บทที่ 3

ทฤษฎีและแนวความคิดที่นำมาใช้

ในการพัฒนาโปรแกรมระบบจินตทัศน์อัลกอริทึม สิ่งที่มีความสำคัญที่สุดก็คือ ความชัดเจนในการสื่อความหมายของการจินตทัศน์แต่ละมุมมองซึ่งจะทำให้ผู้ใช้งานเกิดความเข้าใจได้ง่ายและเกิดความสนใจและนอกจากนั้นโปรแกรมควรจะมีความง่ายต่อการใช้งานอีกด้วย ดังนั้นผู้วิจัยจึงเลือกที่จะทำการพัฒนาภายใต้สภาพปฏิบัติการไมโครซอฟต์วินโดวส์และใช้ตัวแปลภาษาวิชวลเบสิก เป็นเครื่องมือในการเขียนโปรแกรม

เพื่อเป็นการสร้างพื้นฐานในการทำความเข้าใจอัลกอริทึมการเรียงลำดับข้อมูลผู้วิจัยใคร่ขอกล่าวถึงอัลกอริทึมในการเรียงลำดับข้อมูลแบบต่างๆที่นำมาใช้ในการพัฒนาโครงงานนี้

<u>การเรียงลำดับข้อมูล</u>

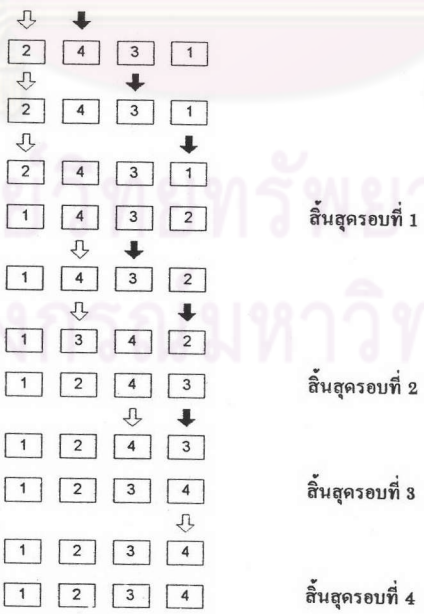
การเรียงลำดับข้อมูล คือขบวนการหรือขั้นตอนที่เราใช้กระทำกับชุดของข้อมูลเพื่อให้ได้ผลลัพธ์ตามเงื่อนไขและทิศทางตามที่ต้องการ ผลของการเรียงลำดับข้อมูลจะทำให้ข้อมูลแต่ละตัวในชุดจะถูกย้ายไปอยู่ในตำแหน่งที่ถูกต้อง เงื่อนไขของการเรียงลำดับข้อมูลมีได้หลายแบบ เช่น การเรียงลำดับโดยใช้ค่าของข้อมูลเป็นเงื่อนไข การเรียงลำดับโดยใช้ขนาดของข้อมูลเป็นเงื่อนไข ฯลฯ ส่วนทิศทางการเรียงลำดับข้อมูลโดยทั่วไปจะมีอยู่ 2 ทิศทางด้วยกันคือ การเรียงจากน้อยไปหามาก และการเรียงจากมากไปหาน้อย สำหรับการเรียงลำดับข้อมูลในสาขาวิชาคอมพิวเตอร์มักจะใช้ค่าของข้อมูลเป็นเงื่อนไข เช่น สมมุติชุดของข้อมูลเป็นดังนี้คือ 5, 3, 2, 7, 4, 1, 0 และเมื่อข้อมูลชุดนี้ได้ผ่านการเรียงลำดับในทิศทางจากน้อยไปหามากจะมีการสลับตำแหน่งใหม่เป็น 0, 1, 2, 3, 4, 5, 7 และในทางกลับกันถ้าทำการเรียงจากมากไปน้อยก็จะได้ผลลัพธ์คือ 7, 5, 4, 3, 2, 1, 0 ในการทำงานจริง ข้อมูลขาเข้าอาจจะเป็นตัวอักษร หรือเป็นได้ทั้งตัวเลขและตัวอักษรปนกันก็ได้

ดังที่ได้กล่าวไว้ในบทที่ 1 คือ ปัญหาหนึ่งๆสามารถมีขบวนการหรือขั้นตอนในการแก้ไขได้หลายวิธี ดังนั้นเมื่อเราต้องการเรียงลำดับข้อมูลก็เช่นกัน ทุกๆอัลกอริทึมของการเรียงลำดับข้อมูลสามารถนำมาใช้แก้ปัญหาได้แต่การที่จะเลือกอัลกอริทึมได้อย่างเหมาะสมนั้นผู้เลือก

ควรจะมีความเข้าใจในขั้นตอนการทำงานของอัลกอริทึมแต่ละชนิดเป็นอย่างดี ในบางสถานะการณ์การเลือกใช้อัลกอริทึมที่มีขั้นตอนการทำงานแบบง่ายๆอาจจะเหมาะสมกว่าการเลือกใช้อัลกอริทึมที่ทำการเรียงลำดับข้อมูลได้เร็วแต่มีความซับซ้อนมากกว่า ดังนั้นเพื่อให้เกิดความเข้าใจในหลักการทำงานจึงขอแนะนำอัลกอริทึมที่ใช้ในการเรียงลำดับข้อมูลแต่ละแบบดังนี้

1. การเรียงลำดับข้อมูลแบบเลือก (Selection Sort)

