บทที่ 4 การเขียนรหัสเทียม (Pseudo Code)

สาระการเรียนรู้

1. ความหมายของการเขียนรหัสเทียม
2. ประโยชน์ของการเขียนรหัสเทียม
3. หลักในการเขียนรหัสเทียม
4. พื้นฐานการเขียนรหัสเทียม
5. ตัวอย่างการเขียนรหัสเทียม

สมรรถนะการเรียนรู้

1. บอกความหมายของรหัสเทียมได้
2. บอกประโยชน์ของการเขียนรหัสเทียมได้
3. บอกหลักในการเขียนรหัสเทียมได้
4. เขียนรหัสเทียมกำหนดค่าได้
5. เขียนรหัสเทียมรับข้อมูลเข้าได้
6. เขียนรหัสเทียมแสดงข้อมูลได้
7. เขียนรหัสเทียม ในการคำนวณได้
8. เขียนรหัสเทียมตัดสินใจทางเลือกได้
9. เขียนรหัสเทียมการทำซ้ำได้
10. เขียนรหัสเทียมเรียกใช้โปรแกรมย่อยได้

แผนผังความคิด (Mind Mapping) ของหน่วยการเรียนรู้

รหัสเทียม (Pseudo-code)

รหัสเทียม (Pseudo-code) ใช้เป็นภาษากลางในการอธิบายขั้นตอนการทำงานของโปรแกรมและขั้นตอนวิธี การเขียนรหัสเทียมไม่มีหลักเกณฑ์ตายตัว สำคัญเพียงแต่เขียนให้ผู้อ่านเข้าใจ โดยปกติแล้วจะประยุกต์รูปแบบการเขียนและโครงสร้างมาจากภาษาคอมพิวเตอร์ แต่การเขียนรหัสเทียมจะเป็นลักษณะการเขียนคำอธิบายมากกว่าการเขียนเป็นคำสั่งต่าง ๆ และการเขียนรหัสเทียมนั้นมักจะไม่ใส่ใจในรายละเอียดของการเขียนมากนัก เช่น อาจไม่มีขั้นตอนการประกาศตัวแปร ตำแหน่งของการแสดงผล เป็นต้น เป้าหมายสำคัญของการเขียนรหัสเทียมคือ การสื่อสารระหว่างคนหนึ่งกับอีกคนหนึ่งให้เข้าใจตรงกัน ซึ่งได้แก่การสื่อสารระหว่างนักออกแบบโปรแกรมกับโปรแกรมเมอร์ เป็นต้น

ที่มา :- http://th.wikipedia.org/wiki

คำสั่งเทียม (Pseudo-code)

คำสั่งเทียม	การใช้งาน	ตัวอย่าง
BEGIN	จุดเริ่มต้นของโปรแกรมหรือบล็อก	BEGIN
INITIAL, INIT ASSIGNMENT, SET	ใช้กำหนดค่าเริ่มต้น ใช้กำหนดค่า (Assignment)	INITIAL COUNT = 0 Set N = 5
GET INPUT, ACCEPT READ	รับค่าข้อมูลไปเก็บไว้ในตัวแปรที่ระบุ	GET A, B INPUT N READ X[I]
COMPUTE CALCULATE	ประมวลผล หรือการคำนวณ	COMPUTE SUM = SUM+I CALCULATE ANS = SQRT(X)
COMPARE	เปรียบเทียบ ตัวแปร กับ ตัวแปร หรือข้อมูล	COMPARE A,B

คำสั่งเทียม (Pseudo-code) (ต่อ)

คำสั่งเทียม	การใช้งาน	ตัวอย่าง
INC, INCREMENT DEC, DECREMENT	การเพิ่มค่า เช่น INC I หมายถึง I = I+1 การลดค่า เช่น DEC A หมายถึง A = A-1	INCREMENT I DECREMENT C
DISPLAY SHOW PRINT PROMPT WRITE OUTPUT	แสดงผล, แสดง, พิมพ์, บันทึก, ส่งออก ค่าหรือค่าของตัวแปรหรือค่าผลลัพธ์ของนิพจน์	DISPLAY “Hello ” + “Student” SHOW A+B PRINT “ANS = ” + A PROMPT “Press any key to continue”
IF THEN IF THEN ELSE	คำสั่งตรวจสอบเงื่อนไข และทำตามเงื่อนไข	IF (A > B) THEN ... IF (X < 0) THEN ... ELSE ...
INC, INCREMENT DEC, DECREMENT	การเพิ่มค่า เช่น INC I หมายถึง I = I+1 การลดค่า เช่น DEC A หมายถึง A = A-1	INCREMENT I DECREMENT C
DISPLAY SHOW PRINT PROMPT WRITE OUTPUT	แสดงผล, แสดง, พิมพ์, บันทึก, ส่งออก ค่าหรือค่าของตัวแปรหรือค่าผลลัพธ์ของนิพจน์	DISPLAY “Hello ” + “Student” SHOW A+B PRINT “ANS = ” + A PROMPT “Press any key to continue”

