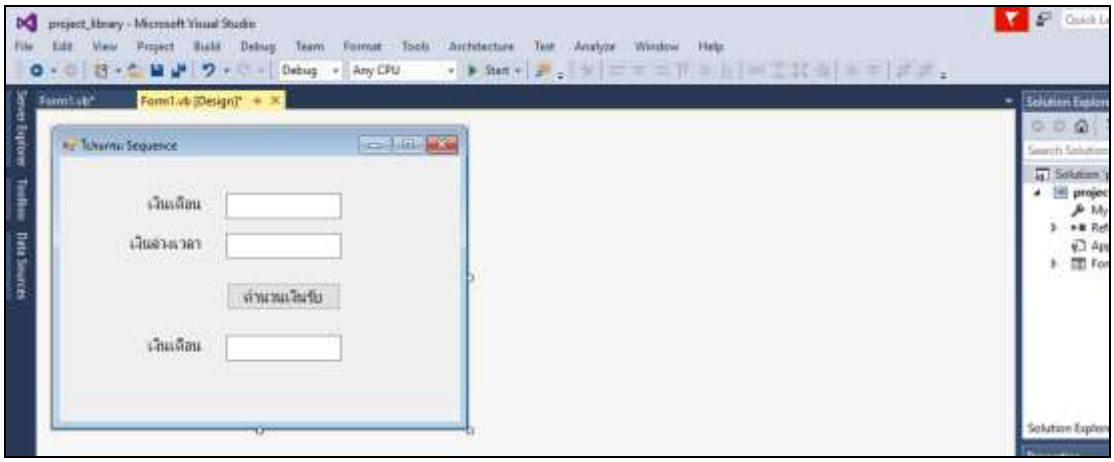
5.1 ทิศทางการทำงานแบบเรียงลำดับ

ทิศทางการทำงานแบบเรียงลำดับ (Sequence Structure) คือ ลักษณะของการทำงานของโปรแกรมที่มีการทำงานทุกขั้นตอน โดยเรียงลำดับการทำงานตั้งแต่ต้นจนจบของการทำงาน ไม่มีการข้ามกระโดดหรือแบ่งเส้นทางการทำงานของโปรแกรม ดังภาพที่ 5.1



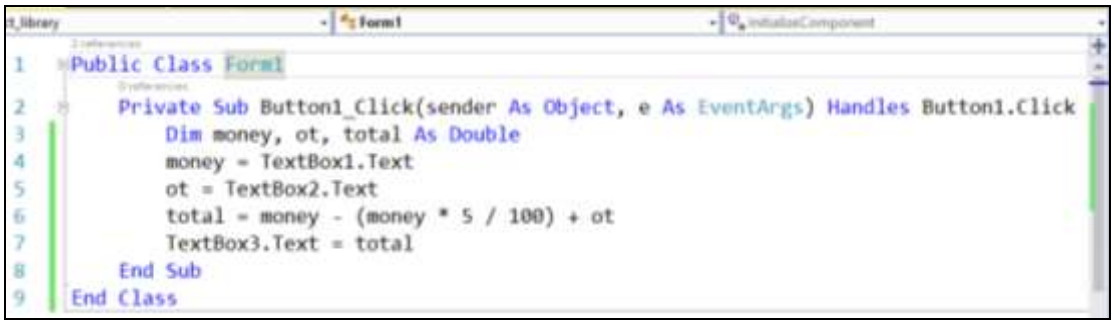
ภาพที่ 5.1 โครงสร้างการทำงานแบบเรียงลำดับ

ทิศทางการทำงานแบบเรียงลำดับนี้ เป็นโครงสร้างที่ทำงานทุกขั้นตอน ดังนั้นจะไม่มีคำสั่งที่จะควบคุมทิศทางการทำงานยกตัวอย่างโปรแกรม ให้คำนวณเงินรับที่ได้ซึ่งได้จากเงินเดือนหักประกันสังคม 5 เปอร์เซ็นต์ รวมกับเงินล่วงเวลา ซึ่งมีรายละเอียดขั้นตอนการพัฒนาโปรแกรมดังต่อไปนี้



ภาพที่ 5.2 ตัวอย่างการออกแบบโปรแกรมแบบเรียงลำดับ

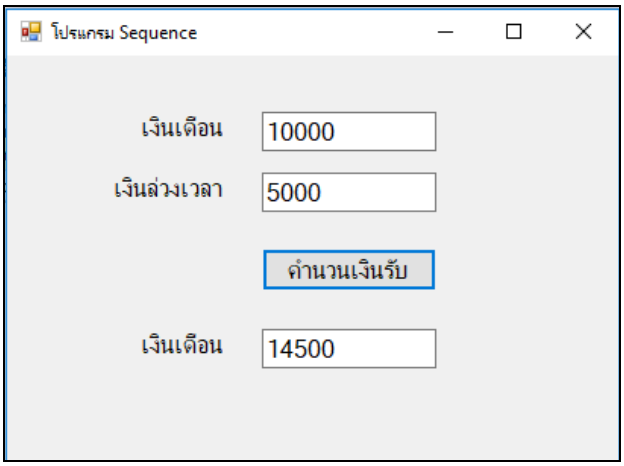
จากภาพที่ 5.2 การออกแบบส่วนประสานกราฟิกกับผู้ใช้ ประกอบด้วยการรับข้อมูลที่เป็นเงินเดือนและล่วงเวลา ปุ่มคำนวณเงินรับ และแสดงผลเงินรับที่ได้



```
Public Class Form1
    Private Sub Button1_Click(sender As Object, e As EventArgs) Handles Button1.Click
        Dim money, ot, total As Double
        money = TextBox1.Text
        ot = TextBox2.Text
        total = money - (money * 5 / 100) + ot
        TextBox3.Text = total
    End Sub
End Class
```

ภาพที่ 5.3 ตัวอย่างชุดคำสั่งโครงสร้างการทำงานแบบเรียงลำดับ

จากภาพที่ 5.3 สร้างโปรแกรมชุดคำสั่งในวัตถุ Button ในอีเวนต์ Button1_Click เพื่อให้คลิกเกิดการประมวลหาค่าเงินรับที่ได้



ภาพที่ 5.4 การทดสอบการทำงานของโปรแกรมแบบเรียงลำดับ

จากภาพที่ 5.4 แสดงผลลัพธ์ของโปรแกรม ประมวลผลการทดสอบการทำงานของโปรแกรม โดยเริ่มต้นจากการกรอกรายละเอียดข้อมูล เงินเดือนและเงินล่วงเวลา และทำการคลิกที่ปุ่มคำนวณเงินรับ โปรแกรมก็จะประมวลผลพร้อมทั้งแสดงเงินรับที่ได้

จากการทำงานข้างต้น การประมวลผลโปรแกรมจะดำเนินการทุกขั้นตอน เป็นลักษณะการทำงานแบบเรียงลำดับ โดยไม่มีการใช้คำสั่งควบคุมทิศทางการทำงานของโปรแกรม มีแต่คำสั่งในการประมวลผลทั่วไปเท่านั้น ลักษณะของการประมวลแบบนี้ ส่วนมากจะเป็นส่วนแรกการทำงานแรกของโปรแกรม ซึ่งข้อมูลที่ได้จากการประมวล จะถูกนำไปใช้ต่อตามทิศทางการทำงานของโปรแกรม ซึ่งจะกล่าวในหัวข้อถัดไป

5.2 คำสั่งควบคุมทิศทางแบบมีเงื่อนไข

โครงสร้างทิศทางแบบมีเงื่อนไข (Condition Structure) คือ โครงสร้างที่มีการตัดสินใจเลือกเส้นทางในการทำงานเส้นทางใดเส้นทางหนึ่ง (สัจจะ จรัสรุ่งรวีวร, 2554) ซึ่งโครงสร้างนี้มีความซับซ้อน ใช้ในการเลือกเส้นทางการทำงานของโปรแกรม จะทำให้สามารถเลือกเส้นทางการทำงานของโปรแกรมตามเงื่อนไขที่ต้องการ (ธีรวัฒน์ ประกอบผล, 2558) ดังนั้นโครงสร้างทิศทางแบบมีเงื่อนไข จึงมีคำสั่งที่ควบคุมทิศทางการทำงานของโปรแกรม ประกอบไปด้วยคำสั่งดังนี้

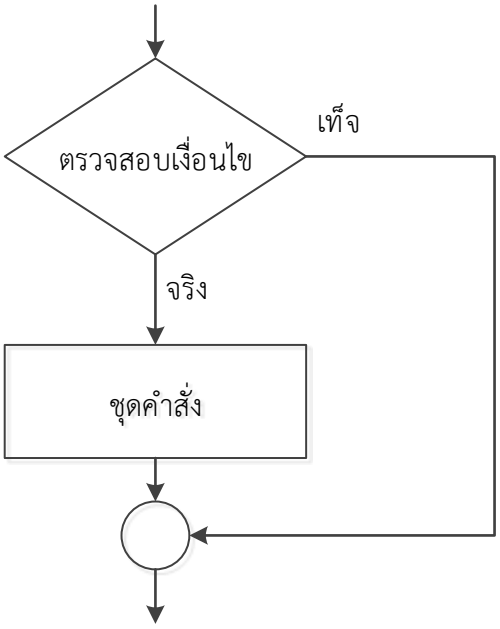
5.2.1 คำสั่ง IF

คำสั่ง If เป็นคำสั่งที่ใช้ในการตรวจสอบเงื่อนไข ที่มีการใช้งานค่อนข้างมากที่สุด เพราะสามารถตรวจสอบเงื่อนไข ได้หลายประเภท และสามารถตรวจสอบเงื่อนไขที่ซับซ้อนได้ ซึ่งประกอบได้โครงสร้างการทำงานของคำสั่งดังนี้

5.2.1.1 If ... Then (ตรวจสอบเงื่อนไขแบบ 1 เงื่อนไข) รูปแบบของคำสั่งมีหลักไวยากรณ์ดังนี้

```
If (เงื่อนไข) then
        ชุดคำสั่ง
End If
```

จากรูปแบบไวยากรณ์ของคำสั่ง If ... Then สามารถอธิบายโครงสร้างการทำงานของคำสั่งได้ดังภาพที่ 5.5



ภาพที่ 5.5 โครงสร้างการทำงานคำสั่ง If ... Then

โครงสร้างคำสั่งแบบ If ... Then มีหลักการทำงาน คือ จะมีการตรวจสอบเงื่อนไขแบบ 1 เงื่อนไข โดยถ้าเงื่อนไขที่ทำการตรวจสอบตรงกับเงื่อนไขที่กำหนดหรือเงื่อนไขที่เป็น “จริง” จะประมวลผลชุดคำสั่งที่อยู่ภายในเงื่อนไขนั้น แต่ถ้าไม่ตรงเงื่อนไขหรือเงื่อนไขที่เป็น “เท็จ จะไม่มีการประมวลผลเกิดขึ้น จากลักษณะรูปแบบการทำงานของคำสั่งดังกล่าว สามารถอธิบายได้ดังภาพที่ 5.5