อัลกอริทึมการเรียงลำดับข้อมูลแบบนี้ มักจะเป็นอัลกอริทึมแรกที่ใช้ในการสอน เนื่องจากสามารถทำความเข้าใจได้ง่าย วิธีที่ง่ายที่สุดในการอธิบายหลักการทำงานของอัลกอริทึมนี้ก็คือ สมมุติว่าเรามีบัตรอยู่จำนวนหนึ่งซึ่งในบัตรแต่ละใบจะมีหมายเลขกำกับอยู่ หากเราต้องการจัดเรียงบัตรเหล่านี้ให้เรียงจากน้อยไปหามากโดยใช้อัลกอริทึมการเรียงลำดับข้อมูลแบบเลือกก็คือ ในขั้นแรกให้ทำการเลือกบัตรขึ้นมาใบหนึ่งถือไว้ในมือจากนั้นให้หยิบบัตรอีกใบหนึ่งขึ้นมาแล้วเปรียบเทียบว่าใบใดมีตัวเลขน้อยกว่าให้ถือบัตรใบนั้นไว้และทิ้งใบที่มากกว่าแยกไว้อีกกองหนึ่ง จากนั้นจึงหยิบบัตรใบต่อไปแล้วเปรียบเทียบอีกเช่นกันบัตรใบใดที่ถูกทิ้งกลับลงไปในกองจะต้องมีค่ามากกว่าบัตรที่อยู่ในมือ ทำเช่นนี้จนบัตรในกองหมดก็จะได้บัตรใบแรกที่มีตัวเลขน้อยที่สุดกำกับอยู่จึงนำบัตรใบนี้มาวางไว้ทางซ้ายมือถือว่าเป็นตำแหน่งที่หนึ่ง นี่คือ 1 รอบการทำงาน จากนั้นให้ทำการเลือกในลักษณะเดียวกันในแต่ละรอบจะได้ตัวเลขที่น้อยที่สุดของกองมาเรียงต่อกันไป สุดท้ายจะได้บัตรนี้มีการเรียงตัวจากน้อยไปหามากในทิศทางจากซ้ายไปขวา



รูปที่ 3.1 แสดงขั้นตอนการเลือกบัตร

จากรูปที่ 3.1 จะสังเกตได้ว่าเราจะได้ตัวเลขที่น้อยที่สุดของกองออกมาจัดเรียง จนสุดท้ายของรอบที่ 4 ก็จะได้ชุดตัวเลขที่เรียงตามลำดับ จากขั้นตอนนี้เราสามารถนำมาเขียนเป็นอัลกอริทึมโดยใช้ภาษาเบสิกได้ตามรูปที่ 3.2

```
Sub SelectionSort (NumData As Integer)
  item = GetDataItem(NumData - 1)
  For i = (NumData - 1) To 0 step -1
       max = i
       For j = (i - 1) To 0 step -1
            if item(j) > item(max) Then
                 max = j
            EndIf
       Next j
       temp = item(i)
       item(i) = item(max)
       item(max) = temp
  Next i
End Sub
```

รูปที่ 3.2 แสดงอัลกอริทึมการเรียงลำดับข้อมูลแบบเลือก

2. การเรียงลำดับข้อมูลแบบแทรก (Insertion Sort)

ในความเป็นจริงแล้วอัลกอริทึมการเรียงลำดับข้อมูลแบบนี้เป็นอัลกอริทึมที่คนเราทั่วไปใช้กันมากและคุ้นเคยที่สุดตัวอย่างที่ชัดเจน คือ การเรียงไพ่ที่อยู่ในมือหรือการเรียงเลขที่เอกสารของพนักงานเสมียน เป็นต้น

จะเห็นว่าในการจัดเรียงดังกล่าวเรามีวิธีการคือ

1. หาตัวเลขตัวใดตัวหนึ่งเป็นตัวตั้งต้น
2. เลือกตัวเลขตัวอื่นขึ้นมา แล้วทำการเปรียบเทียบกับตัวเลขตั้งต้น หากมีค่าน้อยกว่าให้วางไว้ทางซ้ายมือ แต่ถ้ามีค่ามากกว่าให้วางลงทางขวามือ
3. ในกรณีที่ตัวเลขที่เลือกขึ้นมามีค่าอยู่ระหว่างตัวเลขสองตัว ให้ทำการแทรกลงไปในระหว่างตัวเลขคู่นั้น
4. ทำเช่นนี้เรื่อยไปจนตัวเลขทั้งหมดเรียงตามลำดับที่ต้องการ

ดังนั้นเราสามารถนำขั้นตอนข้างต้นมาเขียนเป็นอัลกอริทึมด้วยภาษาเบสิกได้ดังรูปที่ 3.3

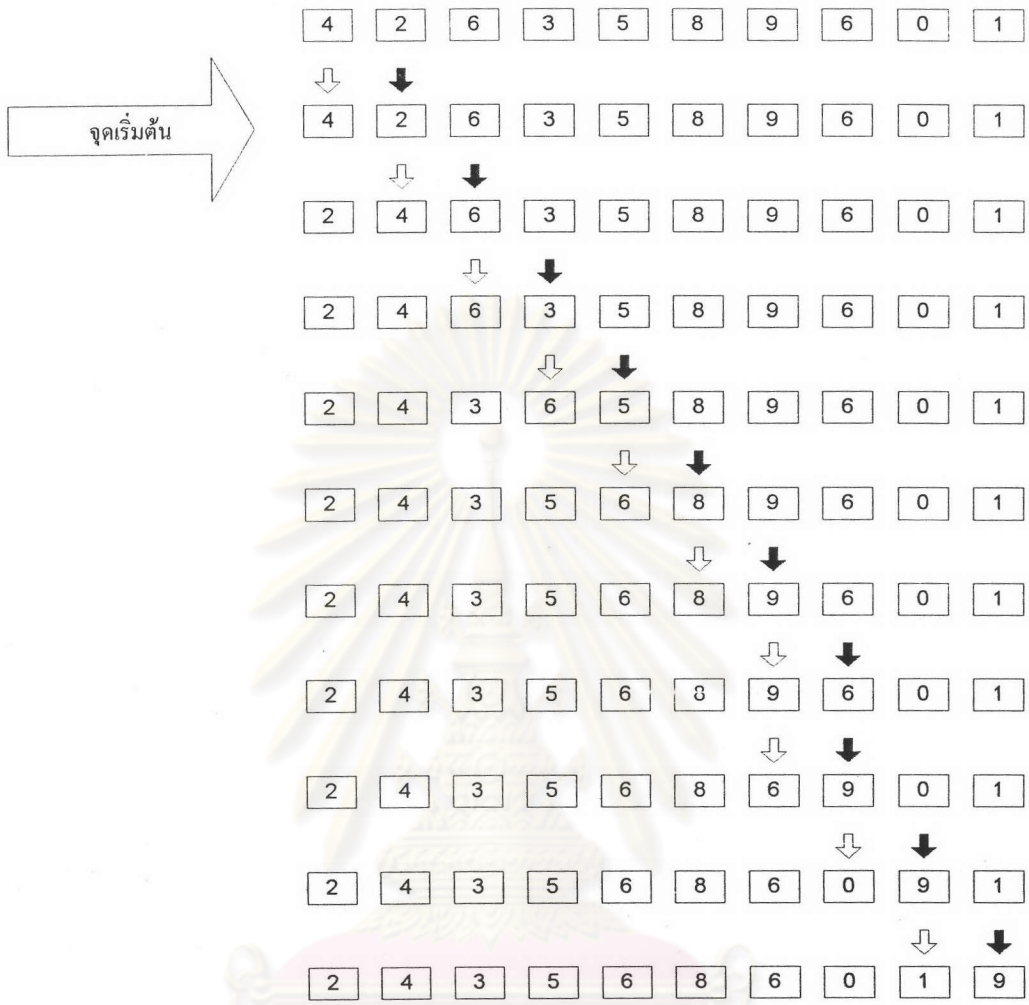
```
Sub InsertionSort (NumData As Integer)
     item = GetDataItem(NumData - 1)
     For i = 1 To (NumData - 1)
        temp = item(i)
        j = i
        Do While (j - 1) >= 0
          If item(j - 1) > temp Then
             item(j) = item(j - 1)
             j = j - 1
          Else
             Exit Do
          EndIf
        Loop
        item(j) = temp
     Next i
End Sub
```