คำสั่งเทียม (Pseudo-code) (ต่อ)

คำสั่งเทียม	การใช้งาน	ตัวอย่าง
DO-WHILE WHILE-DO FOR-NEXT	คำสั่งการทำงานซ้ำ เมื่อเงื่อนไขเป็นจริง	DO ... WHILE (I <= N)
REPEAT-UNTIL UNTIL-DO	คำสั่งการทำงานซ้ำ เมื่อเงื่อนไขเป็นเท็จ	UNTIL (I > N) DO Begin ... End
CALL	การเรียกใช้โปรแกรมย่อยหรือโพซีเยอร์	CALL Report (ชื่อ Sub หรือ Procedure)
REM	การเขียนหมายเหตุ	REM คำอธิบาย... หรือ 'คำอธิบาย...
END, STOP, RETURN	จุดสิ้นสุด	END / ENDIF / ENDFOR / ENDWHILE / ENDDO
DO-WHILE WHILE-DO FOR-NEXT	คำสั่งการทำงานซ้ำ เมื่อเงื่อนไขเป็นจริง	DO ... WHILE (I <= N)
REPEAT-UNTIL UNTIL-DO	คำสั่งการทำงานซ้ำ เมื่อเงื่อนไขเป็นเท็จ	UNTIL (I > N) DO Begin ... End

ความหมายของรหัสเทียม หรือซูโดโค้ด (Pseudo Code)

การเขียนอัลกอริทึมเป็นการแสดงขั้นตอนวิธีการที่ใช้ภาษาเขียนที่เข้าใจได้ง่ายอาจใช้ภาษาไทยหรือภาษาอังกฤษก็ได้ขึ้นอยู่กับความสะดวกของผู้เขียนและกิจกรรมที่จะนำเสนอ รหัสเทียมเป็นส่วนผสมผสานของการใช้ภาษาธรรมชาติและโปรแกรมภาษา เพื่อใช้ในการสื่อสารระหว่างผู้ใช้หลายกลุ่มที่มีความต้องการใช้ขั้นตอนวิธีต่างกันออกไป

ซูโดโค้ด (Pseudo code) ในภาษาไทย เรียกว่า รหัสจำลองหรือรหัสเทียม คือ การจำลองการเขียนโปรแกรม โดยเขียนคำสั่งเป็นภาษาอังกฤษ ที่ไม่ใช่ภาษาคอมพิวเตอร์ภาษาใดภาษาหนึ่ง การเขียนรหัสเทียมนี้จะไม่ขึ้นกับภาษาคอมพิวเตอร์ใด ๆ โดยเฉพาะ ใช้อธิบายการทำงานของโปรแกรม เนื่องจากสื่อด้วยภาษาง่ายๆ ทำให้ผู้อ่านเข้าใจได้ง่ายและตรงกัน สามารถนำมาแปลเป็นภาษาคอมพิวเตอร์ได้ง่าย

โดยสรุป รหัสเทียมหรือรหัสจำลอง คือรหัสที่ใช้เป็นตัวแทนของอัลกอริทึมโดยมีถ้อยคำหรือประโยคคำสั่งที่เขียนอยู่ในรูปแบบของภาษาอังกฤษที่ไม่ขึ้นกับภาษาคอมพิวเตอร์ใดภาษาหนึ่ง เพื่อใช้อธิบายรายละเอียดของอัลกอริทึม และสื่อให้กับบุคคลอื่นมีความเข้าใจตรงกัน

ประโยชน์ของการเขียนรหัสเทียม (Pseudo code)

- *นำมาใช้ทบทวนความถูกต้องกับสิ่งที่ได้ออกแบบไป*
- *นำมาใช้เพื่อกำหนดงานเขียนโปรแกรม*
- *เพื่อให้โปรแกรมเมอร์นำไปเพิ่มเติมรายละเอียดและใช้เป็นแนวทางในการเขียนโปรแกรมต่อไป*
- *นำมาใช้เป็นเอกสารประกอบโปรแกรม เพื่องานบำรุงรักษาโปรแกรมที่อาจเกิดขึ้นในอนาคต*
- *นำมาใช้เป็นเครื่องมือเพื่อการออกแบบต่าง ๆ ที่เกี่ยวข้องกับเทคนิคเชิงโครงสร้าง*

ที่มา :- http://chummy-online.blogspot.com/2012/10/pseudocode_9.html

หลักในการเขียน Pseudo code

- คำสั่งที่เขียนใช้ภาษาที่เข้าใจง่าย ไม่ต้องคำนึงถึงภาษาคอมพิวเตอร์
- ในหนึ่งบรรทัด ให้มีเพียงหนึ่งคำสั่งเท่านั้น
- ใช้ย่อหน้าในการแสดงกลุ่มของคำสั่ง ที่เป็นคำสั่งย่อยในคำสั่งพวกเงื่อนไข เช่น if-else, while, for เป็นต้น
- ในการเขียนแต่ละคำสั่งให้เรียงการทำงานจากบนลงล่าง และมีทางออกหรือจุดสิ้นสุดเพียงจุดเดียว
- กลุ่มคำสั่งอาจจะเขียนรวมกันเป็นมอดูล และเวลาเรียกใช้ก็เรียกใช้ผ่านชื่อมอดูล