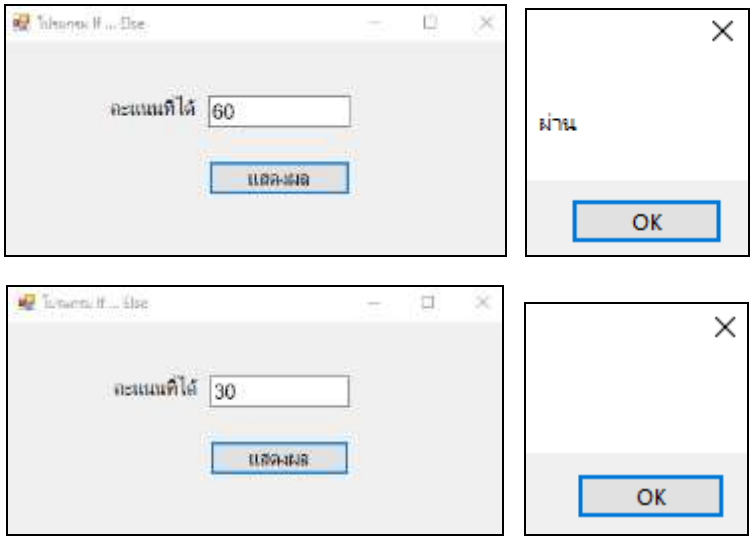
ตัวอย่างโปรแกรม ให้ตรวจสอบคะแนนกิจกรรม ถ้าคะแนนกิจกรรมมากกว่าหรือเท่ากับ 50 ให้แสดงผลลัพธ์ “ผ่าน” แต่ถ้าน้อยกว่าไม่ต้องแสดงผลลัพธ์

จากโจทย์ที่กำหนดให้ คือ ตัวอย่างการใคำสั่ง If ... Then เมื่อนำไปเขียนโปรแกรม จะได้โปรแกรมดังภาพที่ 5.6

```
Public Class Form1
    Private Sub Button1_Click(sender As Object, e As EventArgs) Handles Button1.Click
        Dim point As Integer
        Dim status As String
        point = TextBox1.Text
        If point >= 50 Then
            status = "ผ่าน"
        End If
        MessageBox.Show(status)
    End Sub
End Class
```

ภาพที่ 5.6 ตัวอย่างโปรแกรมหาการใช้คำสั่ง if ... then

จากผลลัพธ์ที่ได้ เมื่อกรอกคะแนน 60 ลงในช่องคะแนนกิจกรรม คลิกปุ่มแสดงผล โปรแกรมก็จะแสดงผลลัพธ์ “ผ่าน” แต่ถ้ากรอกคะแนนน้อยกว่า 50 จะไม่มีผลลัพธ์เกิดขึ้น ดังภาพที่ 5.7

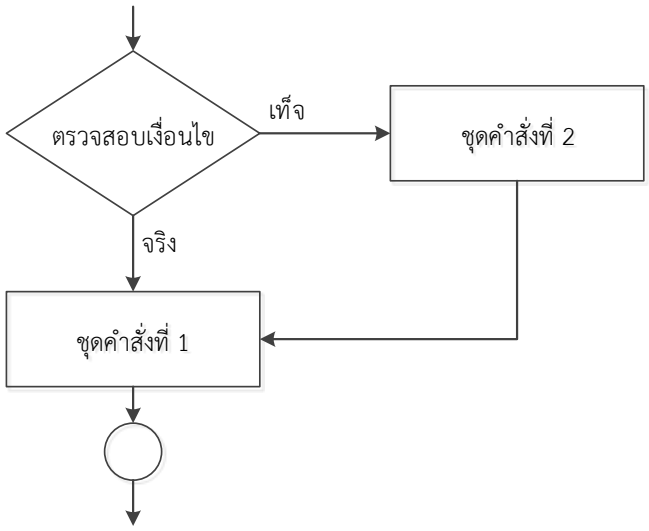


ภาพที่ 5.7 ผลลัพธ์ตัวอย่างโปรแกรมใช้คำสั่ง If ... Then

5.1.1.1 If ... then ... else (ตรวจสอบเงื่อนไขแบบ 2 เงื่อนไข) รูปแบบของคำสั่งมีหลักไวยากรณ์ดังนี้

```
If (เงื่อนไข) then
        ชุดคำสั่งที่ 1
Else
        ชุดคำสั่งที่ 2
End If
```

จากรูปแบบไวยากรณ์ของคำสั่ง If ... Then ... Else สามารถอธิบายโครงสร้างการทำงานของคำสั่งได้ดังภาพที่ 5.8



ภาพที่ 5.8 โครงสร้างการทำงานคำสั่ง If ... Then ... else

โครงสร้างคำสั่งแบบ If ... Then ... Else มีหลักการทำงาน คือ จะมีการตรวจสอบเงื่อนไขแบบ 1 เงื่อนไข โดยถ้าเงื่อนไขที่ทำการตรวจสอบตรงกับเงื่อนไขที่กำหนดหรือเงื่อนไขที่เป็นจริง จะประมวลผลชุดคำสั่งชุดที่ 1 ที่อยู่ภายในเงื่อนไขนั้น แต่ถ้าไม่ตรงเงื่อนไขหรือเงื่อนไขที่เป็นเท็จ จะประมวลผลชุดคำสั่งชุดที่ 2 จากลักษณะรูปแบบการทำงานของคำสั่งดังกล่าว สามารถอธิบายได้ดังภาพที่ 5.8

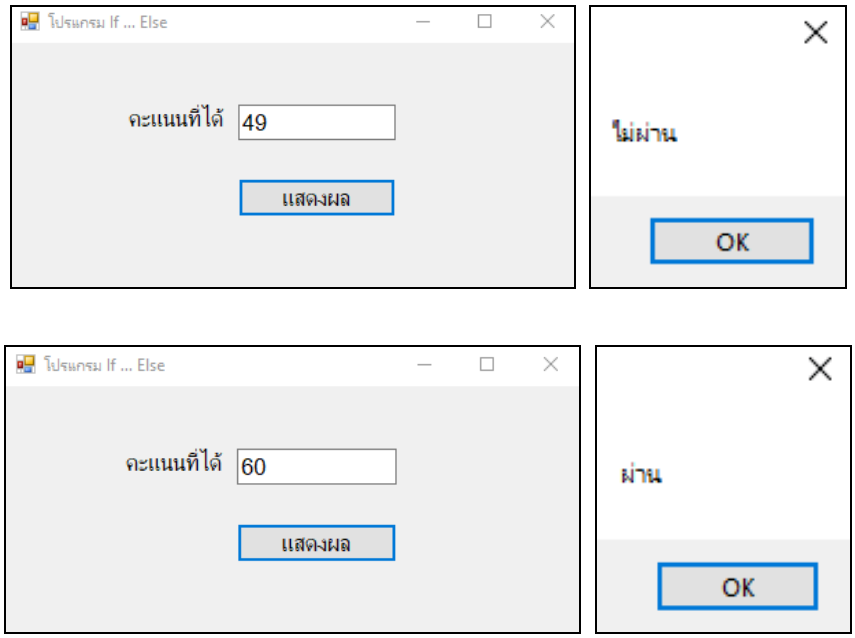
ตัวอย่างโปรแกรม ให้ตรวจสอบคะแนนกิจกรรม ถ้าคะแนนกิจกรรมมากกว่าหรือเท่ากับ 50 ให้แสดงผลลัพธ์ “ผ่าน” ถ้าน้อยกว่าแสดงผลลัพธ์ “ไม่ผ่าน”

จากโจทย์ที่กำหนดให้ คือ ตัวอย่างการใคำสั่ง If ... Then ... Else เมื่อนำไปเขียนโปรแกรมจะได้โปรแกรมดังภาพที่ 5.9

```
1  Public Class Form1
2      Private Sub Button1_Click(sender As Object, e As EventArgs) Handles Button1.Click
3          Dim point As Integer
4          Dim status As String
5          point = TextBox1.Text
6          If point >= 50 Then
7              status = "ผ่าน"
8          Else
9              status = "ไม่ผ่าน"
10         End If
11         MessageBox.Show(status)
12     End Sub
13 End Class
```

ภาพที่ 5.9 ตัวอย่างโปรแกรมหาการใช้คำสั่ง If ... then ... else

จากการทำงานของตัวอย่างโปรแกรม ผลลัพธ์ที่ได้ เมื่อดำเนินการกรอกคะแนนตัวอย่าง 49 คะแนน ลงในช่องคะแนนกิจกรรม คลิกปุ่มแสดงผล โปรแกรมก็จะแสดงผลลัพธ์ “ไม่ผ่าน” แต่กรอกคะแนน 60 จะแสดงผลลัพธ์ “ผ่าน” ตามเงื่อนไขที่กำหนดไว้ในเบื้องต้น ซึ่งการทำงานของโปรแกรมดังภาพที่ 5.10



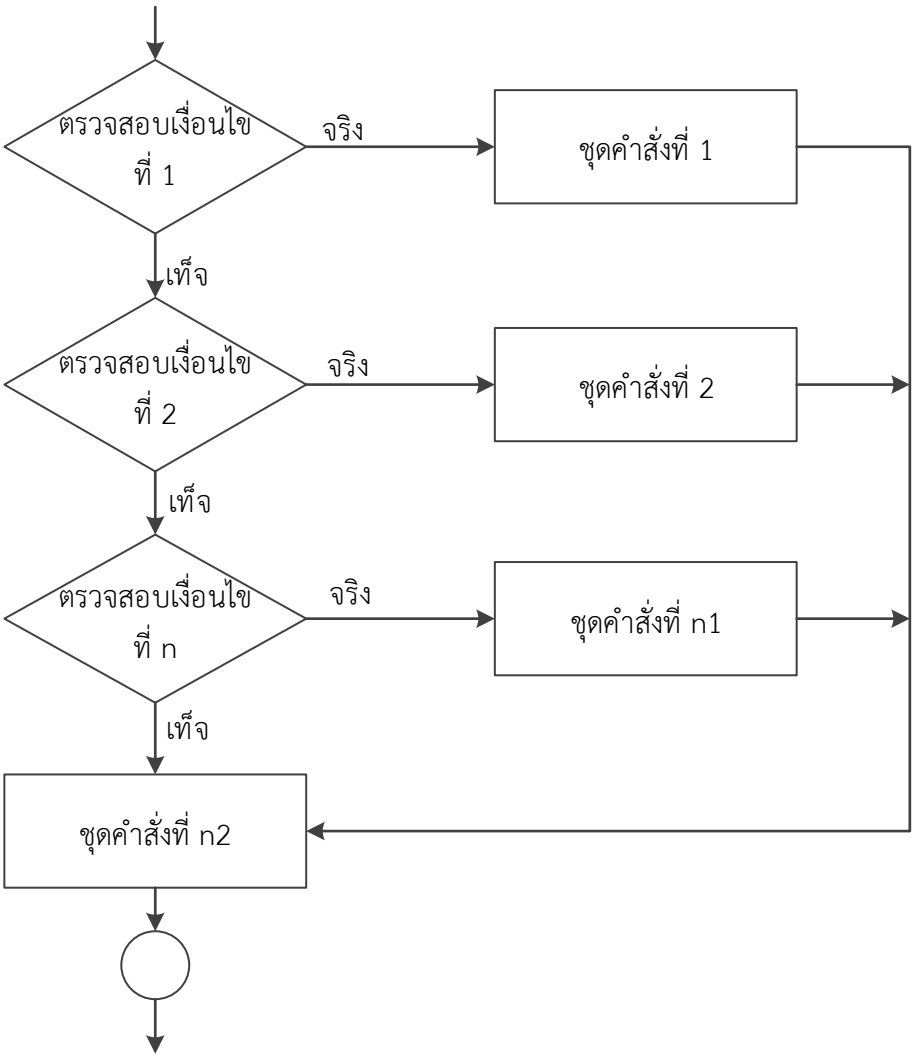
ภาพที่ 5.10 ผลลัพธ์ตัวอย่างโปรแกรมใช้คำสั่ง If ... Then ... Else

