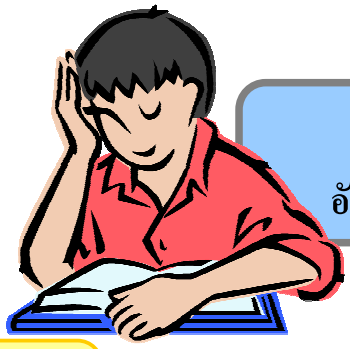


ใบความรู้ที่ 8.1
อัลกอริทึม (Algorithms)

อัลกอริทึม

คอมพิวเตอร์จะสามารถทำการประมวลผลงานใด ๆ ได้ก็ต่อเมื่อมีการป้อนชุดคำสั่งที่เขียนขึ้นด้วยภาษาโปรแกรม ดังนั้นการประมวลผลในคอมพิวเตอร์จึงจำเป็นต้องมีชุดคำสั่งที่สั่งให้คอมพิวเตอร์ทำงานด้วยการกำหนดขั้นตอนของชุดคำสั่งอย่างมีระบบและมีความแน่นอน เพื่อให้ได้ผลลัพธ์ตามต้องการ

จากข้อมูลข้างต้น จึงกล่าวได้ว่า คอมพิวเตอร์จะทำงานตามคำสั่งที่เราได้สั่งให้ทำงานเท่านั้น ซึ่งแตกต่างกับมนุษย์ที่สามารถทำการตีความและสามารถตัดสินใจว่าจะต้องทำอย่างไรต่อไป ยกตัวอย่างเช่น อาจารย์ผู้สอนได้สั่งรายงานเรื่องหนึ่งให้นักเรียนทำ ด้วยคำสั่งเพียงเท่านี้ตัวเราก็จะเข้าใจแล้วว่าจะต้องปฏิบัติอย่างไรต่อไป ไม่ว่าจะเป็นการจัดกลุ่มผู้ร่วมทำรายงาน การแบ่งงานภายในกลุ่ม การค้นคว้าหาหัวข้อรายงานตามห้องสมุด หอสมุดแห่งชาติ หรือในอินเทอร์เน็ต แต่คอมพิวเตอร์หาเป็นเช่นนั้นไม่ เนื่องจากคอมพิวเตอร์นั้นไม่มีความสามารถเทียบเท่าสมองของมนุษย์เราได้ทั้งหมด ดังนั้นเมื่อมีการสั่งให้คอมพิวเตอร์ทำงานจึงจำเป็นต้องมีขั้นตอนที่ละเอียดและชัดเจนและมีวิธีการที่แน่นอน ไม่คลุมเครือหรือตีความได้หลายความหมาย

ขอบเขตของปัญหาต่าง ๆ สามารถนำคอมพิวเตอร์มาช่วยแก้ไขปัญหาดังกล่าวได้ และบ่อยครั้งที่มักนำคอมพิวเตอร์มาช่วยแก้ปัญหาทางคณิตศาสตร์ที่มีความซับซ้อน ดังนั้นจึงเกิดวิธีคิดที่เกิดจากตัวเราเพื่อหาแนวทางแก้ไขปัญหาด้วยคอมพิวเตอร์ กลุ่มความคิดเหล่านั้นเมื่อรวมกันก็จะกลายเป็น **กระบวนการแก้ปัญหาเชิงอัลกอริทึม**

ดังนั้นปัญหาเชิงอัลกอริทึม จึงหมายถึงปัญหาต่าง ๆ ที่ผลลัพธ์สามารถแสดงได้ด้วยการกระทำตามคำสั่งไปทีละขั้นตอนได้ ส่วนอัลกอริทึมนั้นสามารถนิยามความหมายได้ดังนี้

อัลกอริทึม หมายถึง กลุ่มของขั้นตอน (list of steps / a set of rules) หรือกฎเกณฑ์ที่จะนำพาไปสู่การแก้ปัญหาได้

อัลกอริทึม หมายถึง ขั้นตอนวิธี ซึ่งจะอธิบายว่างาน ๆ นั้นทำอย่างไร โดยจะประกอบด้วยชุดลำดับเป็นขั้นตอนที่ชัดเจน และรับประกันว่าเมื่อได้ปฏิบัติถูกต้องตามขั้นตอนจนจบ ก็จะได้ผลลัพธ์ที่ถูกต้องตามต้องการ