พื้นฐานการเขียนซูโดโค้ด

รหัสเทียมไม่มีกฎในการเขียนตายตัว โดยมากขึ้นอยู่กับความถนัดของผู้ใช้ แต่มีข้อตกลงบางอย่างร่วมกันเป็นสากล ส่วนประกอบที่สำคัญของรหัสเทียมได้แก่ คำสั่งกำหนดค่า คำสั่งรับค่า คำสั่งแสดงผล คำสั่งควบคุม กลุ่มของคำสั่ง และข้อบันทึกหรือคำอธิบาย

แม้ว่าการเขียนซูโดโค้ดจะไม่มีรูปแบบที่แน่นอนแต่โดยทั่วไปแล้วมักทำกันดังลักษณะต่อไปนี้

การกำหนดค่า (Assignments)

การกำหนดค่าเริ่มต้นให้กับตัวแปรจะใช้คำว่า Assignment หรือ INIT หรือ SET ถ้าหากเป็นการประกาศตัวแปรจะต้องระบุด้วยคำว่าตัวแปรประเภทใด และถ้าเป็นการประกาศค่าคงที่จะใช้เครื่องหมายเท่ากับในการกำหนดค่า

ใช้ คีย์เวิร์ด Set, Init สำหรับกำหนดค่าเริ่มต้นให้กับตัวแปร

รูปแบบเดิม

Set Variable ← value

Init Variable ← value

ปัจจุบัน นิยมเขียนเครื่องหมาย = แทนเครื่องหมาย ← จึงเปลี่ยนมาเป็น

Set variable = value

Init variable = value

แต่ก็มีผู้เขียนโปรแกรมยุคใหม่ ตัดคำสั่งเทียมออกไป ใช้เพียงเครื่องหมาย = (เดิมใช้เครื่องหมาย ←) สำหรับกำหนดค่าให้แก่ตัวแปรทางด้านซ้าย เช่น

max ← 100 เป็นการกำหนดค่า 100 ให้กับตัวแปร max

b = 5 เป็นการกำหนดค่าให้กับตัวแปร b ให้มีค่าเป็น 5

การคำนวณ (Compute)

ในการประมวลผลแบบคำนวณจะขึ้นต้นด้วยคำว่า Compute หรือ Calculate แล้วตามด้วยตัวแปรที่ต้องการเก็บค่าจากการคำนวณเครื่องหมายเท่ากับและนิพจน์การคำนวณซึ่งประกอบไปด้วยเครื่องหมายการกระทำทางคณิตศาสตร์ โดยใช้คำสั่ง Calculate หรือ Compute สำหรับประมวลผลข้อมูล

รูปแบบ

เดิมใช้ (แต่ก็ยังมีใช้กันอยู่ในปัจจุบัน)

Calculate Variable ← expression

Compute Variable ← expression

ปัจจุบันใช้เครื่องหมาย = มาแทนเครื่องหมาย ←

Calculate Variable = expression

Compute Variable = expression

เช่น

Calculate Sum = (n+1)*n/2

Compute Average = Sum/n

การคำนวณ (Compute) (ต่อ)

แต่ก็มีผู้เขียนโปรแกรมยุคใหม่ ตัดคำสั่งเทียมออกไป ใช้เพียงเครื่องหมาย = สำหรับกำหนดค่าจากนิพจน์ทางขวาให้แก่ตัวแปรทางด้านซ้าย เช่น

a = a + 1	เป็นการเพิ่มค่า a ขึ้นอีกหนึ่ง
a = n + 10	เป็นการนำค่าของตัวแปร n มาบวก 10 ได้ผลลัพธ์เท่าไรไปเก็บไว้ในตัวแปร a
r3 = sqrt(3)	เป็นการหาค่ารากที่ 2 ของ 3 แล้วเก็บผลลัพธ์ในตัวแปร r3
sum = (n+1)*n/2	เป็นการหาผลรวมของเลขอนุกรมตั้งแต่ 1 ถึง n แล้วเก็บผลรวมไว้ที่ตัวแปร sum

การรับข้อมูลเข้า

ในการรับข้อมูลจะนิยมใช้คำว่า Read หรือ Input หรือ Get ตามด้วยตัวแปรที่ต้องการใช้เก็บข้อมูล ถ้าหากมีตัวแปรหลายตัวจะใช้เครื่องหมายคอมมา (“,”) คั่นระหว่างตัวแปร