การตรวจสอบเส้นทางการทำงานแบบ สองเงื่อนไข คือ จริงและเท็จ ถือว่าเป็นโครงสร้างหลักของการทำงานของโปรแกรม ซึ่งนอกเหนือแบบหนึ่งเงื่อนไข การทำงานแบบวนซ้ำจะมีการกำหนดเงื่อนไขการทำงานแบบ 2 เงื่อนไขเช่นกัน

5.1.1.2 If ... Then ... ElseIf ... Else (แบบมากกว่า 2 เงื่อนไข) รูปแบบของคำสั่งมีหลักไวยากรณ์ดังนี้

```
If (เงื่อนไขที่ 1) Then  ชุดคำสั่งที่ 1
ElseIf (เงื่อนไขที่ 2) Then ชุดคำสั่งที่ 2
        •
        •
ElseIf (เงื่อนไขที่ n) Then ชุดคำสั่งที่ n1
Else
        ชุดคำสั่ง n2
```

จากรูปแบบไวยากรณ์ของคำสั่ง If ... Then ... ElseIf.....Else สามารถอธิบายโครงสร้างการทำงานของคำสั่งได้ดังภาพที่ 5.11



ภาพที่ 5.11 โครงสร้างการทำงานของคำสั่ง If ... Then ... ElseIf.....Else

โครงสร้างคำสั่งแบบ If ... Then ... ElseIf ... Else มีหลักการทำงาน คือ จะมีการตรวจสอบเงื่อนไขแรก โดยถ้าเงื่อนไขที่ทำการตรวจสอบตรงกับเงื่อนไขที่กำหนดหรือเงื่อนไขที่เป็นจริง จะประมวลผลชุดคำสั่งชุดที่ 1 ที่อยู่ภายในเงื่อนไขนั้น แต่ถ้าไม่ตรงเงื่อนไขหรือเงื่อนไขที่เป็นเท็จ จะทำการตรวจสอบเงื่อนไขที่ 2 และทำการตรวจสอบเงื่อนไข ถ้าตรงกับเงื่อนไขหรือเงื่อนไขที่เป็นจริง จะประมวลผลชุดคำสั่งชุดที่ 2 แต่ถ้าไม่ตรงเงื่อนไขหรือเงื่อนไขที่เป็นเท็จ จะมีการตรวจสอบเงื่อนไขที่อยู่ถัดไปเรื่อย ๆ จนกระทั่งถึงเงื่อนไขสุดท้าย รูปแบบการใช้คำสั่งนี้ จะมีการนำไปใช้มาก ใช้สำหรับในการตรวจสอบเงื่อนไขที่มีจำนวนมาก จากลักษณะรูปแบบการทำงานของคำสั่งดังกล่าว สามารถอธิบายได้ดังภาพที่ 5.11

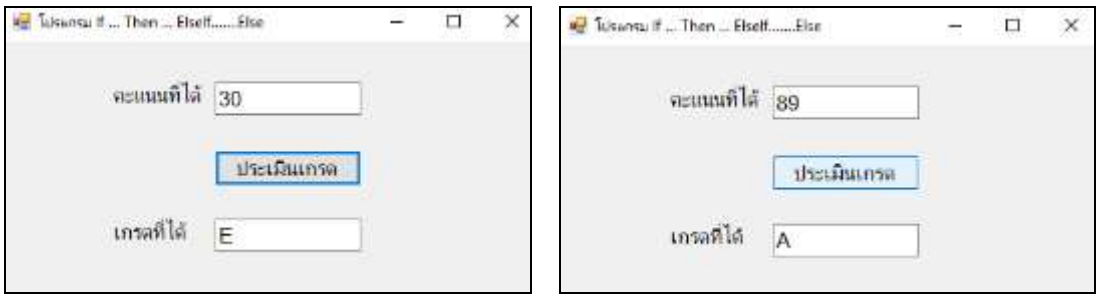
ตัวอย่างโปรแกรม ให้แสดงผลเกรดที่ได้ โดยมีเงื่อนไข แสดงเกรด “A” ถ้าคะแนน 80 คะแนนขึ้นไป แสดงเกรด “B” ถ้าคะแนน 70 คะแนนขึ้นไปแต่ไม่ถึง 80 คะแนน แสดงเกรด “C” ถ้าคะแนน 60 คะแนนขึ้นไปแต่ไม่ถึง 70 คะแนน แสดงเกรด “D” ถ้าคะแนน 50 คะแนนขึ้นไปแต่ไม่ถึง 60 คะแนน และแสดงเกรด “E” ถ้าคะแนนต่ำกว่า 50 คะแนน

จากโจทย์ที่กำหนดให้ คือ ตัวอย่างการใช้คำสั่ง If ... Then ... ElseIf.....Else เมื่อนำไปเขียนโปรแกรม จะได้โปรแกรมดังภาพที่ 5.12

```
Public Class Form1
    Private Sub Button1_Click(sender As Object, e As EventArgs) Handles Button1.Click
        Dim point As Integer
        point = TextBox1.Text
        If point >= 80 Then
            TextBox2.Text = "A"
        ElseIf point >= 70 Then
            TextBox2.Text = "B"
        ElseIf point >= 60 Then
            TextBox2.Text = "C"
        ElseIf point >= 50 Then
            TextBox2.Text = "D"
        Else
            TextBox2.Text = "E"
        End If
    End Sub
End Class
```

ภาพที่ 5.12 ตัวอย่างโปรแกรมหาการใช้คำสั่ง If ... Then ... ElseIf.....Else

จากการทำงานของโปรแกรม ผลลัพธ์ที่ได้ เมื่อกรอกคะแนนตัวอย่าง เช่น 30 ลงในช่องคะแนนที่ได้ คลิกปุ่มประเมินผล โปรแกรมก็จะแสดงผลลัพธ์เกรดที่ได้ “E” ซึ่งเข้ากับเงื่อนไขถ้าคะแนนน้อยกว่า 50 คะแนน แสดงเกรด “A” และเมื่อกรอกคะแนน 89 จะแสดงผลลัพธ์เกรดที่ได้ “A” ซึ่งเข้ากับเงื่อนไขถ้าคะแนน 80 คะแนนขึ้นไป แสดงเกรด “A”ดังภาพที่ 5.13

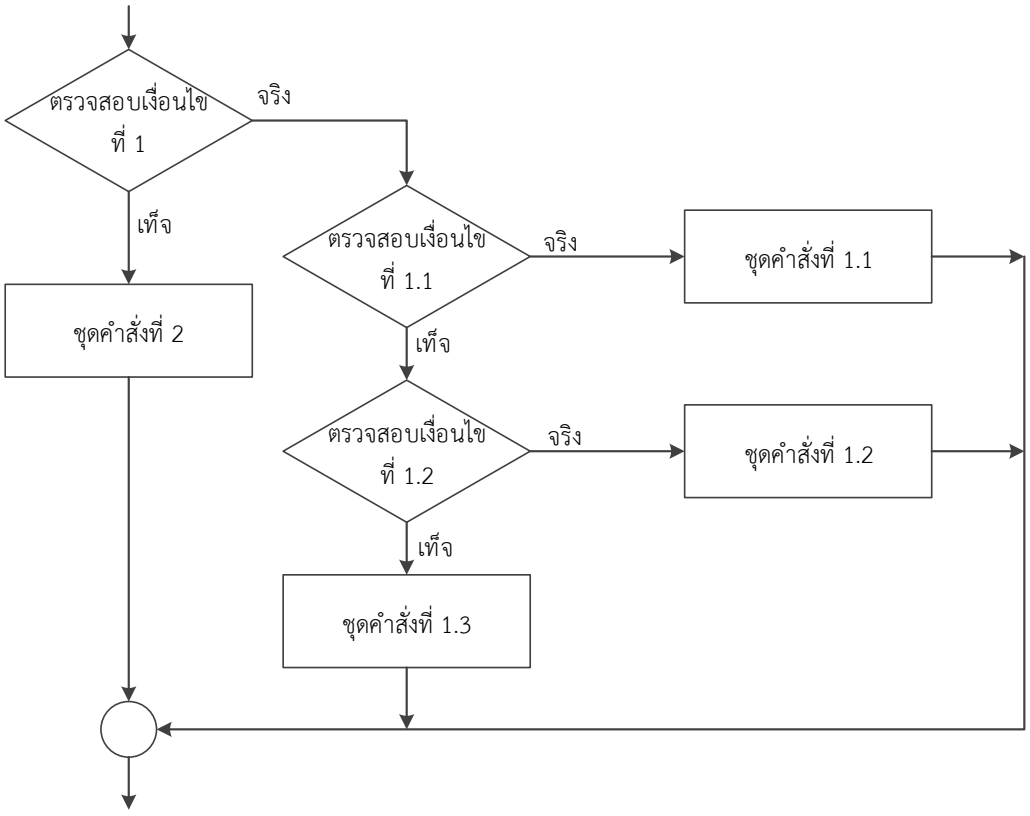


ภาพที่ 5.13 ผลลัพธ์ตัวอย่างโปรแกรมใช้คำสั่ง If ... Then ... ElseIf.....Else

5.1.1.3 nested if (แบบซ้อน) โครงสร้างคำสั่งนี้ รูปแบบของคำสั่งมีหลักไวยากรณ์ดังนี้

```
If (เงื่อนไขที่ 1) then
        (เงื่อนไขที่ 1.1 ) then ชุดคำสั่งที่ 1.1
        (เงื่อนไขที่ 1.2 ) then ชุดคำสั่งที่ 1.2
                •
                •
                •
                •
        (เงื่อนไขที่ n ) then ชุดคำสั่งที่ n
Else
        ชุดคำสั่ง 2
```

จากรูปแบบไวยากรณ์ของคำสั่ง If ... Then ... ElseIf.....Else ซึ่งเป็นคำสั่งในการตรวจสอบเงื่อนไขที่มีจำนวนมากกว่าคำสั่ง If ... Then และคำสั่ง If ... Else สามารถอธิบายโครงสร้างการทำงานของคำสั่งได้ดังภาพที่ 5.14



ภาพที่ 5.14 โครงสร้างการทำงานของคำสั่ง Nested If

โครงสร้างคำสั่งแบบ Netted If มีหลักการทำงาน คือ จะมีการตรวจสอบเงื่อนไขที่ 1 โดยถ้าเงื่อนไขที่ทำการตรวจสอบตรงกับเงื่อนไขที่กำหนดหรือเงื่อนไขที่เป็นจริง จะมีการตรวจสอบเงื่อนที่อยู่ภายในอีกครั้ง ถ้าทำการตรวจสอบเงื่อนไขที่ 1.1 เป็นจริงให้ประมวลผลชุดคำสั่งที่ 1.1 แต่ถ้าไม่ตรงกับเงื่อนไขหรือเงื่อนไขที่เป็นเท็จ จะทำการตรวจสอบเงื่อนไขที่ 1.2 ภายในอีกครั้ง ซึ่งจะทำการตรวจสอบเงื่อนไขไปเรื่อย ๆ จนสิ้นสุด จึงจะหลุดจากเงื่อนไขที่ซ้อน รูปแบบการใช้คำสั่งนี้ มีความซับซ้อนมาก การตรวจสอบเงื่อนไขอาจมีตั้งแต่ 2 ชั้นขึ้นไป จากลักษณะรูปแบบการทำงานของคำสั่งดังกล่าว สามารถอธิบายได้ดังภาพที่ 5.14