รูปที่ 3.3 แสดงอัลกอริทึมการเรียงลำดับข้อมูลแบบแทรก

3. การเรียงลำดับข้อมูลแบบฟอง (Bubble Sort)

อัลกอริทึมของการเรียงลำดับข้อมูลแบบนี้มีหลักการสำคัญคือทำการเปรียบเทียบข้อมูลที่อยู่ติดกันทีละคู่ ถ้าข้อมูลที่อยู่ติดกันนั้นเรียงผิดลำดับอยู่ เช่นตัวที่น้อยกว่ากลับไปอยู่ทางขวามือของอีกตัวหนึ่งก็จะทำการสลับตำแหน่งข้อมูลคู่นั้นให้ถูกต้อง ลักษณะเช่นนี้จะทำให้ข้อมูลที่อยู่ผิดตำแหน่งค่อยๆทำการสลับเลื่อนตำแหน่งไปเรื่อยๆจนกว่าจะได้ตำแหน่งที่ถูกต้อง และเพื่อให้เข้าใจง่ายขึ้นจะขอยกตัวอย่างของชุดข้อมูล 10 ตัว และทำการเปรียบเทียบตามการทำงานของอัลกอริทึมจะได้ดังรูปที่ 3.4

เมื่อเสร็จสิ้นการสลับตำแหน่งในรอบแรกจะเห็นว่าเลข 9 ถูกสลับไปอยู่ในตำแหน่งที่ถูกต้องแล้วหนึ่งตัว จากนั้นก็จะทำการเปรียบเทียบชุดตัวเลขที่ได้มาใหม่นี้ตั้งแต่ต้นโดยใช้วิธีเดียวกันกับในตอนแรก คือเริ่มต้นเปรียบเทียบ ระหว่างเลข 2 และเลข 4 ต่อไปจนในที่สุดข้อมูลทั้งหลายจะถูกสลับไปอยู่ในตำแน่งที่ถูกต้อง ซึ่งหากเราสังเกตดูแล้วจะเหมือนกับว่าข้อมูลที่มีค่ามากกำลังขยับไปข้างหน้าส่วนข้อมูลที่มีค่าน้อยกว่ากำลังขยับถอยหลังไปยังตำแหน่งที่ถูกต้องจนดูราวกับว่ากำลังค่อยๆล่องลอยไปยังตำแหน่งของตัวเอง จึงได้ชื่อว่า Bubble Sort หรือการเรียงลำดับข้อมูลแบบฟองนั่นเอง (คล้ายกับฟองอากาศที่กำลังลอยขึ้นไปเรื่อยๆ) จากขั้นตอนการทำงานข้างต้นเราสามารถนำมาเขียนเป็นอัลกอริทึมได้ดังรูปที่ 3.5



รูปที่ 3.4 ผลการทำงานของอัลกอริทึมการเรียงลำดับข้อมูลแบบฟอง

หลังจากจบการทำงานในรอบแรก

```
Sub BubbleSort (NumData As Integer)
    item = GetDataItem(NumData - 1)
    For i = (NumData - 1) To 0 Step - 1
       For j = 1 To i
          If item(j - 1) > item(j) Then
            temp = item(j - 1)
            item(j - 1) = item(j)
            item(j) = temp
          EndJf
       Next j
    Next i
End Sub
```

รูปที่ 3.5 แสดงอัลกอริทึมการเรียงลำดับข้อมูลแบบฟอง

4. การเรียงลำดับข้อมูลแบบเชลล์ (Shell Sort)

จากการทำงานของอัลกอริทึมทั้ง 3 ชนิดที่ผ่านมาจะสังเกตได้ว่ามีขั้นตอนการเปรียบเทียบและการเคลื่อนย้ายตำแหน่งของข้อมูลบ่อยครั้งมาก ซึ่งจะมีผลทำให้การทำงานของอัลกอริทึมช้าลงไปด้วยและยิ่งไปกว่านั้น ใน 3 วิธีดังกล่าว เมื่อมีการเปรียบเทียบแต่ละครั้งข้อมูลจะถูกเคลื่อนย้ายไปเพียงหนึ่งตำแหน่งเนื่องจากเป็นการเปรียบเปทียบข้อมูลที่อยู่ติดกัน ดังนั้น หากเราสามารถทำการเปรียบเทียบข้อมูลที่อยู่ห่างๆกันก่อนเพื่อให้ข้อมูลที่อยู่ผิดตำแหน่งสามารถทำการสลับที่กันไปอยู่ในตำแหน่งที่ถูกต้องได้เร็วขึ้น ผลที่ได้คือประสิทธิภาพในการเรียงลำดับข้อมูลจะดีขึ้นด้วยนั่นเอง

อัลกอริทึมการเรียงลำดับข้อมูลแบบเชลล์เป็นอัลกอริทึมหนึ่งที่ใช้หลักการดังกล่าวโดยการนำข้อมูลมาแบ่งเป็น 2 กลุ่มหรือมากกว่า รูปที่ 3.6 แสดงอัลกอริทึมการเรียงลำดับข้อมูลแบบเชลล์

```
Sub ShellSort(NumData - 1)
    item = GetDataItem(NumData - 1)
    incStep = (NumData - 1)
    While incStep > 1
       incStep = Int(incStep / 3) + 1
       For idx = incStep To (NumData - 1 )
         For jdx = (idx - incStep) To 0 Step -incStep
            If item(jdx) <= item(jdx + incStep) Then
               Exit For
            Else
               tmp = item(jdx)
               item(jdx) = item(jdx + incStep)
               item(jdx + incStep) = tmp
            End If
         Next jdx
       Next idx
    Wend
End Sub
```