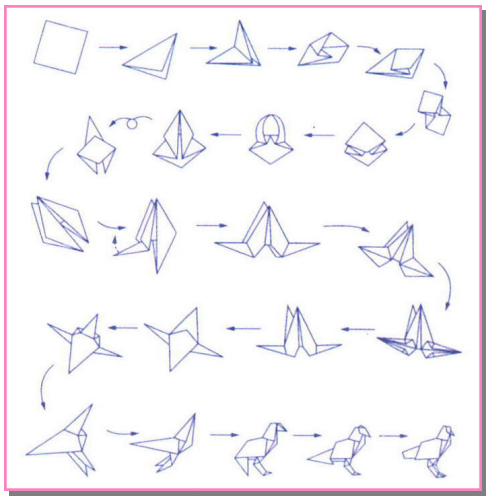
และโดยปกติในชีวิตประจำวันของมนุษย์เราก็มักจะข้องเกี่ยวกับปัญหาเชิงอัลกอริทึมและอัลกอริทึมเป็นกิจวัตรอยู่แล้ว ซึ่งอาจข้องเกี่ยวกับคอมพิวเตอร์หรือไม่ก็ตาม ยกตัวอย่างเช่น การล้างรถและการเคลือบสีรถ

อัลกอริทึมได้รับการออกแบบเพื่อใช้สำหรับในการแก้ไขปัญหาในทุก ๆ กรณีของขอบเขตปัญหานั้น ๆ อีกทั้งอัลกอริทึมหนึ่งก็อาจประกอบไปด้วยหลาย ๆ อัลกอริทึมประกอบกันเพื่อใช้สำหรับแก้ไขปัญหาเหล่านั้น และก็ใช่ว่าอัลกอริทึมทุกประเภทจะใช้คอมพิวเตอร์กระทำได้ทั้งหมด เพราะคอมพิวเตอร์มีความสามารถเฉพาะงานด้วยการปฏิบัติคำสั่งในแต่ละขั้นตอนที่ตัวคอมพิวเตอร์เองเข้าใจและสามารถจะกระทำได้เท่านั้น ดังนั้นเราจึงจำเป็นต้องเรียนรู้คำสั่งเพื่อที่ได้ทราบว่ามีคำสั่งใดที่คอมพิวเตอร์สามารถรับรู้และเข้าใจเพื่อกระทำการนั้น ๆ ได้เพื่อที่จะได้ทำการสร้างหรือออกแบบอัลกอริทึมมาจากคำสั่งเหล่านั้นเพื่อแก้ปัญหาดังกล่าว

ปัญหาเชิงอัลกอริทึม ▸ การล้างรถ
อัลกอริทึม ▸ รูปแบบการล้างรถ
ขั้นตอนอัลกอริทึม ▸ ฉีดน้ำล้างรถให้ทั่วเพื่อขจัดฝุ่นและเศษดินทรายต่าง ๆ ออก ผสมแชมพูล้ารถ 1 ฝา ต่อน้ำครึ่งถัง นำฟองน้ำชุบจ้ำที่สมแชมพูเช็ดทำความสะอาดให้ทั่ว ฉีดน้ำล้างให้สะอาด ใช้ผ้านุ่ม ๆ หรือผ้าชามัวร์ที่สะอาดเช็ดให้แห้ง

ปัญหาเชิงอัลกอริทึม ▸ การเคลือบสีรถ
อัลกอริทึม ▸ รูปแบบการเคลือบสีรถ
ขั้นตอนอัลกอริทึม ▸ ล้างรถ (นำอัลกอริทึมรูปแบบการล้างรถมาใช้) นำฟองน้ำจุ่มน้ำยาเคลือบสีรถ ป้ายบนตัวถังรถด้วยการรวนเป็นก้อนหอยให้ทั่วทั้งตัวถังรถ ปล่อยให้แห้งสักพักหนึ่ง เช็ดด้วยผ้านุ่ม ๆ ที่สะอาด

อย่างไรก็ตามในบางครั้งอัลกอริทึมก็อาจนำเสนอให้อยู่ในรูปแบบของภาพเพื่อสาธิตขั้นตอน ดังตัวอย่างขั้นตอนวิธีการพับนกจากกระดาษที่แสดงดังภาพที่ 8.1



ภาพที่ 8.1 แสดงขั้นตอนการพับนกจากกระดาษด้วยการสาธิตขั้นตอนในลักษณะของภาพ

คราวนี้ลองมาดูตัวอย่างอัลกอริทึมที่ใช้คอมพิวเตอร์แก้ปัญหา ดังภาพที่ 4.2 ซึ่งเป็นอัลกอริทึมให้แสดงเลขคู่จากเลขจำนวนเต็ม 1 ถึง 20

อัลกอริทึมแสดงเลขคู่จากเลขจำนวนเต็ม 1 ถึง 20

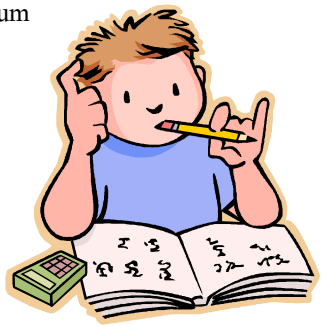
ขั้นตอนที่ 1 กำหนดให้ num มีค่าเท่ากับ 1