ตัวอย่างโปรแกรม หาเงินจ่ายค่าสินค้า โดยถ้าเป็นสมาชิกของร้าน เมื่อซื้อสินค้า 10,000 บาท ขึ้นไป ได้รับสดลดที่ 10% ถ้าซื้อสินค้า 5,000 บาท แต่ไม่ถึง 10,000 ได้รับส่วนลดที่ 5% ถ้าซื้อสินค้า 1,000 บาท แต่ไม่ถึง 5,000 บาท ได้รับส่วนลดที่ 3% และซื้อสินค้าต่ำกว่า 1,000 บาท ได้รับส่วนลด ที่ 2%

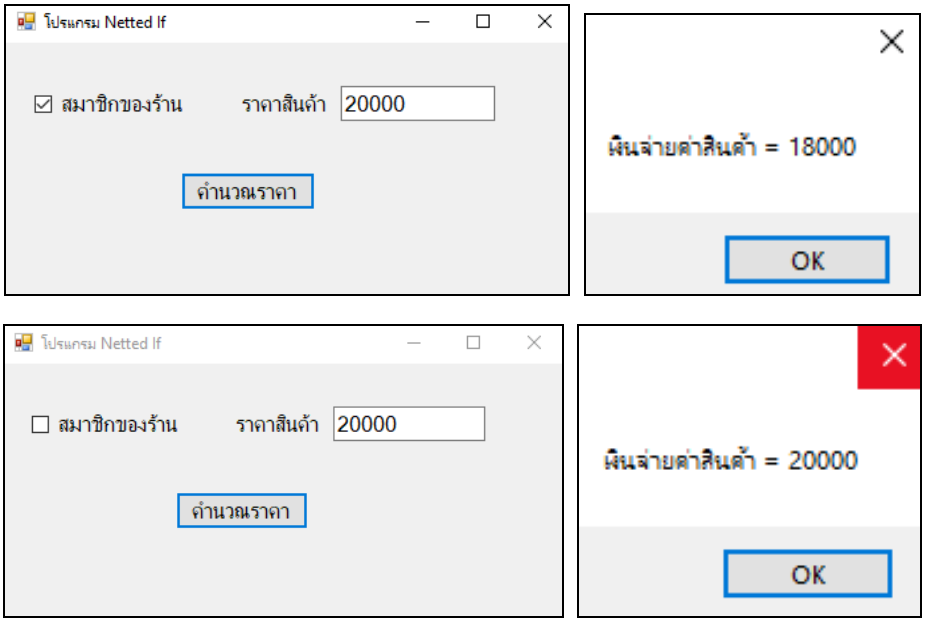
จากโจทย์ที่กำหนดให้ คือ ตัวอย่างการใช้คำสั่ง Netted If เมื่อนำไปเขียนโปรแกรม จะได้โปรแกรมดังภาพที่ 5.15

```
1  Public Class Form1
2      Private Sub Button1_Click(sender As Object, e As EventArgs) Handles Button1.Click
3          Dim price, discount, money As Double
4          price = TextBox1.Text
5          If CheckBox1.Checked = True Then
6              If price >= 10000 Then
7                  discount = 10
8              ElseIf price >= 5000 Then
9                  discount = 5
10             ElseIf price >= 1000 Then
11                 discount = 3
12             Else
13                 discount = 2
14             End If
15         End If
16         money = price - (price * discount / 100)
17         MessageBox.Show("เงินจ่ายค่าสินค้า = " & money)
18     End Sub
19 End Class
```

ภาพที่ 5.15 ตัวอย่างโปรแกรมหาการใช้คำสั่ง Netted If

จากการทำงานของตัวอย่างโปรแกรม ผลลัพธ์ที่ได้ ในกรณีที่ 1 เมื่อทำการกำหนดสถานะเป็นสมาชิกของร้าน และกรอกราคาสินค้า 20,000 บาท โปรแกรมจะทำการประมวลผลโดยเงื่อนไขเป็นสมาชิก และมีราคาสินค้ามากว่า 10,000 บาท จึงได้รับส่วนลด 10% ดังนั้นจึงแสดงผลลัพธ์เงินจ่ายค่าสินค้า 18,000 และในกรณีที่ 2 ไม่ได้กำหนดสถานะเป็นสมาชิกของร้าน และกรอกราคาสิค้า 20,000

บาท ผลลัพธ์ของเงินจ่ายค่าสินค้าคือ 20,000 เหมือนเดิม เนื่องจากไม่ได้เป็นไปตามเงื่อนไขที่เป็นสมาชิก ดังนั้นจึงไม่ได้รับส่วนลด ผลลัพธ์ของโปรแกรมดังภาพที่ 5.16



ภาพที่ 5.16 ผลลัพธ์ชุดคำสั่งแบบ Netted If

คำสั่ง If มีรูปแบบของการนำไปใช้งานหลายรูปแบบ ซึ่งมีตั้งแต่ 1 เงื่อนไข 2 เงื่อนไข และมากกว่า 2 เงื่อนไขขึ้นไป พร้อมทั้งมีรูปแบบพิเศษคือการซ้อนกันของคำสั่ง ดังนั้นการนำไปใช้ก็ต้องพิจารณาจำนวน และลักษณะของเงื่อนไข เพื่อนำไปใช้ให้ถูกต้องและเหมาะสมต่อไป

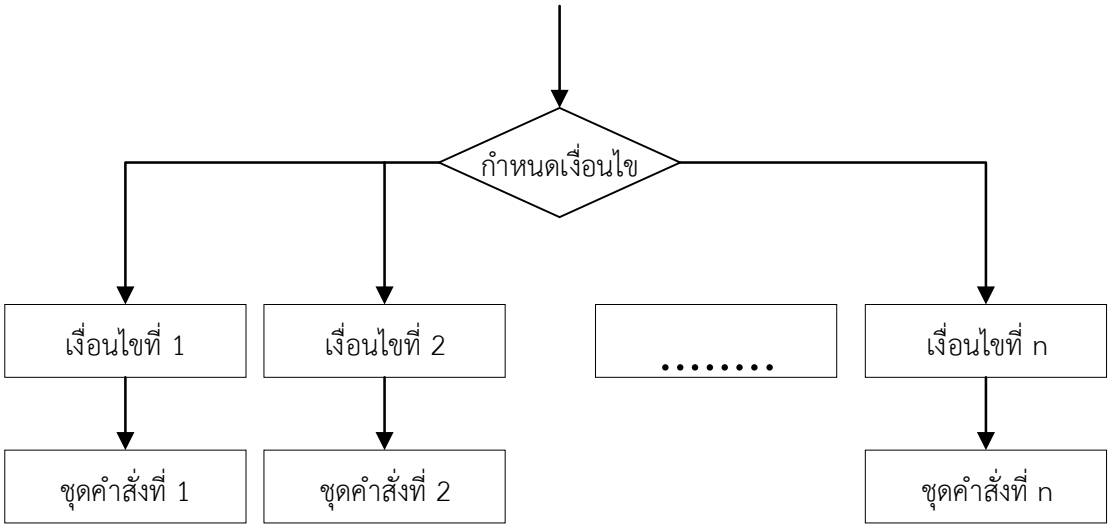
5.1.1 select ... case

คำสั่งที่ใช้ในการควบคุมทิศทางการทำงานของโปรแกรม select ... case เป็นรูปแบบคำสั่งอีกประเภทหนึ่ง ที่เป็นลักษณะของคำสั่งแบบมีเงื่อนไข การทำงานคล้ายกับ if แต่จะมีรูปแบบของการเขียนง่ายกว่า เหมาะสมกับโครงสร้างของโปรแกรมที่ไม่เงื่อนไขจำนวนมาก แต่เงื่อนไขไม่ซับซ้อนมาก ซึ่งรูปแบบไวยากรณ์ของคำสั่งดังนี้

```
Select Case ข้อมูล
        Case เงื่อนไขที่ 1
ชุดคำสั่งที่ 1
        Case เงื่อนไขที่ 2
ชุดคำสั่งที่ 2
        •
        •
```

```
Case เงื่อนไขที่ n          หรือ     Case Else
    ชุดคำสั่งที่ n                       ชุดคำสั่งสุดท้าย
```

จากรูปแบบไวยากรณ์ของคำสั่ง Select Case ... สามารถอธิบายโครงสร้างการทำงานของคำสั่งได้ดังภาพที่ 5.17

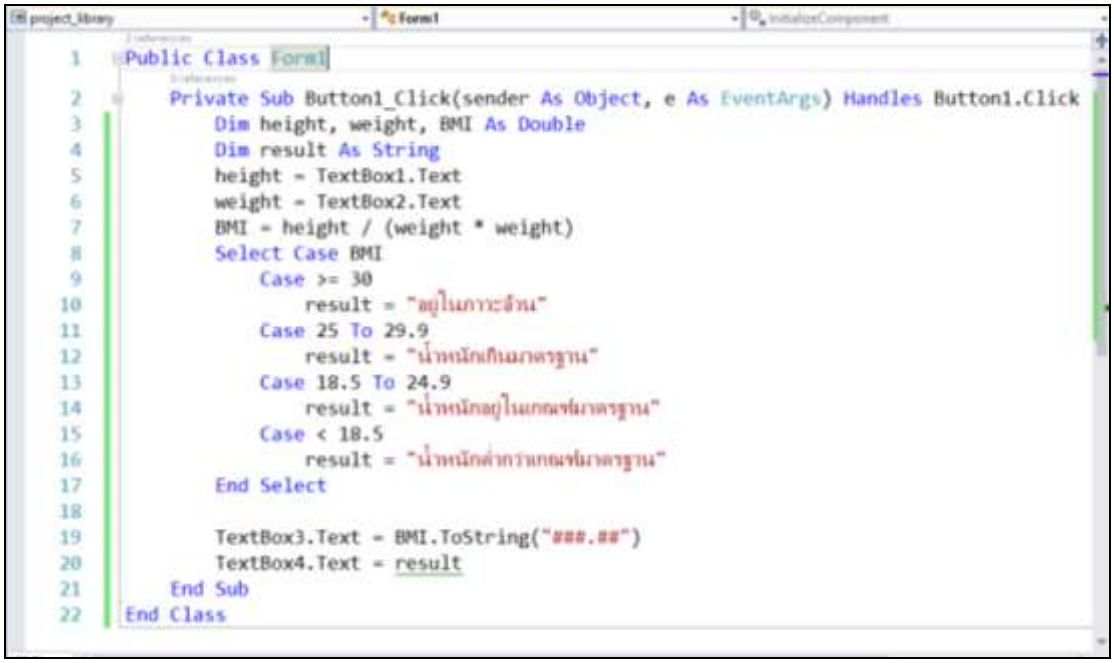


ภาพที่ 5.17 โครงสร้างการทำงานของคำสั่ง Select Case

โครงสร้างคำสั่งแบบ Select Case ... มีหลักการทำงาน คือ จะมีการตรวจสอบเงื่อนไขที่กำหนดที่ละเงื่อนไข ถ้าเงื่อนไขที่ทำการตรวจสอบตรงกับเงื่อนไขที่กำหนดหรือเงื่อนไขที่เป็นจริง เงื่อนไขที่ 1 จะทำการประมวลผลชุดคำสั่งที่ 1 หากไม่ตรงกับเงื่อนไขหรือเงื่อนไขที่เป็นเท็จ จะข้ามไปตรวจสอบเงื่อนไขที่ 2 ถ้าเงื่อนไขที่ทำการตรวจสอบตรงกับเงื่อนไขที่กำหนด จะทำการประมวลผลชุดคำสั่งที่ 2 หากไม่ตรงกับเงื่อนไข จะข้ามไปตรวจสอบเงื่อนไขที่อยู่ถัดไปเรื่อย ๆ ซึ่งลักษณะโครงสร้างการทำงานของคำสั่งสามารถอธิบายได้ดังภาพที่ 5.17