รูปที่ 3.6 แสดงอัลกอริทึมการเรียงลำดับข้อมูลแบบเชลล์

5. การเรียงลำดับข้อมูลแบบเร็ว (Quick Sort)

อัลกอริทึมชนิดนี้ถูกคิดค้นขึ้นโดย C.A.R. Hoare ซึ่งหลักการทำงานของอัลกอริทึมชนิดนี้จะเริ่มจากการแบ่งข้อมูลออกเป็นสองส่วน โดยอาจใช้วิธีสุ่มเลือกใช้ค่าของข้อมูลตัวหนึ่งเป็นตัวเปรียบเทียบ หรืออาจจะใช้ข้อมูลตัวที่อยู่ตำแหน่งกลางของข้อมูลทั้งหมดก็ได้ จากนั้นข้อมูลจะถูกแบ่งออกเป็นสองส่วนแล้วจึงพยายามทำให้ข้อมูลที่อยู่กลุ่มทางซ้ายมือมีค่าน้อยกว่าค่าของข้อมูลที่ใช้เป็นตัวเปรียบเทียบ ส่วนทางขวามือก็พยายามทำให้เป็นกลุ่มที่มีค่ามากกว่าค่าของตัว

เปรียบเทียบจากนั้นก็นำเอาแต่ละกลุ่มมาเรียงลำดับใหม่โดยใช้หลักการเช่นเดิมอีกคือ เลือกตัวเปรียบเทียบขึ้นมาใหม่ภายในกลุ่มย่อยทั้งสองกลุ่มทำเช่นนี้เรื่อยไปจนกระทั่งข้อมูลในกลุ่มเหลือเพียงตัวเดียวก็เป็นอันเสร็จสิ้นการทำงาน จากวิธีการทำงานดังกล่าวสามารถนำมาเขียนเป็นอัลกอริทึมได้ดัง รูปที่ 3.7

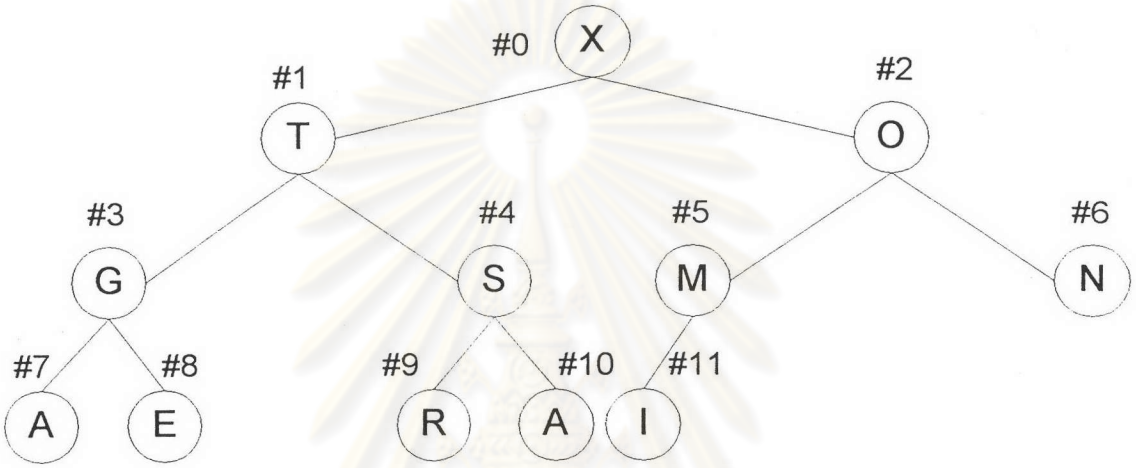
```
Sub QuickSort(NumData As Integer)
    item = GetDataItem(NumData - 1)
    QS 0,(NumData-1)
End Sub

Sub QS(left As Integer, right As Integer)
   idx = left
   jdx = right
   v = item((left + right) / 2)
   Do
        While item(idx) < v And idx < right
             idx = idx + 1
        Wend
        While item(jdx) > v And jdx > left
             jdx = jdx - 1
        Wend
        If idx <= jdx Then
             temp = item (idx)
             item(idx) = item(jdx)
             item(jdx) = temp
             idx = idx + 1
             jdx = jdx - 1
        EndIf
   Loop While idx <= jdx
   If left < jdx Then
        QS left,jdx
   EndIf
   If idx < right Then
        QS idx, right
   EndIf
End Sub
```

รูปที่ 3.7 แสดงอัลกอริทึมการเรียงลำดับข้อมูลแบบเร็ว

6. การเรียงลำดับข้อมูลแบบฮีบ (Heap Sort)

ในการแข่งขันกีฬาเพื่อหาผู้ชนะเราจะพบว่า นักกีฬาที่ชนะเลิศจะขึ้นรับรางวัลบนแท่นรับรางวัลชั้นสูงสุด และนักกีฬาที่ชนะเป็นอันดับสองและสามก็จะขึ้นรับในตำแหน่งถัดลงมา หากพิจารณาถึงแท่นรับรางวัลนี้จะเห็นว่าคล้ายคลึงกับโครงสร้างข้อมูลแบบต้นไม้ โดยมีโหนด (node)แต่ละโหนดแทนตำแหน่งของรางวัลนั่นเอง ในการทำงานของอัลกอริทึมการเรียงลำดับข้อมูลแบบฮีบสามารถแบ่งได้เป็นสองส่วนคือ การสร้างฮีบในส่วนแรกและส่วนหลังคือการนำฮีบ

มาใช้ ในการสร้างฮีบนั้นผู้คิดค้นอัลกอริทึมนี้ได้นำโครงสร้างข้อมูลแบบต้นไม้มาใช้โดยมีเงื่อนไขคือ โหนดลูกจะต้องมีค่าของข้อมูลน้อยกว่าโหนดพ่อที่อยู่เหนือขึ้นไปเสมอ จากเงื่อนไขนี้จึงทำให้โหนดรากที่อยู่บนสุดจะมีค่าของข้อมูลมากที่สุดนั่นเอง ดังแสดงตัวอย่างในรูปที่ 3.8.1 ดังนั้นจึงเปรียบการสร้างฮีบนี้เหมือนกับการแข่งขันกีฬา นักกีฬาที่ชนะเป็นอันดับหนึ่งก็คือข้อมูลที่มีค่ามากที่สุดและอยู่ในตำแหน่งบนสุดของฮีบนั่นเอง