ใช้ คำสั่ง Input หรือ Get สำหรับรับข้อมูลแป้นพิมพ์

ใช้ คำสั่ง Read สำหรับอ่านข้อมูลจาก Data ในโปรแกรม หรือการอ่านจากแฟ้มข้อมูล

รูปแบบ

Input Variable_List

Get Variable_List

Read Variable_List

เช่น

Read X, Yอ่านข้อมูลจาก Data ในโปรแกรมมาเก็บไว้ที่ตัวแปร X และ Y

Input N รับข้อมูลจากแป้นพิมพ์มาเก็บไว้ที่ตัวแปร N

Get Key รับการกดคีย์จากแป้นพิมพ์ นำรหัสคีย์ที่กดมาไว้ในตัวแปร Key

การแสดงผลข้อมูล

การแสดงผลมักใช้คำว่า Print หรือ Display และ Write ตามด้วยตัวแปรที่ต้องการนำมาแสดงผล ถ้าหากมีตัวแปรหลายตัวจะใช้เครื่องหมายคอมมา (“,”) คั่นระหว่างตัวแปร

ใช้คำสั่ง Display หรือ Showสำหรับแสดงผลออกทางจอภาพ

ใช้คำสั่ง Print สำหรับพิมพ์ออกเครื่องพิมพ์ (แต่บางคนใช้แสดงผลทางจอแทน)

ใช้คำสั่ง Write สำหรับเขียนหรือบันทึกข้อมูลลงแฟ้มข้อมูล

รูปแบบ

```
Display     Variable_List
Show        Variable_List
Print       Variable_List
Write       Variable_List
```

เช่น

```
Total = 10
N = 5
Print “Total = ”, total     - พิมพ์ Total = 10 ออกทางเครื่องพิมพ์
Show “Total = ”, total      - แสดง Total = 10 ออกทางจอภาพ
Display N                   - แสดงเลข 5 ออกทางจอภาพ
Write total                 - บันทึกข้อมูลที่อยู่ตัวแปร total ลงในแฟ้มข้อมูล
```

กลุ่มของคำสั่ง (Blocks of Statements)

คำสั่งอาจมีการรวมกลุ่มเพื่อให้ง่ายต่อการเข้าใจโดยขึ้นต้นกลุ่ม (Block) ด้วยคำว่า Begin และสิ้นสุดด้วยคำว่า End ทั้งนี้แต่ละบรรทัดของคำสั่งจะมีการย่อหน้าที่เท่ากันด้วย

เช่น

```
Begin
1       b = i
2       n = sqrt(b)
3       q = b/n
End
```

กลุ่มของคำสั่ง (ต่อ)

กลุ่มของประโยคคำสั่ง (Statements) จะต้องเขียนอยู่ภายใน Block นี้เสมอ คำสั่งที่ Block คือกลุ่มคำสั่งควบคุมซึ่งได้แก่ คำสั่ง if คำสั่ง For และคำสั่งการทำงานแบบวนรอบ (Loop) ทั้งหลาย นอกจากนี้แล้ว อาจพบการนำรูปแบบของภาษาเบสิกมาใช้แทน Block เช่น คำสั่ง

If (condition) Then

ประโยคคำสั่งหรือกลุ่มประโยคคำสั่ง

End if

Then จะแทน Begin และ End if จะแทน End ไปในตัวอยู่แล้ว ก็ไม่จำเป็นต้องมี Begin และ End อีก แต่ถ้าไม่มี End if ก็ต้องเขียนกลุ่มประโยคคำสั่งอยู่ภายใน Block (begin - end)

การตัดสินใจทางเลือก

คำสั่ง IF เป็นคำสั่งเงื่อนไข

การทำงานเริ่มจากเงื่อนไขจะถูกตรวจสอบก่อน ถ้าเป็นจริงแล้วคำสั่ง(หรือกลุ่มของคำสั่ง) ที่กำหนดไว้จะถูกปฏิบัติต่อไป โดยอาจมีทางเลือกด้วยว่าถ้าเงื่อนไขไม่เป็นจริง จะปฏิบัติคำสั่ง(หรือกลุ่มของคำสั่ง) อื่นแทน คำสั่งเงื่อนไขมี 2 รูปแบบ คือ

- แบบเลือกทำกับไม่ทำ

IF (เงื่อนไข) Then

ประโยคคำสั่ง(หรือกลุ่มของประโยคคำสั่ง) ที่ต้องปฏิบัติเมื่อเงื่อนไขเป็นจริง

เช่น

```
IF (max < a) Then
        max = a
```

การตัดสินใจทางเลือก

- แบบเลือกทำอย่างใดอย่างหนึ่ง

If (เงื่อนไข) Then

ประโยคคำสั่ง(หรือกลุ่มของประโยคคำสั่ง) ที่ต้องปฏิบัติเมื่อเงื่อนไขเป็นจริง

Else

ประโยคคำสั่ง(หรือกลุ่มของประโยคคำสั่ง) ที่ต้องปฏิบัติเมื่อเงื่อนไขไม่เป็นจริง

เช่น

```
If (a <> b) Then
        Display "Not Equal"
Else
        Display "Equal"
```