ตัวอย่างโปรแกรม ให้แปลผลจากค่าดัชนีมวลกาย (BMI) โดยค่าดัชนีมวลกาย 30 ขึ้นไป อยู่ในภาวะอ้วน ดัชนีมวลกาย 25-29.9 น้ำหนักเกินมาตรฐาน ดัชนีมวลกาย 18.5-24.9 น้ำหนักอยู่ในเกณฑ์มาตรฐาน และน้ำหนักน้อยกว่า 18.5 น้ำหนักต่ำกว่าเกณฑ์มาตรฐาน

จากโจทย์ที่กำหนดให้ คือ ตัวอย่างการใช้คำสั่ง Select Case ... เมื่อนำไปเขียนโปรแกรมจะได้โปรแกรมดังภาพที่ 5.18

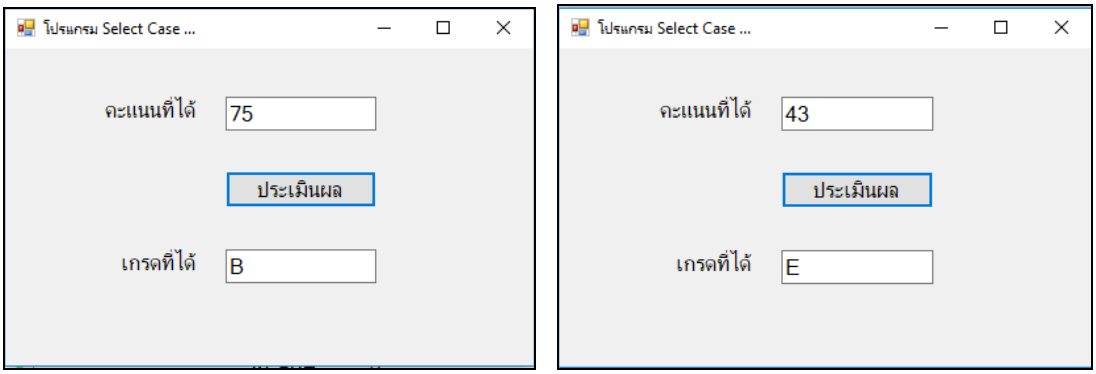


```
Public Class Form1
    Private Sub Button1_Click(sender As Object, e As EventArgs) Handles Button1.Click
        Dim height, weight, BMI As Double
        Dim result As String
        height = TextBox1.Text
        weight = TextBox2.Text
        BMI = height / (weight * weight)
        Select Case BMI
            Case >= 30
                result = "อยู่ในภาวะอ้วน"
            Case 25 To 29.9
                result = "น้ำหนักเกินมาตรฐาน"
            Case 18.5 To 24.9
                result = "น้ำหนักอยู่ในเกณฑ์มาตรฐาน"
            Case < 18.5
                result = "น้ำหนักต่ำกว่าเกณฑ์มาตรฐาน"
        End Select

        TextBox3.Text = BMI.ToString("###.##")
        TextBox4.Text = result
    End Sub
End Class
```

ภาพที่ 5.18 ตัวอย่างโปรแกรมหาการใช้คำสั่ง select ... case

จากภาพที่ 5.18 ลักษณะของใช้โครงสร้างของโปรแกรมแบบมีเงื่อนไข ที่มีจำนวนเงื่อนไขจำนวน 4 เงื่อนไข ซึ่งสังเกตได้ว่ารูปแบบของการใช้คำสั่งจะสั้นกว่าคำสั่ง if และรูปแบบการเขียนไม่ซับซ้อนมากจนเกินไป ถ้ามีจำนวนเงื่อนไขเป็นจำนวนมาก จะสามารถตรวจสอบคำสั่งได้สะดวกและง่ายกว่าคำสั่ง if แต่ด้วยความง่ายจึงไม่เหมาะสมกับการใช้กับโครงสร้างของโปรแกรมที่มีความซับซ้อนมาก ๆ ซึ่งผลลัพธ์จากการประมวลผลโปรแกรมดังภาพที่ 5.19



ภาพที่ 5.19 ผลลัพธ์ชุดคำสั่งแบบ Select Case

คำสั่งควบคุมทิศทางการทำงานของโปรแกรมแบบมีเงื่อนไข ในโปรแกรมภาษาวิชวลเบสิกประกอบไปด้วยคำสั่ง If ซึ่งจะมีรูปแบบการใช้คำสั่งหลายประเภท และคำสั่ง Select Case ซึ่งผู้เขียนโปรแกรมจะต้องพิจารณาเลือกใช้ให้เหมาะสม

5.2 คำสั่งควบคุมทิศทางแบบทำซ้ำ

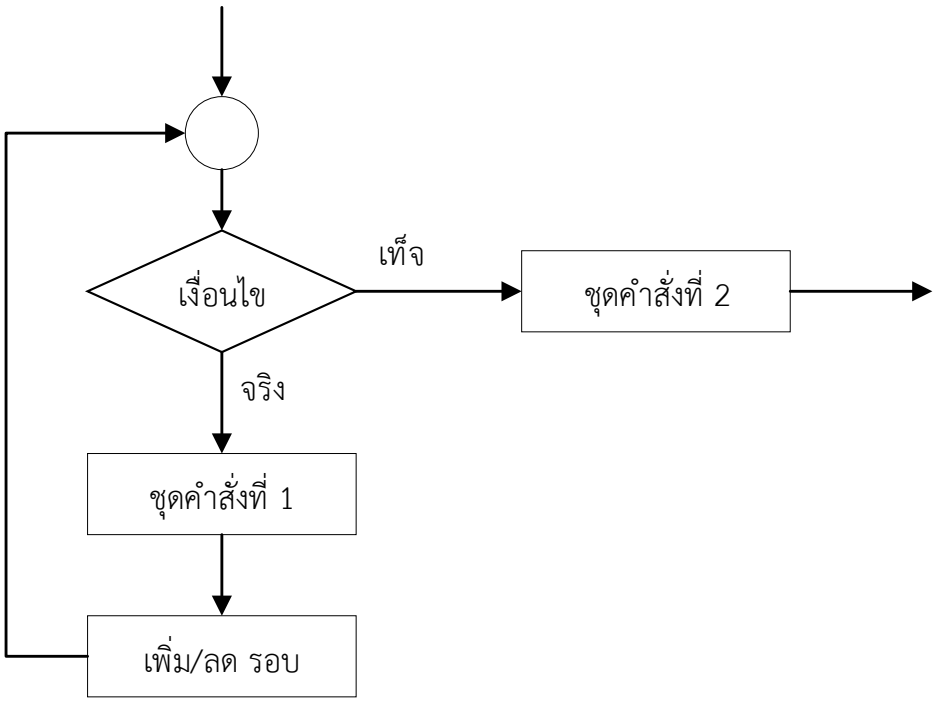
โครงสร้างแบบวนซ้ำ (Loop Structure) คือ การทำงานแบบซ้ำไปซ้ำมาตามเงื่อนไขที่ได้กำหนดไว้ โดยใช้ตัวดำเนินการแบบต่าง ๆ มาเป็นตัวช่วยในการกำหนดเงื่อนไข (กิตินันท์ พลสวัสดิ์, 2554) การควบคุมทิศทางการทำงานแบบทำซ้ำถือว่ามีความสำคัญมากอีกโครงสร้าง ซึ่งในโปรแกรมภาษาวิชวลเบสิก มีคำสั่งที่ใช้ทำงานดังต่อไปนี้

5.2.1 For

คำสั่ง For คือ คำสั่งทำซ้ำ ที่ใช้หลักการในการนับรอบการทำงาน มีจำนวนครั้งที่แน่นอน โดยมีโครงสร้างของคำสั่งดังต่อไปนี้

```
For  ตัวแปรนับรอบ = ค่าเริ่มต้น To ค่าสุดท้าย
     ชุดคำสั่ง
Next
```

จากรูปแบบไวยากรณ์ของคำสั่ง For สามารถอธิบายโครงสร้างการทำงานของคำสั่งได้ดังภาพที่ 5.20



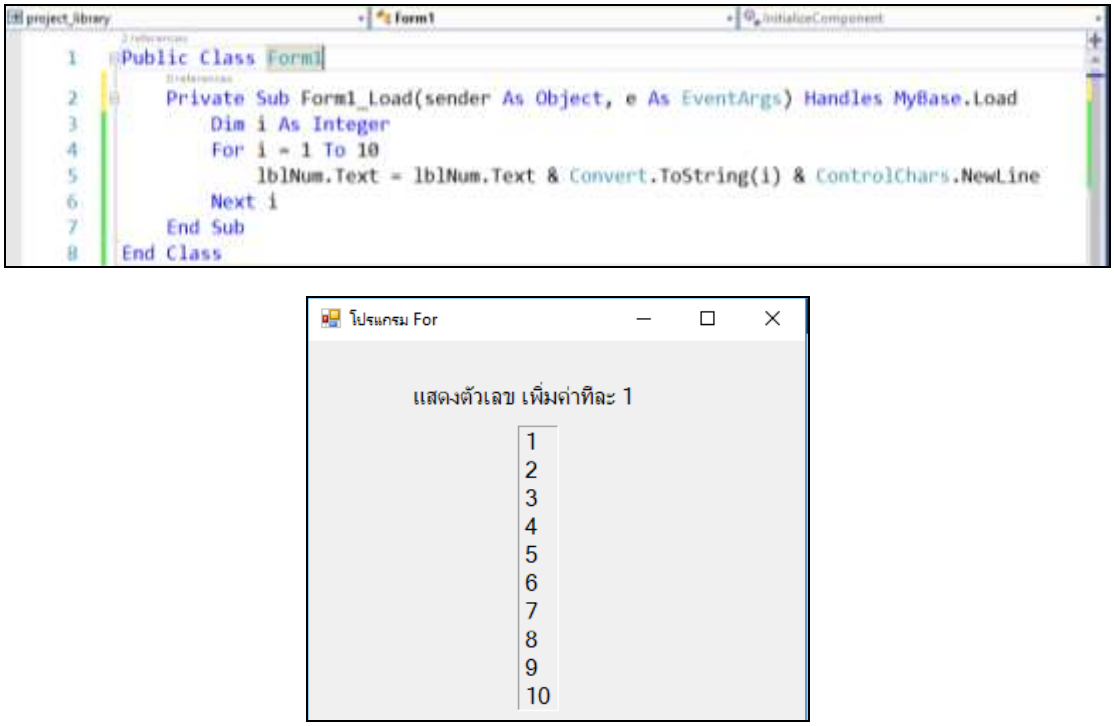
ภาพที่ 5.20 โครงสร้างการทำงานของคำสั่ง For

คำสั่ง For จะใช้วิธีการในการนับรอบการทำงานของโปรแกรม เมื่อตรวจสอบเงื่อนไขตรงกับเงื่อนไขหรือเงื่อนไขที่เป็นจริง จะทำชุดคำสั่งชุดที่ 1 และตรวจสอบจำนวนรอบอาจได้จากการลดรอบหรือเพิ่มรอบ และกลับไปตรวจสอบเงื่อนไขอีกครั้ง จะดำเนินการทำซ้ำไปเรื่อย ๆ จนสิ้นสุดรอบการทำงานหรือเงื่อนไขเป็นเท็จ จะหยุดทำซ้ำและประมวลผลคำสั่งชุดที่ 2 ดังภาพที่ 5.16 และการใช้คำสั่ง For มีรูปแบบการใช้คำสั่งดังนี้