รูปที่ 3.8.1 แสดงเงื่อนไขของการสร้างฮีบ

หลังจากที่ได้ทำการสร้างฮีบแล้วในขั้นตอนต่อไปคือการนำฮีบนี้มาใช้ ก็คือทำการสลับตำแหน่งแรกสุดของข้อมูล (ในตอนนี้ก็คือข้อมูลที่มีค่ามากสุด) กับตำแหน่งสุดท้ายแล้วจึงทำการสร้างฮีบขึ้นใหม่ทั้งนี้จะต้องไม่รวมข้อมูลตัวสุดท้ายโดยเริ่มจากข้อมูลตำแหน่งแรกสุดไล่ลงไปเมื่อทำลักษณะนี้ไปจนกระทั่งเหลือข้อมูลในฮีบเพียงตัวเดียวจึงจบการทำงาน รูปที่ 3.8.2 แสดงอัลกอริทึมการเรียงลำดับข้อมูลแบบฮีบในภาษาวิชวลเบสิก

```
Sub HeapSort_ASC(NumData As Integer)
    item = GetDataItem(NumData - 1)
    For idx = 1 To (NumData -1)
        elt = item(idx)
        s = idx
        f = Int((s - 1) / 2)
        Do
          If (item(f) < elt) Then
               item(s) = item (f)
                s = f
               f = Int(Abs((s - 1) / 2))
          Else
               Exit Do
          End If
        Loop While s > 0
        item (s) = elt
    Next idx
    For idx = n To 1 Step -1
        ivalue = item(idx)
        item(idx) = item(0)
        f = 0
        If idx = 1 Then
          s = -1
        Else
          s = 1
        End If
        If idx > 2 And item(2) > item(1) Then
          s = 2
        End If
        Do While s >= 0
          If ivalue < item(s) Then
               item(f) = item(s)
               f = s
               s = 2 * f + 1
               If (s + 1) <= (idx - 1) Then
                    If item(s) < item(s + 1) Then
                         s = s + 1
                    End If
               End If
               If s > (idx - 1) Then
                    s = -1
               End If
          Else
               Exit Do
          End If
        Loop
        item(f) = ivalue
    Next idx
End Sub
```

รูปที่ 3.8.2 แสดงอัลกอริทึมการเรียงลำดับข้อมูลแบบฮีบ

7. การเรียงลำดับข้อมูลแบบผสาน (Merge Sort)

การเรียงลำดับข้อมูลแบบผสานมีหลักการพื้นฐานมาจากการรวมชุดข้อมูลสองชุดที่ผ่านการเรียงลำดับแล้วเป็นข้อมูลชุดใหม่ที่ถูกเรียงลำดับ หลักการดังกล่าวสามารถอธิบายเป็นขั้นตอนได้ดังนี้คือ

1. จากข้อมูลทั้งหมดทำการแบ่งข้อมูลออกเป็นสองชุด

2. จากข้อ 1 ให้ทำซ้ำเช่นเดียวกันจนไม่สามารถแบ่งย่อยลงไปได้อีก นั่นคือเหลือสมาชิกเพียงตัวเดียวในข้อมูลแต่ละชุด

3. นำค่าของข้อมูลในชุดที่ย่อยที่สุดสองชุดมาใส่ลงในพื้นที่สำรอง

4. จากพื้นที่สำรองทั้งสองชุดให้ทำการเลือกค่าของข้อมูลที่น้อยที่สุดออกมาก่อน (ในขั้นตอนนี้จะคล้ายกับการเรียงลำดับข้อมูลแบบเลือก) จนกระทั่งหมด สุดท้ายจะได้ข้อมูลชุดใหม่ที่เกิดจากการรวมกันของข้อมูลสองชุดและมีการเรียงลำดับอย่างถูกต้อง ณ จุดนี้เองจึงเรียกการเรียงลำดับข้อมูลชนิดนี้ว่า การเรียงลำดับข้อมูลแบบผสาน

5. ย้อนกลับไปทำวิธีเดียวกันในข้อ 3 และ 4 แต่กระทำกับชุดข้อมูลย่อยที่มีขนาดใหญ่เหนือขึ้นไปหนึ่งชั้น จนถึงรอบสุดท้ายจะเหลือข้อมูลเพียงสองชุดซึ่งในแต่ละชุดจะมีข้อมูลที่เรียงลำดับอยู่แล้วจึงนำไปใส่ลงในพื้นที่สำรองเช่นเดิมแล้วจึงทำการเลือกอีกครั้งหนึ่งก็จะได้ข้อมูลชุดใหม่เพียงชุดเดียวที่มีการเรียงลำดับอย่างถูกต้อง

จากขั้นตอนดังกล่าวจะเห็นว่าการเรียงลำดับข้อมูลแบบผสานนี้จะมีลักษณะที่แตกต่างจากการเรียงลำดับข้อมูลชนิดอื่นคือ ในขณะทำงานจำเป็นจะต้องใช้พื้นที่สำรองที่มีขนาดเท่ากับชุดข้อมูลขาเข้าเพื่อใช้เป็นพื้นที่สำรองสำหรับเก็บข้อมูลชุดย่อยที่ถูกแบ่งออกมา จึงถือได้ว่าเป็นปัจจัยอันหนึ่งที่จะต้องนำมาพิจารณาหากมีการนำอัลกอริทึมการเรียงลำดับข้อมูลชนิดนี้มาใช้งาน รูปที่ 3.9 แสดงอัลกอริทึมการเรียงลำดับข้อมูลแบบผสานในภาษาวิชวลเบสิก

```
Sub MergeSort_ASC(NumData As Integer)
    ReDim ax (0 To (NumData - 1))
    item = GetDataItem(NumData - 1)
    MS_ASC 0,(NumData - 1)
End Sub

Sub MS_ASC(left As Integer, right As Integer)
    If (right > left) Then
        m = Int ( (right + left) / 2 )
        Call MS_ASC (left, m)
        Call MS_ASC (m + 1, right)
        For idx = (m + 1) To (left + 1) Step -1
          ax(idx - 1) = item(idx - 1)
        Next idx
        For jdx = m To (right - 1)
          ax(right + m - jdx) = item(jdx + 1)
        Next jdx
        For k = left To right
          If ax(idx) < ax(jdx) Then
               item(k) = ax(idx)
               idx = idx + 1
          Else
               item (k) = ax(jdx)
               jdx = jdx - 1
          End If
        Next k
     End If
End Sub
```