ขั้นตอนที่ 2 กำหนดให้ ans = num MOD 2

ขั้นตอนที่ 3 ถ้าคำตอบของ ans เท่ากับศูนย์ ให้แสดงค่า num ทางจอภาพ

ขั้นตอนที่ 4 ตรวจสอบค่า num

- ถ้าค่า num <> 20 ให้ num = num + 1 ไปทำขั้นตอนที่ 2
- ถ้าค่า num = 20 จบการทำงาน



ภาพที่ 8.2 อัลกอริทึมแสดงเลขคู่จากเลขจำนวน 1 ถึง 20

จากภาพที่ 8.2 จะเห็นได้ว่าอัลกอริทึมนั้นเป็นเทคนิคการออกแบบโปรแกรมชนิดหนึ่งที่มิได้มุ่งเน้นภาษาโปรแกรมภาษาใดภาษาหนึ่งโดยเฉพาะ แต่จะระบุขั้นตอนในลักษณะภาพรวมไว้ก่อน ด้วยการระบุเป็นประโยคข้อความเพื่ออธิบาย โดยมุ่งเน้นที่ความกระชับเข้าใจง่าย แต่การออกแบบอัลกอริทึมเพื่อให้คอมพิวเตอร์ช่วยในการแก้ปัญหา จำเป็นต้องแสดงขั้นตอนวิธีต่าง ๆ ในแต่ละขั้นด้วยการบรรยายถึงกระบวนการ รวมทั้งมีการกำหนดค่าตัวแปรต่าง ๆ ซึ่งตัวแปรเหล่านี้จะถูกจัดเก็บในหน่วยความจำของคอมพิวเตอร์ และซีพียูหรือโปรเซสเซอร์ก็จะทำงานตามกระบวนการที่ได้บรรยายไว้ในแต่ละขั้นตอน

วิธีการสร้างอัลกอริทึม

ในการสร้างอัลกอริทึมเพื่อใช้งานทางคอมพิวเตอร์ สามารถสร้างได้หลายวิธีด้วยกัน โดยในที่นี้จะขอกล่าวถึงวิธีต่าง ๆ ดังต่อไปนี้

1. การบรรยาย (Narrative Description)
2. การเขียนผังงาน (Flowchart)
3. การเขียนรหัสจำลอง (Pseudo Code)

การบรรยาย (Narrative Description)

เป็นวิธีที่ว่าด้วยการใช้คำพูดบรรยายเป็นตัวอักษร ซึ่งวิธีนี้จะค่อนข้างง่ายสำหรับตัวผู้เขียน แต่จะยากต่อการนำไปใช้ปฏิบัติจริง เนื่องจากอาจก่อให้เกิดปัญหาต่าง ๆ ไม่ว่าจะเป็นขอบเขตการบรรยายที่กว้างเกินไป ยืดเยื้อเกินไป รวมถึงการบรรยายที่ยากต่อความเข้าใจ

ขั้นตอนการลงทะเบียนเรียน

ในการลงทะเบียนเรียนภาคการศึกษาปกติจะสามารถทำการลงทะเบียนเรียนได้ไม่เกินกว่า 21 หน่วยกิต คอมพิวเตอร์จะต้องทำการสะสมค่าหน่วยกิตของแต่ละรายวิชาที่นักศึกษาลงทำเบียนเรียน หากมีการลงทะเบียนเรียนเกินกว่านั้นก็จะต้องแสดงข้อความเตือน ส่วนการลงทะเบียนเรียนภาคฤดูร้อนก็เป็นไปได้ในลักษณะเดียวกัน แต่จำนวนหน่วยกิตสูงสุดที่สามารถลงทะเบียนได้คือ 9 หน่วยกิต