5.2.1.1 คำสั่ง For เพิ่มค่าทีละ 1 คือ การนับรอบการทำงานโดยให้มีการเพิ่มค่าที่ละ 1 โดยมีรูปแบบไวยากรณ์ดังนี้

```
For ตัวแปร = ค่าเริ่มต้น To ค่าสุดท้าย
    ชุดคำสั่ง
Next
```

ตัวอย่างการใช้คำสั่ง เพิ่มค่าทีละ 1 โดยกำหนดค่าเริ่มต้นเป็น 1 ทำการเพิ่มค่าที่ละ 1 เมื่อนำไปเขียนโปรแกรมและประมวลผลลัพธ์ ได้ดังภาพที่ 5.21

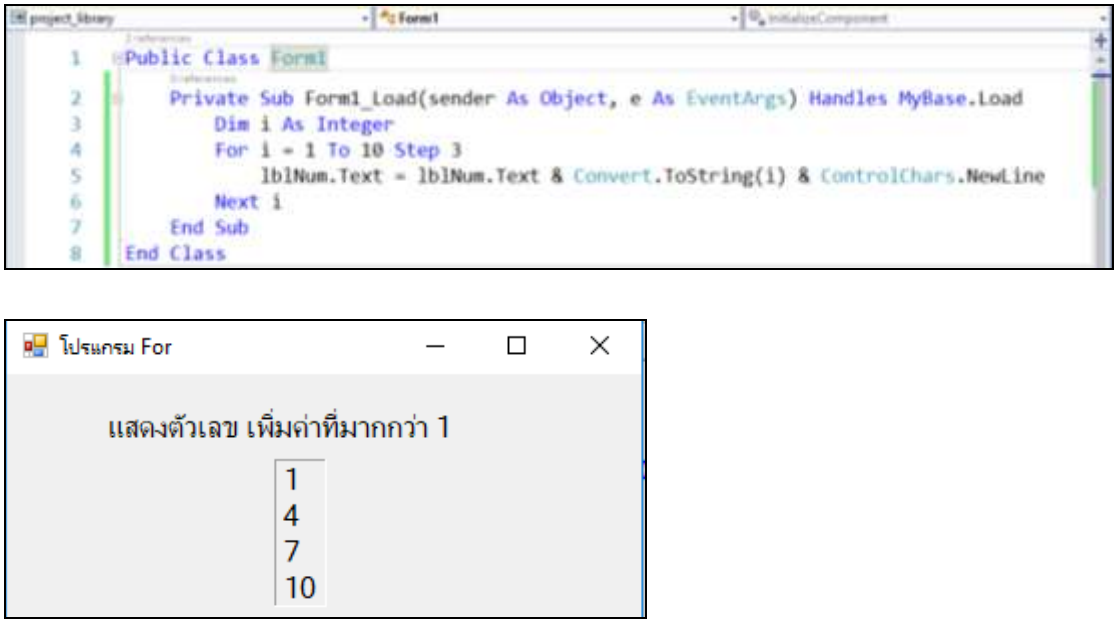


ภาพที่ 5.21 โครงสร้างการทำงานของคำสั่ง For เพิ่มค่าทีละ 1

5.2.1.2 คำสั่ง For เพิ่มค่าที่มากกว่า 1 คือ การนับรอบการทำงานโดยให้มีการเพิ่มค่าตามจำนวนที่กำหนด โดยมีรูปแบบไวยากรณ์ดังนี้

```
For  ตัวแปร = ค่าเริ่มต้น To ค่าสุดท้าย Step จำนวนค่าที่เพิ่ม
     ชุดคำสั่ง
Next
```

ตัวอย่างการใช้คำสั่ง เพิ่มค่าทีละ 1 โดยกำหนดค่าเริ่มต้นเป็น 1 ทำการเพิ่มค่าที่ละ 3 เมื่อนำไปเขียนโปรแกรมและประมวลผลลัพธ์ ได้ดังภาพที่ 5.22



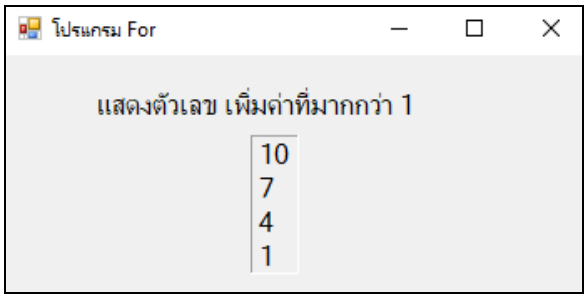
ภาพที่ 5.22 โครงสร้างการทำงานของคำสั่ง For เพิ่มค่าที่มากกว่า 1

5.2.1.3 คำสั่ง For ลดค่า คือ การนับรอบการทำงานโดยให้มีการลดค่าตามจำนวนที่ต้องการ โดยมีรูปแบบไวยากรณ์ดังนี้

```
For  ตัวแปร = ค่าเริ่มต้น To ค่าสุดท้าย Step -จำนวนค่าที่ลด
     ชุดคำสั่ง
Next
```

ตัวอย่างโปรแกรม การใช้คำสั่งลดค่า โดยกำหนดค่าเริ่มต้นเป็น 1 ทำการลดค่าที่ละ 3 เมื่อนำไปเขียนโปรแกรมและประมวลผลลัพธ์ ได้ดังภาพที่ 5.23

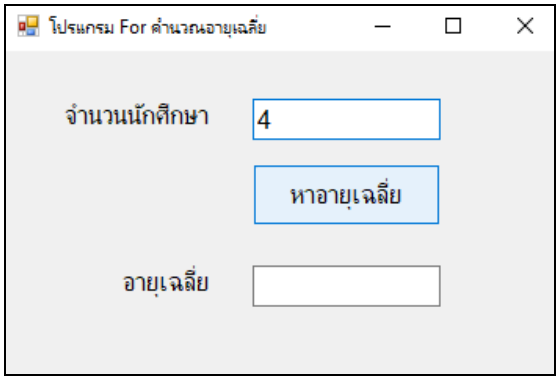
```vb
Public Class Form1
    Private Sub Form1_Load(sender As Object, e As EventArgs) Handles MyBase.Load
        Dim i As Integer
        For i = 10 To 1 Step -3
            lblNum.Text = lblNum.Text & Convert.ToString(i) & ControlChars.NewLine
        Next i
    End Sub
End Class
```

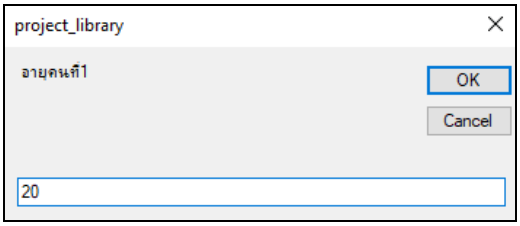


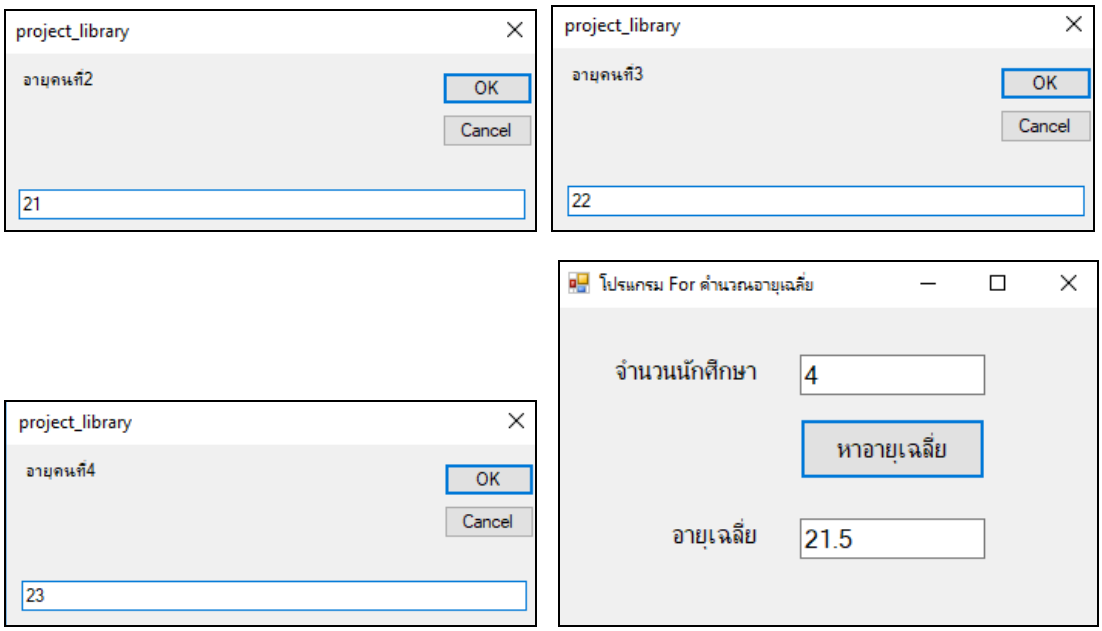
ภาพที่ 5.23 โครงสร้างการทำงานของคำสั่ง For ลดค่า

ตัวอย่างที่ 5.6 การประยุกต์รูปแบบการใช้คำสั่ง For กับตัวอย่าง ให้หาอายุเฉลี่ยของนักศึกษาตามจำนวนศึกษาที่กำหนด ซึ่งสามารถเขียนโปรแกรมและประมวลผลลัพธ์ลำดับขั้นตอนการทำงาน ได้ดังภาพที่ 5.24

```vb
Public Class Form1
    Private Sub Button1_Click(sender As Object, e As EventArgs) Handles Button1.Click
        Dim num, i As Integer
        Dim age, total, aver As Double
        num = TextBox1.Text
        For i = 1 To num
            age = InputBox("อายุคนที่" & i)
            total = total + age
        Next
        aver = total / num
        TextBox2.Text = aver
    End Sub
End Class
```







ภาพที่ 5.24 ตัวอย่างโปรแกรมหาอายุเฉลี่ยตามจำนวนนักศึกษาโดยคำสั่ง For

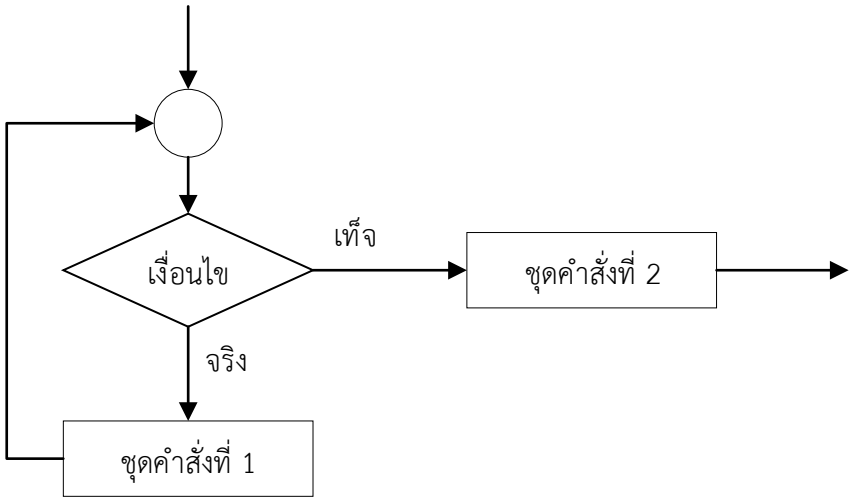
การนับรอบการทำงาน สามารถประยุกต์โดยให้สามารถรับค่าข้อมูลจากโปรแกรมได้ โดยไม่ต้องกำหนดเป็นค่าคงที่ในโปรแกรม ซึ่งจะทำให้โปรแกรมมีความยืดหยุ่นในการทำงาน เปลี่ยนแปลงค่าในโปรแกรมได้ การทำงานเช่นดังภาพที่ 5.24