รูปที่ 3.9 แสดงอัลกอริทึมการเรียงลำดับข้อมูลแบบผสาน

<u>เทคนิคการเขียนโปรแกรมภายใต้สภาพปฏิบัติการไมโครซอฟต์วินโดวส์</u>

1. ความรู้เบื้องต้นเกี่ยวกับวินโดวส์

ไมโครซอฟต์วินโดวส์ เป็นโปรแกรมระบบที่ทำงานร่วมกับ เอ็มเอสดอส (MS-DOS) อย่างใกล้ชิดเพื่อสร้างสิ่งที่เรียกว่าสภาพแวดล้อมในการปฏิบัติการแบบกราฟิกบนเครื่องไมโครคอมพิวเตอร์

โปรแกรมวินโดวส์ได้เข้าไปทดแทนการใช้บรรทัดคำสั่งของเอ็มเอสดอส จึงทำให้ผู้ใช้ไม่จำเป็นต้องจำคำสั่งของเอ็มเอสดอสอีกต่อไปเช่น แทนที่จะพิมพ์ชื่อของโปรแกรมลงไป ที่เครื่องหมายพร้อมรับคำสั่ง เพื่อเรียกโปรแกรมมาทำงานก็สามารถทำได้ด้วยการคัดเลือกสัญญลักษณ์ที่เป็นรูปภาพที่จำได้ง่าย ซึ่งเรียกว่า สัญรูป (icon) แทนชื่อของโปรแกรม ซึ่งทำให้ง่ายต่อการใช้งาน

ในปัจจุบันบริษัทไมโครซอฟต์ได้ทำการออกวินโดวส์รุ่น 3.1 ซึ่งได้รับการพัฒนาให้มีความสามารถมากขึ้น เช่น การสนับสนุนการทำงานกับระบบเครือข่าย การใช้หน่วยความจำแบบขยาย (Extended Memory) ได้ถึง 16 เมกะไบต์ และหากทำงานบนซีพียู 80386 ขึ้นไปในสภาวะเอนฮานด์แล้ว วินโดวส์จะสามารถใช้การจัดการหน่วยความจำเสมือน (Virtual Memory) ซึ่งทำให้สามารถอ้างอิงหน่วยความจำได้มากขึ้น และนอกจากนั้น ในวินโดวส์รุ่น 3.1 ยังได้มีการพัฒนารูปแบบตัวอักษรใหม่เรียกว่า ทรูไทป์ (True Type) ซึ่งเป็นตัวอักษรแบบโครงร่าง ทำให้สามารถย่อหรือขยายขนาดตัวอักษรได้ตามต้องการ สำหรับในโครงงานวิจัยนี้ได้นำคุณสมบัติที่มีอยู่ในวินโดวส์มาใช้หลายประการด้วยกัน เช่น การแลกเปลี่ยนข้อมูลแบบพลวัต (Dynamic Data Exchange) ซึ่งจะทำให้โปรแกรมภายใต้วินโดวส์ 2 โปรแกรมขึ้นไปมองเห็นข้อมูลชิ้นเดียวกัน หากโปรแกรมใดมีการเปลี่ยนแปลงข้อมูลชิ้นนี้ก็จะทำให้โปรแกรมอื่นๆมองเห็นสิ่งที่เปลี่ยนแปลงไปในทันที สำหรับในรายละเอียดจะได้กล่าวถึงต่อไปในเรื่อง การออกแบบระบบจินตทัศน์ อัลกอริทึมการเรียงลำดับข้อมูล

การทำงานของวินโดวส์รุ่น 3.1 จะแบ่งเป็น 2 ภาวะซึ่งแต่ละภาวะจะขึ้นอยู่กับชนิดและความสามารถของเครื่องคอมพิวเตอร์ที่ใช้งาน ภาวะการทำงานของวินโดวส์ดังกล่าวได้แก่

- ภาวะมาตรฐาน (Standard Mode) เป็นภาวะปกติในการใช้งานวินโดวส์ เพราะในภาวะเช่นนี้จะทำให้วินโดวส์ใช้หน่วยความจำหลักร่วมกับหน่วยความจำแบบขยายได้เท่าที่เครื่องคอมพิวเตอร์เครื่องนั้นจะมีได้

- ภาวะเอนฮานด์ (386 Enhanced mode) เป็นภาวะที่ใช้ความสามารถของซีพียู 80386 ได้อย่างเต็มที่ และใช้หน่วยความจำเสมือนได้ ทำให้โปรแกรมสามารถอ้างอิงหน่วยความจำได้มากกว่าที่มีอยู่จริง นั่นคือเมื่อหน่วยความจำภายหลักจะเต็มแล้ว วินโดวส์จะทำการเขียนข้อมูลที่ยังไม่ได้ใช้งานในขณะนั้นไปไว้ในจานแม่เหล็กแบบแข็ง (hard disk) ก่อน ต่อเมื่อต้องการใช้ข้อมูลนั้นอีก วินโดวส์ก็จะทำการเรียกข้อมูลนั้นกลับคืนมาสู่หน่วยความจำหลักและนำข้อมูลอื่นออกไปไว้ในจานแม่เหล็กแทน ซึ่งวิธีดังกล่าวนี้ทำให้เหมือนกับว่าเครื่องคอมพิวเตอร์นั้นมีหน่วยความจำมากกว่าที่มีอยู่จริง จึงเรียกวิธีการนี้ว่าใช้หน่วยความจำเสมือน

2. คุณลักษณะเด่นของวินโดวส์

2.1 การประสานกับผู้ใช้แบบกราฟิก (Graphic User Interface : GUI)

เพื่อให้เกิดการใช้งานที่เป็นเอกภาพและง่ายต่อความเข้าใจของผู้ใช้ วินโดวส์จึงใช้รูปภาพในการสื่อความหมาย เช่น สัญรูป แถบเลื่อน (Scroll Bar) และเมนู (Menu) จึงทำให้เกิดความสวยงาม ความน่าใช้ สะดวกและง่ายต่อการเรียนรู้

2.2 การทำงานแบบหลายภารกิจ (multitasking)