ตัวอย่างการคำนวณระดับคะแนนหรือการคิดเกรด

```
1       Read  score
2       If (score >= 80 And score <= 100) Then
        1       grade = 4
        Else if (score >= 70 And score <= 79) Then
        1       grade = 3
        Else if (score >= 60 And score <= 69) Then
        1       grade = 2
        Else if (score >= 50 And score <= 59) Then
        1       grade = 1
        Else
        1       grade = 0
        End if
3       Display  grade
```

การทำซ้ำหรือการวนรอบ (Loop)

ในการทำซ้ำหมายความว่าให้ระบบทำงานซ้ำ ๆ ตามเงื่อนไขที่กำหนด โดยจะมีการเปรียบเทียบเงื่อนไขในการทำซ้ำ แบ่งออกได้ 5 รูปแบบ คือ

1. คำสั่งทำซ้ำตามดัชนี (For Loop)
2. คำสั่งทำซ้ำในขณะที่เงื่อนไขเป็นจริงแบบตรวจสอบเงื่อนไขก่อน (While-Do)
3. คำสั่งทำซ้ำในขณะที่เงื่อนไขเป็นจริงแบบตรวจสอบเงื่อนไขภายหลัง (Do-While)
4. คำสั่งทำซ้ำในขณะที่เงื่อนไขเป็นเท็จแบบตรวจสอบเงื่อนไขก่อน (Until-Do)
5. คำสั่งทำซ้ำในขณะที่เงื่อนไขเป็นเท็จแบบตรวจสอบเงื่อนไขภายหลัง (Repeat-Until)

คำสั่งทำซ้ำตามดัชนี (For Loop)

เริ่มต้นด้วยการกำหนดค่าของดัชนีให้มีค่าเท่ากับ ค่าเริ่มต้น แล้วจึงปฏิบัติคำสั่ง (หรือกลุ่มของคำสั่ง) จนสิ้นสุด จากนั้นค่าของดัชนีจะถูกเพิ่มขึ้นอีกครั้งละ 1 สลับกับการปฏิบัติคำสั่ง (หรือกลุ่มของคำสั่ง)

คำสั่งทำซ้ำตามดัชนีจะสิ้นสุดก็ต่อเมื่อตัวแปรที่เป็นดัชนีมีค่ามากกว่าค่าสุดท้าย

คำสั่งทำซ้ำตามดัชนีมีรูปแบบคือ

For ตัวแปรที่เป็นดัชนี = ค่าเริ่มต้น to ค่าสุดท้าย Step ค่าเพิ่ม Do

คำสั่ง(หรือกลุ่มของคำสั่ง)ที่ต้องปฏิบัติ

หมายเหตุ ถ้าไม่ระบุค่าเพิ่ม(คือไม่มี Step ค่าเพิ่ม) ค่าเพิ่มจะมีค่าเป็น +1 (เป็นค่าโดยปริยาย(Default) ของ Step)

ตัวอย่าง

การหาผลรวมตั้งแต่ 1 - N (Sum = 1 + 2 + 3 + ... + N)

```
1       Input N
2       Sum = 0
3       For i = 1 to N Do
            1       Sum = Sum + i
4       Display Sum
```

ตัวอย่าง

*การหาค่าแฟคตอเรียล N (Fact = 1 * 2 * 3 * ... * N)*

```
1       Input N
2       Fact = 1
3       For j = 1 to N Do
                1       Fact = Fact * j
4       Display Fact
```

คำสั่งทำซ้ำในขณะที่เงื่อนไขเป็นจริง

แบบตรวจสอบเงื่อนไขก่อน (While-Do)

เริ่มต้นด้วยการตรวจสอบเงื่อนไขก่อนถ้าเป็นจริง คำสั่ง(หรือกลุ่มของคำสั่ง)ที่กำหนดไว้จะถูกปฏิบัติต่อไปจนสุด จากนั้นเงื่อนไขจะถูกตรวจสอบอีก ถ้าเป็นจริงแล้ว คำสั่ง(หรือกลุ่มของคำสั่ง) ที่กำหนดไว้จะถูกปฏิบัติต่อไปจนสิ้นสุดอีกครั้งหนึ่ง คำสั่งทำซ้ำในขณะที่เงื่อนไขเป็นจริงนี้ จะสิ้นสุดก็ต่อเมื่อตรวจสอบเงื่อนไขแล้วพบว่าไม่จริงหรือเป็นเท็จ

คำสั่งทำซ้ำในขณะที่เงื่อนไขเป็นจริงมีรูปแบบ คือ

```
While (เงื่อนไข) Do
        คำสั่งที่ต้องการทำซ้ำ
```

ตัวอย่าง

ต้องการหาผลรวมของตัวเลขตั้งแต่ 1 ถึง 100

โดยใช้คำสั่ง While-Do

```
1       Sum = 0, N = 1
2       While (N <= 100) Do
        Begin
        1       Sum = Sum+N
        2       N = N+1
        End
3       Display  Sum
```

คำสั่งทำซ้ำในขณะที่เงื่อนไขเป็นจริง

แบบตรวจสอบเงื่อนไขภายหลัง (Do-While)