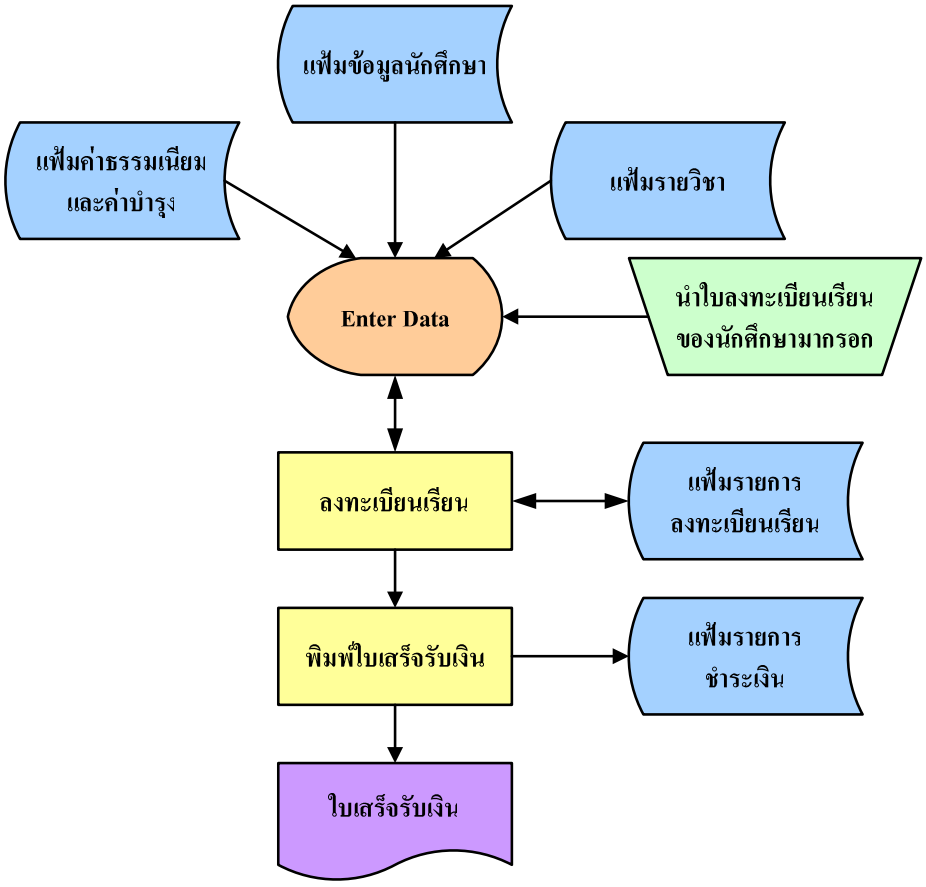
ภาพที่ 8.3 อัลกอริทึมที่เขียนในรูปแบบการบรรยาย

การเขียนผังงาน (Flowchart)

ผังงานจะเป็นการนำเสนอในรูปแบบของแผนภาพ ซึ่งจะประกอบด้วยสัญลักษณ์ต่าง ๆ ที่ได้มีการกำหนดไว้เป็นมาตรฐาน ทำให้แสดงรายละเอียดของขั้นตอนต่าง ๆ ได้ชัดเจนกว่าแบบวิธีการบรรยายในลักษณะคำพูด ผังงานยังแบ่งออกเป็นผังงานระบบ (System Flowchart) และผังงานโปรแกรม (Program Flowchart)

ผังงานระบบ (System Flowchart)