5.2.2 Do While ... Loop

คำสั่ง While คือ คำสั่งที่จะทำการตรวจสอบเงื่อนไขก่อนการทำงานซ้ำ แตกต่างจากคำสั่ง For ที่จะต้องกำหนดค่าเริ่มต้นในคำสั่ง แต่คำสั่ง while จะไม่มีการกำหนดค่าเริ่มต้นแต่จะทำการตรวจสอบเงื่อนไขจากข้อมูล ซึ่งมีรูปแบบของไวยากรณ์ดังต่อไปนี้

```
Do While (เงื่อนไข)
    ชุดคำสั่ง (เมื่อเงื่อนไขเป็นจริง)
Loop
```

5.2.3 จากรูปแบบไวยากรณ์ของคำสั่ง Do While ... Loop สามารถอธิบายโครงสร้างการทำงานของคำสั่งได้ดัง ภาพที่ 5.25



ภาพที่ 5.25 โครงสร้างการทำงานของคำสั่ง Do While ... Loop

คำสั่ง Do While ... Loop จะไม่มีการเพิ่มค่าหรือนับค่านับรอบการทำงาน เมื่อมีการตรวจสอบเงื่อนไขเป็นจริงจะประมวลผลชุดคำสั่งที่ 1 และเมื่อเสร็จสิ้นการการประมวลผล จะกลับไปตรวจสอบเงื่อนไขซ้ำอีกครั้ง ถ้าเงื่อนไขเป็นจริงก็จะประมวลผลชุดคำสั่งที่ 1 ไปเรื่อย ๆ จนกว่าเงื่อนไขจะเป็นเท็จ จึงไปประมวลผลชุดคำสั่งที่ 2 ดังภาพที่ 5.25

จากตัวอย่างโปรแกรม ให้หาอายุเฉลี่ยของนักศึกษาตามจำนวนศึกษาที่กำหนด (โจทย์คำสั่ง For) สามารถนำมาเขียนโปรแกรม โดยใช้คำสั่ง Do While ... Loop ได้ดังภาพที่ 5.26

```
1  Public Class Form1
2      Private Sub Button1_Click(sender As Object, e As EventArgs) Handles Button1.Click
3          Dim num, i As Integer
4          Dim age, total, aver As Double
5          num = TextBox1.Text
6          i = 1
7          Do While i <= num
8              age = InputBox("อายุคนที่" & i)
9              total = total + age
10             i = i + 1
11         Loop
12         aver = total / num
13         TextBox2.Text = aver
14     End Sub
15 End Class
```

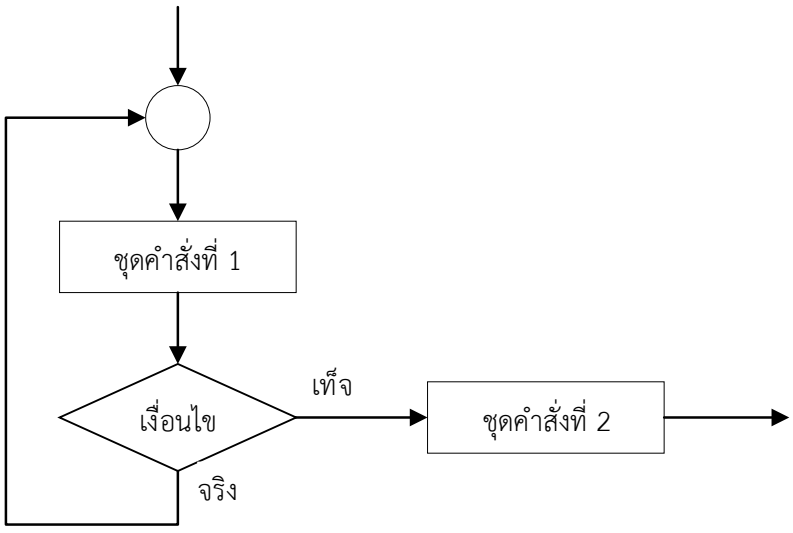
ภาพที่ 5.26 ตัวอย่างโปรแกรมหาอายุเฉลี่ยตามจำนวนนักศึกษาโดยคำสั่ง Do While ... Loop

5.2.4 Do ... Loop While

คำสั่ง Do ... Loop While คือ คำสั่งที่ให้ทำซ้ำ มีลักษณะการทำงานแตกต่างจากคำสั่ง While คือ จะประมวลชุดคำสั่งก่อนถึงไปตรวจสอบเงื่อนไข แต่คำสั่ง Do While จะตรวจสอบเงื่อนไขก่อนประมวลผลชุดคำสั่ง ซึ่งมีรูปแบบไวยากรณ์ดังนี้

```
Do
    ชุดคำสั่ง (เมื่อเงื่อนไขเป็นจริง)
Loop While (เงื่อนไข)
```

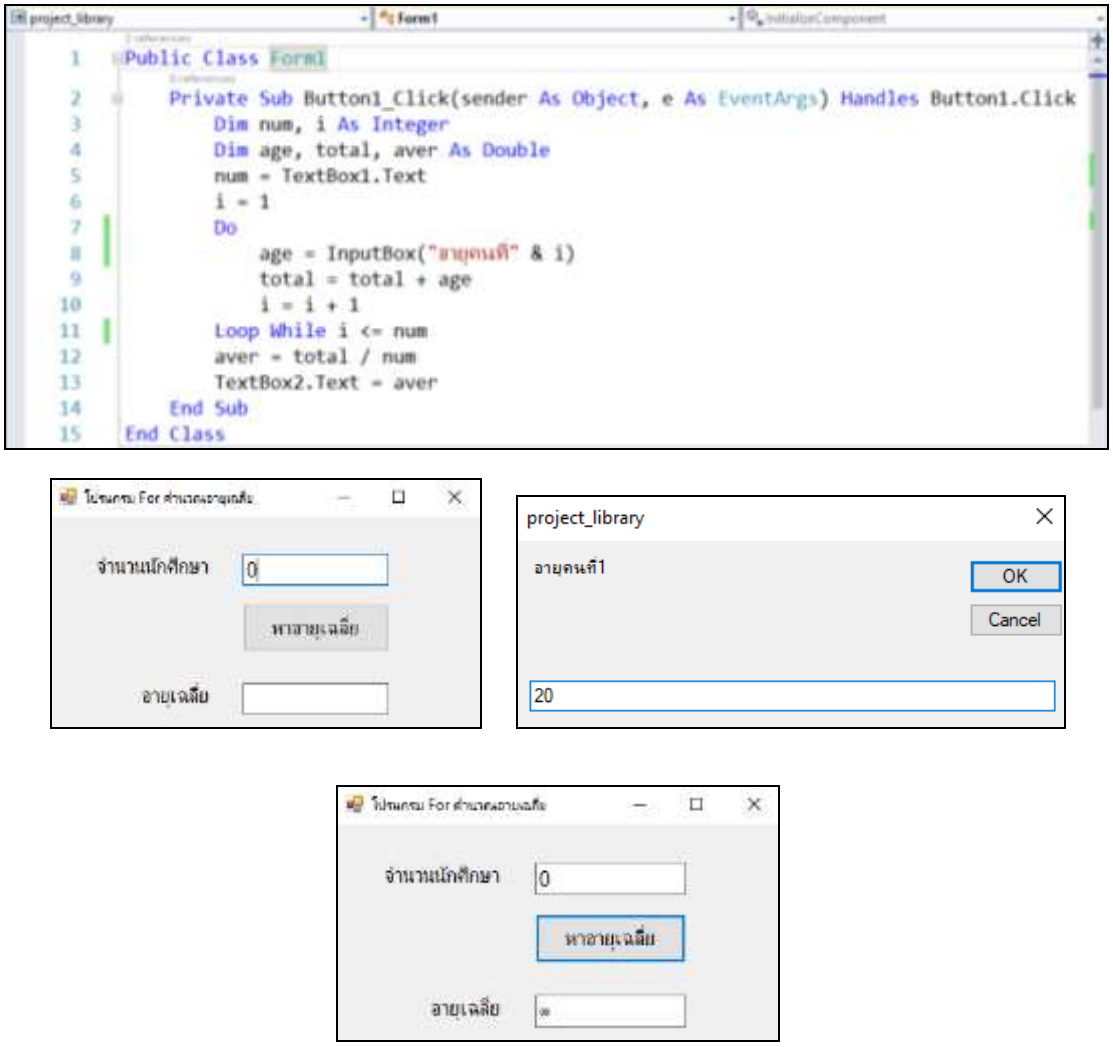
จากรูปแบบไวยากรณ์ของคำสั่ง Do ... Loop While สามารถอธิบายโครงสร้างการทำงานของคำสั่งได้ดัง ภาพที่ 5.27



ภาพที่ 5.27 โครงสร้างการทำงานของคำสั่ง Do ... Loop While

คำสั่ง Do ... Loop While จะประมวลผลชุดคำสั่งที่ 1 ก่อนเสมอ จากนั้นจะทำการตรวจสอบเงื่อนไข ถ้าตรงกับเงื่อนไขหรือเงื่อนไขเป็นจริง จะดำเนินการกลับไปประมวลผลชุดคำสั่งที่ 1 ซ้ำอีกครั้ง จนกระทั่งไม่ตรงเงื่อนไขหรือเป็นเท็จ จึงไปประมวลผลชุดคำสั่งที่ 2 ดังภาพที่ 5.27

จากตัวอย่าง ให้หาอายุเฉลี่ยของนักศึกษาตามจำนวนศึกษาที่กำหนด ที่ได้กล่าวมาในข้างต้น สามารถนำมาเขียนโปรแกรม โดยใช้คำสั่ง Do ... Loop While ได้ดังภาพที่ 5.28



ภาพที่ 5.28 ตัวอย่างโปรแกรมหาอายุเฉลี่ยตามจำนวนนักศึกษาโดยคำสั่ง Do ... Loop While

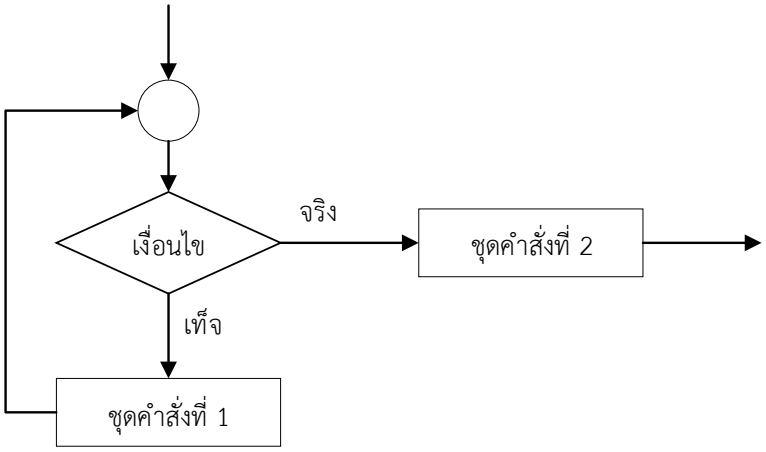
จากภาพที่ 5.28 แสดงให้ว่าข้อมูลที่ป้อนเข้าไปในโปรแกรม จะไม่เข้าเงื่อนไข แต่โปรแกรมก็จะประมวลผลชุดคำสั่ง 1 ครั้งเสมอ ซึ่งเป็นลักษณะการทำงานของคำสั่ง Do ... Loop While