วินโดวส์เป็นสภาพปฏิบัติการที่เอื้ออำนวยต่อการทำงานแบบหลายภารกิจ ทำให้สามารถพัฒนาการจินตทัศน์ได้มากกว่าหนึ่งอัลกอริทึมและนำเสนอมุมมองได้มากกว่าหนึ่งมุมมองพร้อมๆกัน

2.3 การประสานแบบหลายเอกสาร

(Multiple Document Interface : MDI)

การแสดงมุมมองการนำเสนอแบบหลายเอกสารเป็นคุณสมบัติประการหนึ่งของวินโดวส์ที่เอื้ออำนวยต่อการแสดงและจัดการวินโดวส์ย่อยหลายๆวินโดวส์ที่อยู่ภายใต้วินโดวส์ใหญ่ วินโดวส์ย่อยเหล่านี้จะแทนมุมมองของการจินตทัศน์แต่ละแบบที่กำลังนำเสนอภายใต้อัลกอริทึมเดียวกัน

2.4 การแลกเปลี่ยนข้อมูลแบบพลวัต (Dynamic Data Exchange : DDE)

คือคุณสมบัติในวินโดวส์ที่ทำให้โปรแกรมที่ทำงานภายใต้วินโดวส์ตั้งแต่ สองโปรแกรมขึ้นไปสามารถมองเห็นข้อมูลชิ้นเดียวกันซึ่งจากคุณสมบัตินี้จะมีประโยชน์อย่างยิ่งในการพัฒนาระบบจินตทัศน์อัลกอริทึมซึ่งจะประกอบไปด้วยโปรแกรมที่เป็นผู้ควบคุมและโปรแกรมที่ถูกควบคุม

3. การพัฒนาโปรแกรมประยุกต์บนวินโดวส์โดยใช้ตัวแปลภาษาวิชวลเบสิก (Visual Basic)

3.1 ภาษาเบสิก

ภาษาเบสิกถูกคิดขึ้นมาในปี ค.ศ.1963 โดย John Kemeny และ Thomas Kurtz แห่งสถาบัน Dartmouth College โดยมีจุดมุ่งหมายที่จะออกแบบภาษาเบสิกขึ้นมาเพื่อใช้สอนหลักการเขียนโปรแกรมโดยเน้นที่ความชัดเจนรวดเร็วและประสิทธิภาพในการเรียนรู้ ประกอบกับเป็นภาษาที่เน้นให้ผู้ใช้สนใจในวิธีการและอัลกอริทึมในการแก้ปัญหามากกว่าการที่จะสนใจขั้นตอนการทำงานในส่วนของฮาร์ดแวร์

ภาษาเบสิกได้ถูกพัฒนามาอย่างต่อเนื่องจนเป็นภาษาที่มีโครงสร้างและสามารถแปลโปรแกรมได้เช่นดียวกับภาษาชั้นสูง (compiler) แทนการแปลทีละคำสั่ง (interpreter) ในยุคแรกๆ จนในที่สุดบริษัทไมโครซอฟต์ได้ทำการปฏิวัติภาษาเบสิกขึ้นใหม่โดยเพิ่มเติมความสามารถเข้าไปอีกเช่น การมีโปรแกรมย่อย การมีโครงสร้างข้อมูลชนิดต่าง ๆ การยกเลิกหมายเลขบรรทัดที่ใช้กำกับขั้นตอนการทำงานแล้วให้ชื่อว่า ควิกเบสิก (Quick Basic) ดังแสดงตัวอย่างในรูปที่ 3.10

```
'This is an example program in Quick Basic'
CLS
PRINT "TESTING...";
FOR I = 1 TO 3
  PRINT I;
NEXT I
PRINT;
END
```

รูปที่ 3.10 ตัวอย่างภาษาเบสิกที่พัฒนาจนมีโครงสร้าง

จนกระทั่งสภาพปฏิบัติการไมโครซอฟต์วินโดวส์เป็นที่แพร่หลายในเครื่องระดับไมโครคอมพิวเตอร์จึงทำให้เกิดความต้องการเครื่องมือที่ใช้ในการพัฒนาโปรแกรม ทางบริษัทไมโครซอฟต์จึงได้ทำการเพิ่มเติมความสามารถให้กับควิกเบสิกสำหรับใช้พัฒนาโปรแกรมภายใต้สภาพปฏิบัติการไมโครซอฟต์วินโดวส์จึงเกิดเป็น วิชวลเบสิกรุ่น 1.0 และได้ทำการพัฒนาเรื่อยมาจนกระทั่งปัจจุบันคือวิชวลเบสิกรุ่น 3.0 ในตัวแปลภาษาวิชวลเบสิกนั้นทางบริษัทไมโครซอฟต์ได้นำเทคนิคการโปรแกรมตามเหตุการณ์ที่เกิดขึ้น (Event - Driven Programming) ซึ่งจะต่างกับการโปรแกรมแบบทั่วไปที่เป็นการโปรแกรมแบบเรียงลำดับ (Sequential Programming) มาใช้ในขั้นตอนการพัฒนาโปรแกรมอีกด้วย