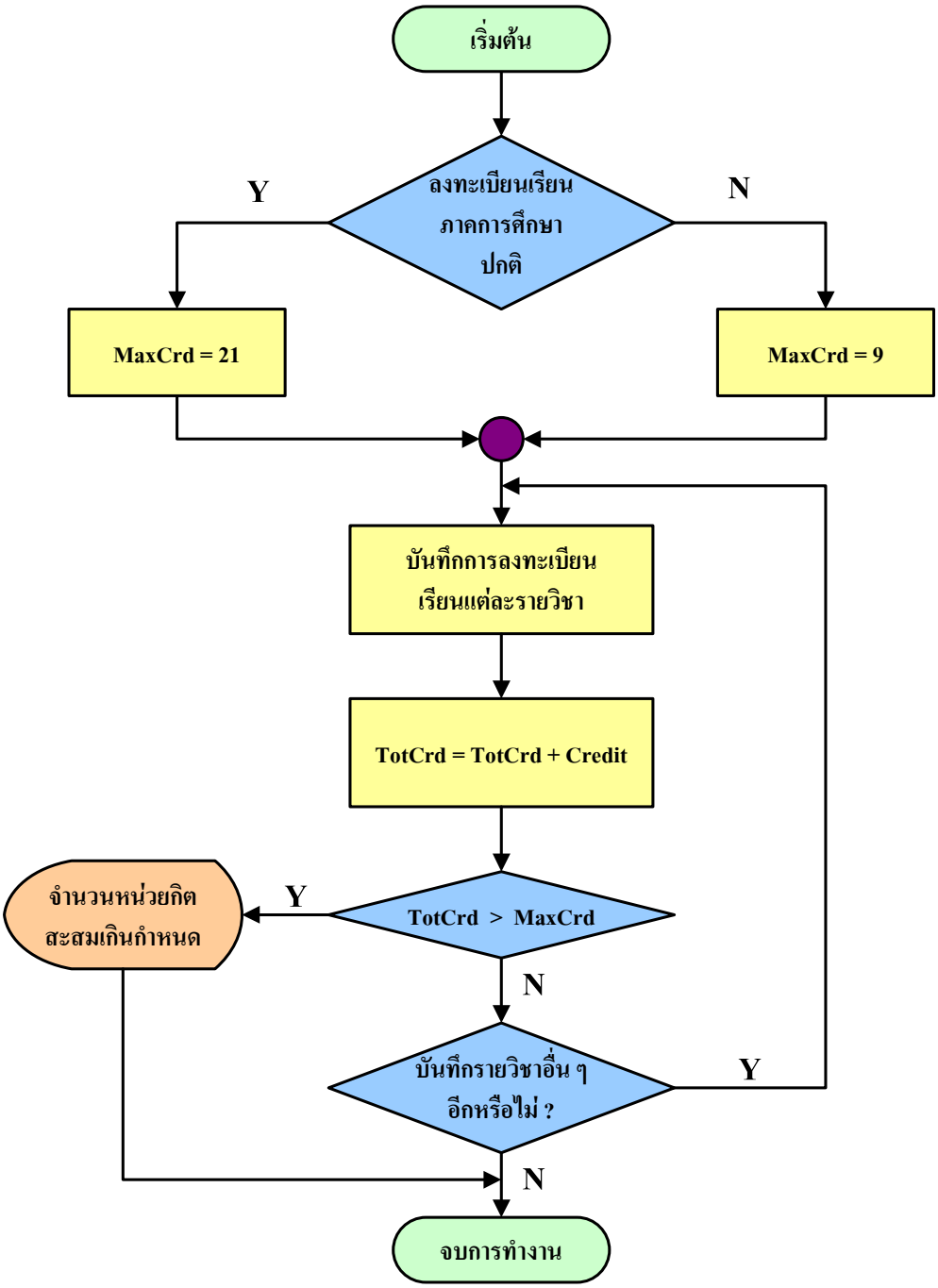
เป็นยังงานที่แสดงขั้นตอนการปฏิบัติงานหลัก ๆ โดยจะแสดงถึงตัวงานหลักที่จะต้องทำในระบบ ซึ่งเป็นการมุ่งเน้นความสัมพันธ์ระหว่างงานหรือขั้นตอนต่าง ๆ ในระบบว่ามีกิจกรรมอะไรบ้าง และความเกี่ยวข้องกันอย่างไร แต่จะไม่แสดงรายละเอียดว่างานนั้นทำอย่างไร



ภาพที่ 8.4 ผังงานระบบลงทะเบียนเรียน

ผังงานโปรแกรม (Program Flowchart)

ผังงานโปรแกรม หรือผังงานแสดงรายละเอียด (Detail Flowchart) เป็นผังงานที่แสดงถึงขั้นตอนของกิจกรรมต่าง ๆ โดยละเอียด ทำให้สามารถถ่ายทอดความเข้าใจหรือสื่อสารระหว่างกันได้ค่อนข้างดีและชัดเจน กล่าวคือ จะมีการแสดงรายละเอียดของกิจกรรม การทำงานในทุกขั้นตอนของโปรแกรมว่าจะต้องทำอย่างไร อีกทั้งผังงานโปรแกรมอาจนำเสนอได้หลายหน้าเนื่องจากกระดาษไม่เพียงพอ ดังนั้นจึงจำเป็นต้องมีการเชื่อมโยงผังงานต่อไปอีกหน้าหนึ่ง โดยรายละเอียดการเขียนผังงานจะกล่าวในหัวข้อเทคนิคการเขียนผังงานต่อไป



ภาพที่ 8.5 ผังงานโปรแกรมลงทะเบียนเรียน

การเขียนรหัสจำลอง (Pseudo Code)

รหัสจำลองหรือซูโดโค้ด เป็นรหัสคำสั่งที่ไม่ใช่คำสั่งภาษาโปรแกรมคอมพิวเตอร์ แต่เป็นชุดรหัสคำสั่งที่เขียนเพื่อเลียนแบบคำสั่งโปรแกรมอย่างย่อ ๆ เพื่อใช้เป็นแนวทางในการสร้างคำสั่งควบคุมของโปรแกรมภาษานั้นต่อไป แต่อย่างไรก็ตาม รหัสจำลองนั้นจะมีความใกล้เคียงกับภาษาระดับสูงมาก จึงทำให้การเขียนรหัสจำลองนั้นเป็นอัลกอริทึมที่ค่อนข้างเป็นที่นิยม เนื่องจากโปรแกรมเมอร์สามารถนำอัลกอริทึมที่เขียนขึ้นจากรหัสจำลองมาพัฒนาเป็นโปรแกรมต่อไปค่อนข้างง่ายนั่นเอง สำหรับการออกแบบอัลกอริทึมในรูปของรหัสจำลองนั้น ถ้อยคำหรือสเตตเมนต์ (Statement) ของรหัสจำลอง คือ การประกาศชื่อของอัลกอริทึม และสเตตเมนต์สุดท้ายจะแสดงถึงจุดจบหรือจุดสิ้นสุดของอัลกอริทึมนั้น แสดงดังภาพที่ 8.6