เริ่มต้นด้วย ทำตามคำสั่ง(หรือกลุ่มของคำสั่ง) ที่กำหนดไว้จะถูกปฏิบัติตามลำดับก่อน จากนั้นจึงตรวจสอบเงื่อนไข ถ้าเงื่อนไขเป็นจริง คำสั่ง(หรือกลุ่มของคำสั่ง) ที่กำหนดไว้ก็จะถูกปฏิบัติซ้ำอีกไปจนกว่าการตรวจสอบเงื่อนไขเป็นเท็จ ก็จะเป็นการสิ้นสุดการทำซ้ำ

คำสั่งทำซ้ำในขณะที่เงื่อนไขเป็นจริงแบบตรวจสอบเงื่อนไขภายหลัง มีรูปแบบคือ

Do คำสั่งที่ต้องการทำซ้ำ

While (เงื่อนไข)

ตัวอย่าง

ต้องการหาผลรวมของตัวเลขตั้งแต่ 1 ถึง 100

โดยใช้คำสั่ง Do-While

```
1       Sum = 0,  N = 1
2       Do
            1       Sum = Sum+N
            2       N = N+1
            While (N <= 100)
3       Display Sum
```

คำสั่งทำซ้ำในขณะที่เงื่อนไขเป็นเท็จ

แบบตรวจสอบเงื่อนไขก่อน (Until-Do)

เริ่มต้นด้วยการตรวจสอบเงื่อนไขก่อนถ้าเป็นเท็จ คำสั่ง(หรือกลุ่มของคำสั่ง) ที่กำหนดไว้จะถูกปฏิบัติต่อไปจนสุด จากนั้นเงื่อนไขจะถูกตรวจสอบอีก ถ้าเป็นเท็จ คำสั่ง(หรือกลุ่มของคำสั่ง) ที่กำหนดไว้จะถูกปฏิบัติต่อไปซ้ำอีกครั้งหนึ่ง คำสั่งทำซ้ำในขณะที่เงื่อนไขเป็นเท็จนี้ จะสิ้นสุดก็ต่อเมื่อตรวจสอบเงื่อนไขแล้วพบว่าเงื่อนไขเป็นจริง

คำสั่งทำซ้ำในขณะที่เงื่อนไขเป็นเท็จแบบตรวจสอบเงื่อนไขก่อนมีรูปแบบคือ

```
Until (เงื่อนไข) Do
        คำสั่งที่ต้องการทำซ้ำ
```

ตัวอย่าง

ต้องการหาผลรวมของตัวเลขตั้งแต่ 1 ถึง 100

โดยใช้คำสั่ง Until-Do

```
1       Sum = 0
2       N = 1
3       Until (N > 100) Do
        Begin
        1       Sum = Sum+N
        2       N = N+1
        End
4       Display Sum
```

คำสั่งทำซ้ำในขณะที่เงื่อนไขเป็นเท็จ

แบบตรวจสอบเงื่อนไขภายหลัง (Repeat-Until)

เริ่มต้นด้วย ทำตามคำสั่ง(หรือกลุ่มของคำสั่ง) ที่กำหนดไว้จะถูกปฏิบัติตามลำดับก่อน จากนั้นจึงตรวจสอบเงื่อนไข ถ้าเงื่อนไขเป็นเท็จ คำสั่ง(หรือกลุ่มของคำสั่ง) ที่กำหนดไว้ก็จะถูกปฏิบัติซ้ำอีกไปจนกว่าการตรวจสอบเงื่อนไขเป็นจริง ก็จะเป็นการสิ้นสุดการทำซ้ำ

คำสั่งทำซ้ำในขณะที่เงื่อนไขเป็นเท็จแบบตรวจสอบเงื่อนไขภายหลัง มีรูปแบบคือ

```
Repeat
        คำสั่งที่ต้องการทำซ้ำ
Until (เงื่อนไข)
```

ตัวอย่าง

ต้องการหาผลรวมของตัวเลขตั้งแต่ 1 ถึง 100

โดยใช้คำสั่ง Repeat-Until

```
1       Sum = 0
2       N = 1
3       Repeat
            1       Sum = Sum+N
            2       N = N+1
        Until (N > 100)
4       Display Sum
```

การเรียกโปรแกรมย่อยและกระโดดข้าม

สำหรับการเรียกโปรแกรมย่อยหรือโพซีเยอร์ ใช้คำว่า CALL แล้วตามด้วยชื่อโปรแกรมย่อยหรือโพซีเยอร์ ในส่วนการเขียนรายละเอียดของโปรแกรมย่อย จะขึ้นต้นด้วย Sub แล้วตามด้วยชื่อของโปรแกรมย่อยนั้น และสิ้นสุดด้วย End Sub และระหว่าง Sub กับ End Sub ก็คือกลุ่มคำสั่งการทำงาน