5.2.5 Do Unit ... Loop

คำสั่ง Do Unit ... Loop คือ คำสั่งทำการตรวจสอบเงื่อนไขก่อน โดยประมวลผลคำสั่งและทำซ้ำเมื่อไม่ตรงกับเงื่อนไขหรือเงื่อนไขเป็นเท็จ แต่ถ้าเงื่อนไขเป็นจริงจะไม่ทำซ้ำ ซึ่งมีรูปแบบไวยากรณ์คำสั่งดังต่อไปนี้

```
Do Until (เงื่อนไข)
    ชุดคำสั่ง (เมื่อเงื่อนไขเป็นเท็จ)
Loop
```

จากรูปแบบไวยากรณ์ของคำสั่ง Do Until ... Loop สามารถอธิบายโครงสร้างการทำงานของคำสั่งได้ดัง ภาพที่ 5.29



ภาพที่ 5.29 โครงสร้างการทำงานของคำสั่ง Do Until ... Loop

คำสั่ง Do Until ... Loop มีการทำงานใกล้เคียงกันกับคำสั่ง Do While แตกต่างกันตรงที่ Do Unit ... Loop ประมวลผลคำสั่งและทำซ้ำเมื่อเงื่อนไขนั้นเป็นเท็จ ดังภาพที่ 5.29

จากตัวอย่าง ให้หาอายุเฉลี่ยของนักศึกษาตามจำนวนศึกษาที่กำหนด ที่ได้กล่าวมาในข้างต้น สามารถนำมาเขียนโปรแกรม โดยใช้คำสั่ง Do Until ... Loop ได้ดังภาพที่ 5.30

```
1  Public Class Form1
2      Private Sub Button1_Click(sender As Object, e As EventArgs) Handles Button1.Click
3          Dim num, i As Integer
4          Dim age, total, aver As Double
5          num = TextBox1.Text
6          i = 1
7          Do Until i > num
8              age = InputBox("อายุคนที่" & i)
9              total = total + age
10             i = i + 1
11         Loop
12         aver = total / num
13         TextBox2.Text = aver
14     End Sub
15 End Class
```

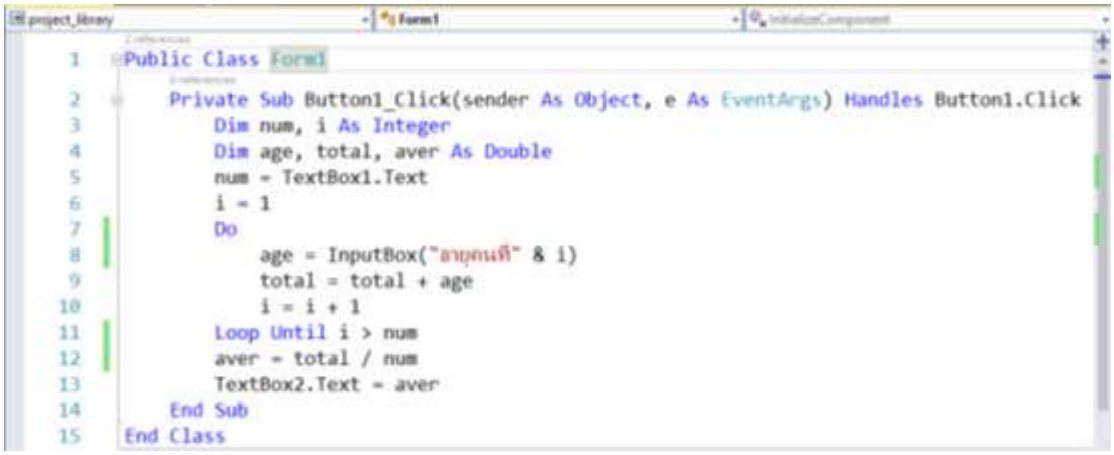
ภาพที่ 5.30 ตัวอย่างโปรแกรมหาอายุเฉลี่ยตามจำนวนนักศึกษาโดยคำสั่ง Do Until ... Loop

5.2.6 Do ... Loop Until

คำสั่ง Do ... Loop Until คือ คำสั่งที่ให้ทำซ้ำ มีลักษณะการทำงานแตกต่างจากคำสั่ง Do Until ... Loop คือ จะประมวลชุดคำสั่งก่อนถึงไปตรวจสอบเงื่อนไข แต่คำสั่ง Do Until ... Loop จะตรวจสอบเงื่อนไขก่อนประมวลผลชุดคำสั่ง ซึ่งมีรูปแบบไวยากรณ์ดังนี้

```
Do
    ชุดคำสั่ง (เมื่อเงื่อนไขเป็นเท็จ)
Loop Until (เงื่อนไข)
```

จากตัวอย่าง ให้หาอายุเฉลี่ยของนักศึกษาตามจำนวนศึกษาที่กำหนด ที่ได้กล่าวมาในข้างต้น สามารถนำมาเขียนโปรแกรม โดยใช้คำสั่ง Do ... Loop Until ได้ดังภาพที่ 5.31



```
1  Public Class Form1
2      Private Sub Button1_Click(sender As Object, e As EventArgs) Handles Button1.Click
3          Dim num, i As Integer
4          Dim age, total, aver As Double
5          num = TextBox1.Text
6          i = 1
7          Do
8              age = InputBox("อายุคนที่" & i)
9              total = total + age
10             i = i + 1
11         Loop Until i > num
12         aver = total / num
13         TextBox2.Text = aver
14     End Sub
15 End Class
```

ภาพที่ 5.31 ตัวอย่างโปรแกรมหาอายุเฉลี่ยตามจำนวนนักศึกษาโดยคำสั่ง Do ... Loop Until

คำสั่งควบคุมทิศทางการทำงานของโปรแกรมแบบทำซ้ำ ถือว่ามีความสำคัญต่อลักษณะของการประมวลผลข้อมูลแบบทำซ้ำ ในลักษณะงานที่มีความซับซ้อน ซึ่งในโปรแกรมวิชวลเบสิก มีคำสั่งควบคุมทิศทางการทำงานแบบทำซ้ำหลายรูปแบบ ซึ่งสามารถเลือกใช้ตามความเหมาะสมได้

สรุป

ในบทนี้กล่าวถึงโครงสร้างการทำงานของโปรแกรม ซึ่งแบ่งออกเป็น 3 ประเภท ได้แก่ แบบเรียงลำดับ แบบมีเงื่อนไข และแบบทำซ้ำ ซึ่งแบบเรียงลำดับ คือ โครงสร้างที่จะประมวลผลคำสั่งทุกชุดคำสั่ง แต่โครงสร้างแบบมีเงื่อนไขและโครงสร้างทำซ้ำ จะกำหนดเส้นทางการทำงานของโปรแกรม ดังนั้นจึงมีคำสั่งควบคุมทิศทางการทำงานของโปรแกรม ซึ่งแบบมีเงื่อนไข ประกอบด้วยคำสั่ง If และ Select ส่วนคำสั่งควบคุมทิศทางการทำงานของโปรแกรมแบบทำซ้ำประกอบไปด้วยคำสั่ง For, Do While, Do ... Loop While, Do Until ... Loop และ Do ... Loop Until ซึ่งสามารถเลือกนำไปใช้ตามรูปแบบของคำสั่ง เพื่อให้การเขียนโปรแกรม ได้ผลลัพธ์ที่ถูกต้องและเกิดประสิทธิภาพสูงสุดในการทำงานของโปรแกรม

แบบฝึกหัด

1. ให้ผู้เรียนบอกโครงสร้างการทำงานของโปรแกรมมีทั้งหมดกี่ประเภท ประกอบด้วยอะไรบ้าง พร้อมทั้งอธิบายความแตกต่างระหว่างโครงสร้างการทำงาน

2. ให้ผู้เรียนบอกคำสั่งควบคุมทิศทางการทำงานของโปรแกรมแบบมีเงื่อนไข มีอะไรบ้าง พร้อมทั้งอธิบายแต่ละลักษณะของคำสั่ง

3. ให้ผู้เรียนบอกคำสั่งควบคุมทิศทางการทำงานของโปรแกรมแบบทำซ้ำ มีอะไรบ้าง พร้อมทั้งอธิบาย แต่ละลักษณะของคำสั่ง

4. ให้ผู้เรียนเขียนโปรแกรมหาพื้นที่ของวงกลม

5. ให้ผู้เรียนเขียนโปรแกรมโดยหาเกรดที่ได้ มีเงื่อนไขช่วงคะแนนดังนี้

80-100	เกรด	A	75-79	เกรด	B+
70-74	เกรด	B	65-69	เกรด	C+
60-64	เกรด	C	55-59	เกรด	D+
50-54	เกรด	D	0-49	เกรด	E

ซึ่งคะแนนได้มาจาก คะแนนเก็บ คะแนนกลางภาค และคะแนนปลายภาค

6. ให้ผู้เรียนเขียนโปรแกรมให้หาเงินงวดต่อเดือนรถยนต์ โดยมีอัตราดอกเบี้ย โดย จะต้องมีเงินดาวน์ 20% ของราคารถยนต์ ซึ่งถ้าผ่อนต่ำกว่า 3 ปี คิดดอกเบี้ย 3.0% ถ้าผ่อน 4-5 ปี คิดดอกเบี้ย 3.3% และ 6 ปีคิดดอกเบี้ย 3.6% โดยมีสูตรการคำนวณดังนี้

เงินดาวน์ = ราคารถยนต์ * 20 / 100

ยอดจัดไฟแนนท์ = ราคารถยนต์ – เงินดาวน์

เงินดอกเบี้ย = (ยอดจัดไฟแนนท์ * ดอกเบี้ย / 100) * จำนวนปี

เงินงวดต่อเดือน = (ยอดจัดไฟแนนท์ + เงินดอกเบี้ย) / จำนวนเดือน

7. ให้ผู้เรียนเขียนโปรแกรมหาผลรวมของเลขคู่ ระหว่าง 10 ถึง 30

8. ให้ผู้เรียนเขียนโปรแกรมให้มีรูปร่างดังนี้

```
***
******
*********
**********
```

9. ให้ผู้เรียนเขียนโปรแกรมให้หาราคารวมจากสินค้าทั้งหมดในตะกร้า พร้อมกับเงินทอน โดยใช้คำสั่ง For และคำสั่ง Do While .... Loop

10. ให้ผู้เรียนยกตัวอย่างการเขียนโปรแกรมที่มีการคำสั่ง Do....Loop While และ Do...Loop Until

เอกสารอ้างอิง

กิตินันท์ พลสวัสดิ์. (2554). **คู่มือเรียนและใช้งาน Visual C# 2010 ฉบับสมบูรณ์**. กรุงเทพฯ: ไอดีซี พรีเมียร์.

ธีรวัฒน์ ประกอบผล. (2558). **การเขียนแอพพลิเคชันด้วย Visual Basic 2010**. กรุงเทพฯ: ซิมพลิฟลาย.

สัจจะ จรัสรุ่งรวีวร. (2554). **คู่มือเรียนและใช้งาน Visual Basic 2010 ฉบับสมบูรณ์**. กรุงเทพฯ: ไอดีซี พรีเมียร์.

สุชาติ คุ้มมณี. (2557). **เชี่ยวชาญการเรียนด้วยภาษาไพธอน**. มหาสารคาม: มหาวิทยาลัยมหาสารคาม.

Dierbach, Charles. (2013). **Introduction to Computer Science Using Python: A Computational Problem-Solving Focus**. New Jersey: Wiley.

Willis, Thearon and Newsome, Bryan. (2011). **Beginning Microsoft Visual Basic 2010**. Indianapolis. Indianapolis: Wiley.