ALGORITHM PROBLEM 1

Input Test Score 1
Input Test Score 2
Input Test Score 3
Add Test Score 1 + Test Score 2 + Test Score 3
Divide total sum by 3
Print result of division

END PROGRAM 1

ALGORITHM PROBLEM 2

VARIABLES : Total, Counter, Sum, Test Score, Average

BEGIN

Input Total
Count ← 0
Sum ← 0
LOOP while Counter is less that Total
BEGIN LOOP
Input Test Score
Sum ← Sum + Test Score
Counter ← Counter + 1
END LOOP
AVERAGE ← Sum / Total
Print Average

END PROGRAM 2

ภาพที่ 8.6 รูปแบบการเขียนรหัสจำลองหรือซูโดโค้ด

เครื่องหมายหรือคำสั่งที่นิยมใช้ในรหัสจำลองมักประกอบด้วยคำสั่งต่อไปนี้ เช่น BEGIN…END, REPEAT, DO…UNTIL, WHILE…DO, IF…THEN…ELSE, FOR, LOOP ซึ่งแสดงได้จากรายละเอียดต่อไปนี้

Name ← expression

Name ในที่นี้คือชื่อตัวแปรที่ใช้สำหรับเก็บค่าของ expression โดยพิจารณาจากสเตตเมนต์ต่อไปนี้

RemainingFunds ← CheckingBalance + SavingBalance

เป็นการนำผลลัพธ์ที่ได้จากผลบวกของ CheckingBalance และ SavingBalance มาเก็บไว้ใน RemainingFunds

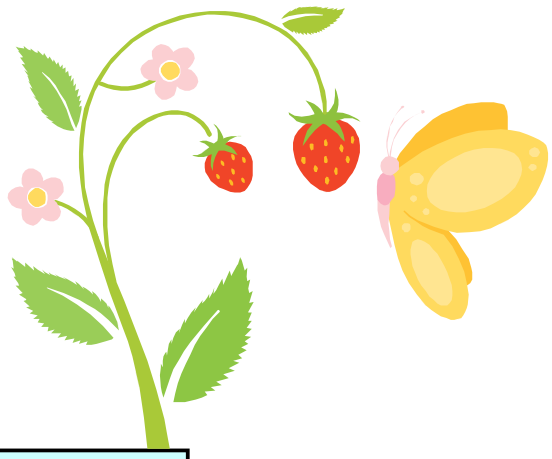
if (condition) **then** (activity)
else (activity)

เป็นการกำหนดเงื่อนไข โดยเงื่อนไขอาจจะมีหลายทางเลือกก็ได้ เช่น

if (sales have decreased) **then** (lower the price by 5%)

if (year is leap year)
then (diary total ← total devided by 366)
else (diary total ← total divided by 365)

if (item is taxable)
then (if (price > limit)
then (pay x)
then (pay y)
)
else (pay z)



while (condition) **do** (activity)

เป็นการตรวจสอบเงื่อนก่อนที่จะทำการดำเนินการ เช่น

while (tickets remain to be sold)
do (sell a tickets)

do (activity) **until** (condition)

เป็นการวนซ้ำ (LOOP) แบบมีเงื่อนไข เช่น ทำการบวกเลขจำนวนเต็ม 1 to 100

num ← 1

sum ← 0

do

sum ← sum + num

num ← num + 1

until (num > 100)

อย่างไรก็ตามในบ้างครั้งการออกแบบรหัสจำลองอาจจำเป็นต้องมีการเรียกใช้แอปพลิเคชันตัวอื่นเพื่อใช้งานได้ ซึ่งเรียกว่าโพรซีเยอร์ (Procedure)

Procedure name

โพรซีเยอร์เปรียบเสมือนกับโปรแกรมย่อยที่สามารถเพิ่มแทรกเข้าไปในลำดับขั้นตอนของอัลกอริทึม โดยการสร้างโพรซีเยอร์เหมาะสมกับในกรณีที่มีหลาย ๆ อัลกอริทึมที่ต้องการใช้งานโพรซีเยอร์เหล่านั้น หรือมีการเรียกใช้งานมากกว่า 1 ครั้ง และรวมถึงเหมาะกับการออกแบบอัลกอริทึมที่มีขนาดใหญ่โดยในขั้นตอนของอัลกอริทึมเมื่อมีการเรียกใช้งานโพรซีเยอร์เป็นที่แล้วเสร็จ ก็จะกลับมาดำเนินการต่อจากขั้นตอนลำดับต่อไปของอัลกอริทึมหลักที่เรียก

if (...) **then** (Execute the procedure ProcessLoan)

else (Execute the procedure RejectApplication)

procedure PrrcessLoan

(activity)

.