การกระโดดข้ามไปทำชุดคำสั่งใด ๆ จะใช้ชื่อ (Label) กำหนดตำแหน่งที่จะกระโดดมาและใช้คำสั่ง Goto ในตำแหน่งที่จะกระโดดไปที่ชื่อลาเบลที่ระบุ แต่ในปัจจุบันไม่ค่อยนิยมใช้คำสั่งนี้ เพราะถ้าหากมีการใช้คำสั่ง Goto มากไปจะทำให้เกิดโอกาสที่จะเขียนอัลกอริทึมผิดพลาดได้ง่าย เนื่องจากเป็นรูปแบบโครงสร้างที่ไม่เป็นระเบียบของโปรแกรมย่อยนั้น

(ดูตัวอย่างที่ 8 ในหัวข้อ 4.6)

ตัวอย่างการเขียนรหัสเทียม

จงเขียน Pseudo code จากโปรแกรมหาผลรวมของตัวเลข 2 ค่า แล้วแสดงผลรวมออกทางหน้าจอ

```
Algorithm Sum of 2 Numbers
Begin
1   Read  X, Y
2   Calculate Sum = X + Y
3   Display Sum
End
```

ตัวอย่างการเขียนรหัสเทียม

จงเขียน Pseudo code จากโปรแกรมหาผลรวมของตัวเลขอนุกรม 1+2+...+N แล้วแสดงผลรวมออกทางหน้าจอ

Algorithm Sum of Serial Numbers 1 to N

Begin

1 Input N

2 Calculate Sum = (N+1)*N/2

3 Display Sum

End

ตัวอย่างการเขียนรหัสเทียม

จงเขียน Pseudo code จากโปรแกรมรับตัวเลข 1 ค่าแล้วตรวจสอบว่าเป็นเลขคู่หรือเลขคี่

```
Algorithm Sum of 2 Numbers
Begin
1    Input X
2    If (X % 2 = 0) Then
       1   Display "Even"
     Else
       1   Display "Odd"
End
```

ตัวอย่างการเขียนรหัสเทียม

จงเขียน Pseudo code จากโปรแกรมหาผลรวมของตัวเลขอนุกรม 1+2+...+N โดยใช้คำสั่ง Do-While แล้วแสดงผลรวมออกทางหน้าจอ

```
Algorithm Sum of Serial Numbers 1 to N
Begin
1    Init  sum = 0
2    Input N
3    Set I = 1
4    Do
            1   Calculate sum = sum+I
            2   Increment I
     While (I <= N)
5    Display sum
End
```

ตัวอย่างการเขียนรหัสเทียม

จงเขียน Pseudo code จากโปรแกรมหาค่าเฉลี่ยของตัวเลข n ค่า แล้วแสดงค่าเฉลี่ยออกทางหน้าจอ

```
Algorithm Find Mean of Data
Begin
1    Init sum = 0
2    Input  n
3    For i = 1 to n
          1  Input num
          2  Calculate sum = sum + num
4    Calculate mean = sum / n
5    Display  mean
End
```

ตัวอย่างการเขียนรหัสเทียม

จงเขียน Pseudo code จากโปรแกรมเปรียบเทียบตัวเลข 2 ค่า แล้วแสดงความสัมพันธ์ออกมา

```
Algorithm Compare two number
Begin
1       Input x, y
2       If  (x > y)  Then
            1   Display "x > y"
            Else
        1   If  (x < y)  Then
                        1           Display "x < y"
                        Else
                        1           Display "x = y"
End
```

ตัวอย่างการเขียนรหัสเทียม

จงเขียน Pseudo code คำนวณหาค่าระดับคะแนน โดยมีเกณฑ์ ดังนี้

- คะแนน 80 - 100 ได้ระดับคะแนน 4
- คะแนน 70 - 79 ได้ระดับคะแนน 3
- คะแนน 60 - 69 ได้ระดับคะแนน 2
- คะแนน 50 - 59 ได้ระดับคะแนน 1
- คะแนน 0 - 49 ได้ระดับคะแนน 0

ตัวอย่างการเขียนรหัสเทียม

```
Algorithm Grade Calculate
Begin
1       Input Score
2       If  (Score>=80 And Score<=100)  Then
            1       Set Grade = 4
            Else
            1       If  (Score>=70 And Score<=79)  Then
                        1       Set Grade = 3
                        Else
            1           If  (Score>=60 And Score<=69)  Then
                                1       Set Grade = 2
                                Else
                        1       If  (Score>=50 And Score<=59)  Then
                                1       Set Grade = 1
                                        Else
                                1       Set Grade = 0
3       Display Grade
End
```

ตัวอย่างการเขียนรหัสเทียม

จงเขียน Pseudo code จากโปรแกรมหาค่า 2!, 3!, 4!, ... 10! โดยวิธีเรียกใช้โปรแกรมย่อย

```
Algorithm Call Factorial Subroutine
Begin
1           Procedure Fact(N)
            1   Set Fact = 1
            2   Set I = 1
            3   While (I <= N) Do
                1   Calculate Fact = Fact*I
                2   Increment I
            4   Display N, "! = ", Fact, NEWLINE
            5   Return
2           End Procedure

3           Begin
            1   For I = 2 to 10
                1   Call Fact(I)
4           End
End
```