3.2 การพัฒนาโปรแกรมตามเหตุการณ์ที่เกิดขึ้น

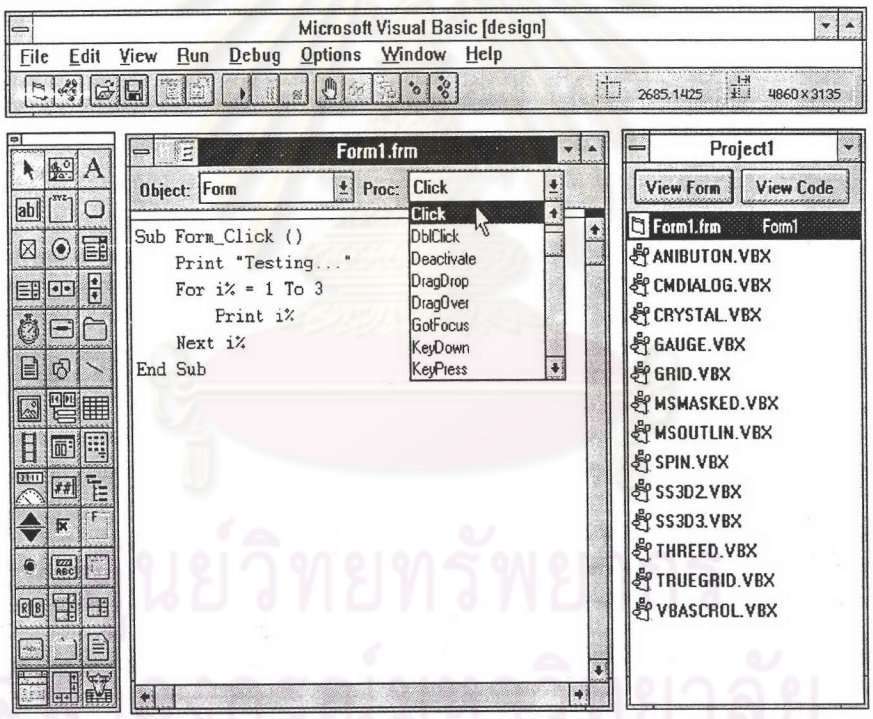
การพัฒนาโปรแกรมโดยใช้ตัวแปลภาษาวิชวลเบสิกใต้วินโดวส์จะเป็นแบบตามเหตุการณ์ที่เกิดขึ้น ซึ่งหลักการพัฒนาโปรแกรมแบบนี้เป็นหัวใจของการพัฒนาโปรแกรมต่างๆที่ทำงานภายใต้วินโดวส์ สามารถอธิบายได้คือ โปรแกรมจะประกอบไปด้วยโปรแกรมย่อยหรือฟังก์ชันเท่านั้นและโปรแกรมย่อยหรือฟังก์ชันต่างๆเหล่านี้จะทำงานเมื่อมีเหตุการณ์มากระตุ้น ตัวอย่างเช่น เมื่อมีการกดปุ่มเมาส์ การเลือกปุ่มคำสั่งหรือแม้แต่การเลื่อนเมาส์ไปมาบนพื้นที่ๆเป็นหน้าจอของโปรแกรมก็ถือเป็นเหตุการณ์เช่นกัน ตัวอย่างของเหตุการณ์ที่สังเกตได้ง่ายคือการแสดงคู่ลำดับของพิกัดตำแหน่งตลอดเวลาที่มีการเคลื่อนเมาส์ไปมาบนจอภาพ จากตัวอย่างโปรแกรมในรูปที่ 3.10 สามารถนำมาเขียนใหม่ในรูปแบบภาษาวิชวลเบสิกซึ่งประกอบด้วยการโปรแกรมตามเหตุการณ์ดังรูปที่ 3.11

และหากโปรแกรมนี้อยู่ในขณะการพัฒนาภายใต้สภาพแวดล้อมของตัวแปลภาษาวิชวลเบสิกจะปรากฏในรูปที่ 3.12 ซึ่งจะเห็นได้ว่าในโปรแกรมวิชวลเบสิกจะมีเครื่องมือ (tools) ต่างๆอยู่มากเพื่อช่วยในการสร้างส่วนประสานกับผู้ใช้แบบกราฟิก เมื่อโปรแกรมข้างต้นถูกสั่งให้ทำงานทำงานจะปรากฏผลลัพธ์ดังรูปที่ 3.13 ซึ่งจะเห็นว่าปรากฏเป็นวินโดวส์ขึ้นมา จนกระทั่งเมื่อ

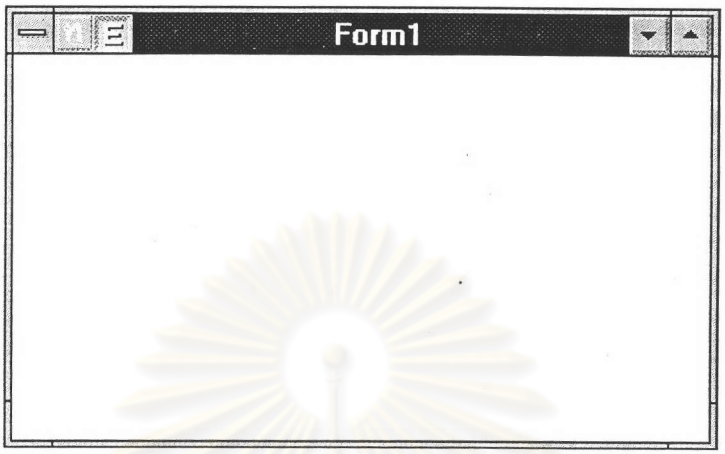
ผู้ใช้ทำการเคลื่อนเมาส์ไปบนพื้นที่ของวินโดวส์แล้วทำการกดปุ่มเมาส์ด้านซ้ายมือโปรแกรมก็จะแสดงผลลัพธ์ดังรูปที่ 3.14

```
'This is a Visval Basic Program'
Sub Form_Click ()
  Dim i% As Integer
  Print "Testing..."
  For i% = 1 To 3
       Print i%
  Next i%
End Sub
```

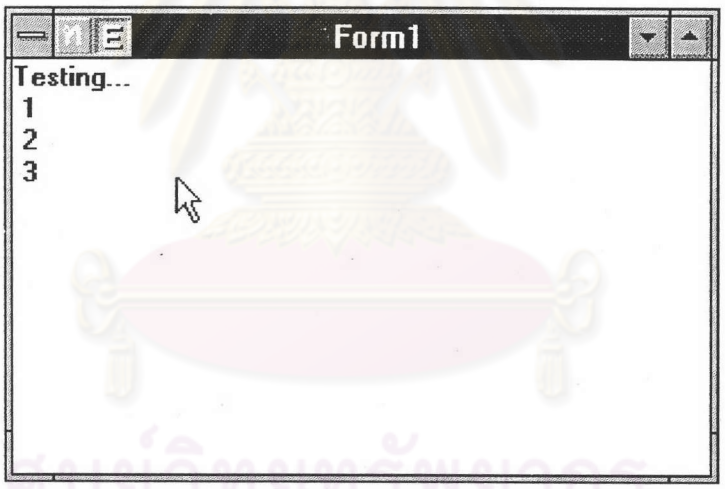
รูปที่ 3.11 ตัวอย่างโปรแกรมในรูปแบบวิชวลเบสิก



รูปที่ 3.12 การพัฒนาภายใต้สภาพแวดล้อมของวิชวลเบสิก



รูปที่ 3.13 ขณะโปรแกรมเริ่มทำงาน



รูปที่ 3.14 ผลจากการกดปุ่มเมาส์ด้านซ้ายมือ