.

procedure RejectApplication

(activity)

.

.

```
ALGORITHM  CALCULATE GRADE
OPEN score_file
INITIAL  variables
        DO
                READ  score_file  recorda
                        If  score between  80  to  100
                                Grade  =  'A'
                        If  score between  75  to  79
                                Grade  =  'B+'
                        If  score between  70  to  74
                                Grade  =  'B'
                        If  score between  65  to  69
                                Grade  =  'C+'
                        If  score between  60  to  64
                                Grade  =  'C'
                        If  score between  55  to  59
                                Grade  =  'D+'
                        If  score between  50  to  54
                                Grade  =  'D'
                        else
                                Grade  =  'F'
                Print  records
        UNITIAL  end of file
CLOSE score_file
END
```

ภาพที่ 8.7 อัลกอริทึมการคำนวณเกรดนักศึกษาในรูปแบบของรหัสจำลอง

```
Procedure  Sort  (List)
N ← 2;
While  (the value of N does not exceed the length of List)  do
        (Select the N^th entry in list as the pivot entry;
         Move a pivot entry to a temporary location leaving a hole in List;
         While  (there is a name above the hole and that name is greater than the pivot)  do
                (move the name above the hole down into the hole leaving a hole above the name)
        Move the pivot entry into the hole in List;
        N ← N + 1
        )
```

ภาพที่ 8.8 อัลกอริทึมการจัดเรียงลำดับตัวเลขจากน้อยไปมากในรูปแบบของรหัสจำลอง

จากภาพที่ 4.8 เป็นตัวอย่างอัลกอริทึมในรูปแบบของรหัสจำลอง ซึ่งเป็นขั้นตอนการจัดเรียงลำดับค่าตัวเลขจากน้อยไปหามาก จะเห็นได้ว่ามีการเขียนอยู่ในรูปแบบของโพรซีเยอร์ ซึ่งทำให้อัลกอริทึมอื่น ๆ ที่ต้องการจัดเรียงตัวเลขก็สามารถเรียกใช้โพรซีเยอร์นี้ได้ตามต้องการ

รหัสจำลองหรือซูโดโค้ดนั้น เป็นรหัสที่ใช้แทนคำสั่งเพื่ออธิบายขั้นตอนการทำงานของโปรแกรมที่มีลักษณะเป็นลำดับแบบบนลงล่าง (Top-Down) เช่นเดียวกับผังงาน แต่รหัสจำลองนั้นจะไม่มีการใช้แผนภาพเพื่อนำเสนอเช่นเดียวกับผังงาน แต่จะใช้ถ้อยคำ (Statement) หรือประโยคภาษาอังกฤษอธิบายแทน ซึ่งประโยคภาษาอังกฤษจะประกอบด้วยคำสั่งต่าง ๆ ที่ไม่ขึ้นกับรูปแบบภาคอมพิวเตอร์ภาษาใดภาษาหนึ่งโดยเฉพาะ แต่อย่างไรก็ตามรหัสจำลองก็ยังสามารถใช้รูปแบบภาษาพูดซึ่งอาจจะเป็นภาษาอังกฤษ หรือภาษาไทยก็ได้ แต่ก็ยังคงไว้ซึ่งขั้นตอนการทำงานหลัก ๆ ของโปรแกรมเอาไว้ โดยไม่ต้องเจาะรายละเอียดการทำงานในแต่ละส่วน

คุณสมบัติของอัลกอริทึม

ในการออกแบบอัลกอริทึม จำเป็นต้องมีคุณสมบัติสำคัญ ดังต่อไปนี้

1. เป็นกระบวนวิธีการที่สร้างขึ้นจากกฎเกณฑ์

เนื่องจากอัลกอริทึมจัดเป็นรูปแบบหนึ่งของการแก้ปัญหา และกระบวนการวิธีการก็คือกลุ่มของขั้นตอนที่อยู่รวมกันเพื่อใช้แก้ปัญหาต่าง ๆ เพื่อให้ได้มาซึ่งผลลัพธ์ ดังนั้นจึงจำเป็นต้องมีกฎเกณฑ์ที่ใช้ในการสร้างกระบวนวิธีการเหล่านั้น ซึ่งอาจอยู่ในรูปแบบประโยคภาษาอังกฤษ สัญลักษณ์ หรือคำสั่งจำลอง