สรุป

ความหมายของรหัสเทียม

ซูโดโค้ดเป็นคำอธิบายขั้นตอนการทำงานของโปรแกรม โดยใช้ถ้อยคำผสมผสานระหว่างภาษาอังกฤษและภาษาโปรแกรมแบบโครงสร้างที่เข้าใจง่ายมาแสดงลำดับการทำงานของโปรแกรม โดยช่วยให้ผู้เขียนโปรแกรมสามารถพัฒนาขั้นตอนต่าง ๆ ให้เป็นโปรแกรมได้ง่ายขึ้น

สรุป

พื้นฐานการเขียนซูโดโค้ด

แม้ว่าการเขียนซูโดโค้ดจะไม่มีรูปแบบที่แน่นอนแต่โดยทั่วไปแล้วมักทำกันดังลักษณะต่อไปนี้

- **การกำหนดค่า**

การกำหนดค่าเริ่มต้นให้กับตัวแปรจะใช้คำว่า INIT และ SET ถ้าหากเป็นการประกาศตัวแปรจะต้องระบุด้วยคำว่าตัวแปรประเภทใด และถ้าเป็นการประกาศค่าคงที่จะใช้เครื่องหมายเท่ากับในการกำหนดค่า

- **การคำนวณ**

ในการประมวลผลแบบคำนวณจะขึ้นต้นด้วยคำว่า Compute หรือ Calculate แล้วตามด้วยตัวแปรที่ต้องการเก็บค่าจากการคำนวณเครื่องหมายเท่ากับและนิพจน์การคำนวณซึ่งประกอบไปด้วยเครื่องหมายการกระทำทางคณิตศาสตร์

- **การรับข้อมูลเข้าและการแสดงผลข้อมูล**

ในการรับข้อมูลจะนิยมใช้คำว่า Read หรือ Input ตามด้วยตัวแปรที่ต้องการใช้เก็บข้อมูลถ้าหากมีตัวแปรหลายตัวจะใช้เครื่องหมายคอมมา (“,”) คั่น ส่วนการแสดงผลมักใช้คำว่า Print หรือ Write สำหรับการแสดงผลข้อมูล

สรุป

ในการตัดสินใจจะต้องมีการเปรียบเทียบเงื่อนไขเกิดขึ้น โดยผลลัพธ์ที่จะเป็นจริงหรือเท็จอย่างใดอย่างหนึ่งเท่านั้น

การตัดสินใจเพื่อเลือกทำระหว่างทางสองทางจะใช้คำว่า IF หรือ IF-THEN-ELSE และ ENDIF โดยจะเปรียบเทียบเงื่อนไข ถ้าเงื่อนไขเป็นจริงจะทำกลุ่มคำสั่ง กลุ่มหนึ่ง ถ้าเป็นเท็จจะทำกลุ่มคำสั่งอีกกลุ่มหนึ่ง

สรุป

การเขียนซูโดโค้ดแบบวนซ้ำ

ในการทำซ้ำหมายความว่าให้ระบบทำงานซ้ำ ๆ ตามเงื่อนไขที่กำหนดโดยจะมีการเปรียบเทียบเงื่อนไขในการทำซ้ำ แบ่งออกได้ 5 รูปแบบดังนี้

1. คำสั่งทำซ้ำตามดัชนี (For Loop)
2. คำสั่งทำซ้ำในขณะที่เงื่อนไขเป็นจริงแบบตรวจสอบเงื่อนไขก่อน (While-Do)
3. คำสั่งทำซ้ำในขณะที่เงื่อนไขเป็นจริงแบบตรวจสอบเงื่อนไขภายหลัง (Do-While)
4. คำสั่งทำซ้ำในขณะที่เงื่อนไขเป็นเท็จแบบตรวจสอบเงื่อนไขก่อน (Until-Do)
5. คำสั่งทำซ้ำในขณะที่เงื่อนไขเป็นเท็จแบบตรวจสอบเงื่อนไขภายหลัง (Repeat-Until)

สรุป

การเขียนซูโดโค้ดเพื่อเรียกโปรแกรมย่อยและกระโดดข้าม

สำหรับการเรียกโปรแกรมย่อย (Sub Program) หรือโพซีเยอร์ (Procedure) ใช้คำว่า CALL แล้วตามด้วยชื่อโปรแกรมย่อยหรือโพซีเยอร์

การกระโดดข้ามไปทำชุดคำสั่งใด ๆ จะใช้ชื่อฉลาก (Label) กำหนดตำแหน่งที่จะกระโดดมาและใช้คำว่า Goto ในตำแหน่งที่จะกระโดด แต่ในปัจจุบัน ไม่นิยมใช้คำสั่ง Goto การเขียนเพราะโครงสร้างไม่เป็นระเบียบ และมีโอกาสเกิดความผิดพลาดได้ง่าย

จบการนำเสนอ

บทที่ 4
การเขียนรหัสเทียม (Pseudo Code)