2. กฎเกณฑ์ที่สร้างอัลกอริทึมต้องไม่คลุมเครือ

รูปแบบของกฎเกณฑ์ใดก็ตามที่ใช้อัลกอริทึมจะต้องมีระบบ ระเบียบ อ่านแล้วไม่สับสน กล่าวคือจะต้องเป็นกฎเกณฑ์ที่เข้าใจตรงกัน และควรหลีกเลี่ยงคำที่ก่อให้เกิดความเข้าใจได้หลายความหมาย

3. การประมวลผลต้องเป็นลำดับขั้นตอน

คำสั่งต่าง ๆ ที่ถูกกำหนดด้วยกฎเกณฑ์จะต้องประมวลผลเป็นลำดับขั้นตอนที่แน่นอน

4. กระบวนวิธีการต้องให้ผลลัพธ์ตามที่กำหนดในปัญหา

กล่าวคือ กลุ่มของขั้นตอนต่าง ๆ ที่กำหนดไว้จะต้องใช้งานทั่วไปได้สำหรับทุก ๆ กรณี และจะต้องมีผลลัพธ์ตรงตามที่กำหนดในปัญหานั้น ๆ

5. อัลกอริทึมต้องมีจุดสิ้นสุด

คุณสมบัติอีกข้อหนึ่งที่สำคัญคือ อัลกอริทึมต้องมีจุดสิ้นสุด เนื่องจากคอมพิวเตอร์จะไม่สามารถประมวลผลแบบไม่สิ้นสุดได้ (Infinity) เช่น การบวกเลขจำนวนเต็มบวกที่ละตัว ในที่นี้จะไม่ถือเป็นอัลกอริทึม เนื่องจากไม่ได้บอกขอบเขตสิ้นสุดของตัวเลขจำนวนเต็ม ดังนั้นจึงเป็นขั้นตอนการทำงานที่ไม่มีจุดสิ้นสุด

เกณฑ์การพิจารณาประสิทธิภาพอัลกอริทึม (Algorithm Efficiency)

ในการออกแบบอัลกอริทึมเพื่อแก้ปัญหา แต่ละคนอาจออกแบบขั้นตอนการแก้ปัญหาที่แตกต่างกันได้ ซึ่งลองเปรียบเทียบขั้นตอนการล้างรถของแต่ละคนก็ย่อมมีความแตกต่างกัน ทั้งนี้ขึ้นอยู่กับเงื่อนไขและข้อจำกัดต่าง ๆ เพื่อให้ได้มาซึ่งอัลกอริทึมที่เหมาะสมที่สุด แต่อย่างไรก็ตามก็จะมีเกณฑ์ในการพิจารณาถึงประสิทธิภาพของอัลกอริทึม ซึ่งประกอบด้วย

1. **อัลกอริทึมต้องใช้เวลาการดำเนินการน้อยที่สุด**

   อัลกอริทึมที่ดีควรมีขั้นตอนต่าง ๆ เท่าที่จำเป็น และหากเป็นไปได้ก็ควรหลีกเลี่ยงการดำเนินงานที่ข้องเกี่ยวกับอุปกรณ์ ซึ่งมักใช้เวลาในการดำเนินงานนาน

2. **อัลกอริทึมต้องใช้หน่วยความจำน้อยที่สุด**

   ภายใจหน่วยวามจำควรจะมีข้อมูลที่จำเป็นต่อการดำเนินงานในขณะนั้นเท่านั้น ดังนั้นควรหลีกเลี่ยงตัวแปรหรือข้อมูลที่ไม่เกี่ยวข้อง

3. **อัลกอริทึมต้องมีความยืดหยุ่น**

   อัลกอริทึมควรออกแบบให้สามารถปรับปรุงการใช้งานได้ง่าย

4. **อัลกอริทึมต้องใช้เวลาในการพัฒนาน้อยที่สุด**

   ควรใช้เวลาในการพัฒนาที่เหมาะสมกับอัลกอริทึมที่ใช้งานนั้น ๆ เช่น ในกรณีที่อัลกอริทึมนั้นไม่มีความซับซ้อนมากมายนัก แต่ใช้เวลาในการพัฒนามากเกินความจำเป็น ก็แลดูไม่เหมาะสม

5. **อัลกอริทึมต้องง่ายต่อความเข้าใจ**

   รูปแบบและคำอธิบายขั้นตอนวิธีในอัลกอริทึมจะต้องออกแบบและใช้รหัสคำสั่งที่เป็นมาตรฐาน โดยผู้คนทั่วไปสามารถอ่านแล้วเข้าใจความหมายตรงกัน