บทที่ 5
คำสั่งเทียมและภาษาออกแบบโปรแกรม

วิชา ขั้นตอนวิธีและการเขียนโปรแกรมคอมพิวเตอร์ขั้นพื้นฐาน

รหัสวิชา 141141005

สอนโดย.... อาจารย์แพรวศรี เดิมราช

สาขาวิชาเทคโนโลยีสารสนเทศ

คณะวิทยาศาสตร์เทคโนโลยีและการเกษตร

มหาวิทยาลัยราชภัฏยะลา

เนื้อหา

1 การเขียนคำสั่งเทียม

2 คำสั่งโครงสร้างของคำสั่งเทียม

2.1 โครงสร้างแบบตามลำดับ (Sequential)

2.2 โครงสร้างแบบมีการเลือก (Decision)

2.2.1 แบบ If-Thenและ แบบ If-Then-Else

2.2.3 แบบ Case

2.3 โครงสร้างแบบการทำซ้ำ (Repetition)

2.3.1 While…Do 2.3.2 Do…Until

3 ภาษาออกแบบโปรแกรม

บทนำ

เครื่องมือที่ใช้ในการออกแบบโปรแกรมอีกแบบหนึ่ง คือ

- คำสั่งเทียม (Pseudo Code) โดยจะมีลักษณะเป็นการใช้ภาษาอังกฤษธรรมดา คล้ายๆ กับการเขียนโปรแกรมภาษาพีแอล/วัน ภาษาปาสคาล แต่คำสั่งเทียมมีอิสระและคล่องตัวมากกว่า เพราะคำสั่งเทียมไม่ขึ้นอยู่กับไวยากรณ์ของภาษาคอมพิวเตอร์ภาษาใดภาษาหนึ่ง
- ภาษาออกแบบโปรแกรม (Program Design Language) หรือ PLD เป็นการพัฒนาคำสั่งเทียมให้มีฟอร์ม ขึ้นมาเป็นมาตรฐานขึ้น โดยกำหนดกฎเกณฑ์แบบอย่างขึ้นมา

การเขียนคำสั่งเทียม

การเขียนคำสั่งเทียมหมายถึง การนำคำในภาษาอังกฤษ มาแสดงถึงการทำงานของคอมพิวเตอร์ ซึ่งเรียบเรียงเป็นประโยคโดยให้สามารถสื่อความหมายให้ผู้อ่านเข้าใจได้ว่าแต่ละขั้นตอนของการแก้ปัญหานั้นจะทำได้อย่างไร

คำสั่งเทียมที่ดีนั้นต้องมีความชัดเจน สั้น กระชับได้ใจความ ข้อมูลต่างๆ ที่ใช้ ควรอยู่ในรูปแบบของตัวแปร

การเขียนคำสั่งเทียม (ต่อ)

1.1 การรับข้อมูลและการแสดงผล

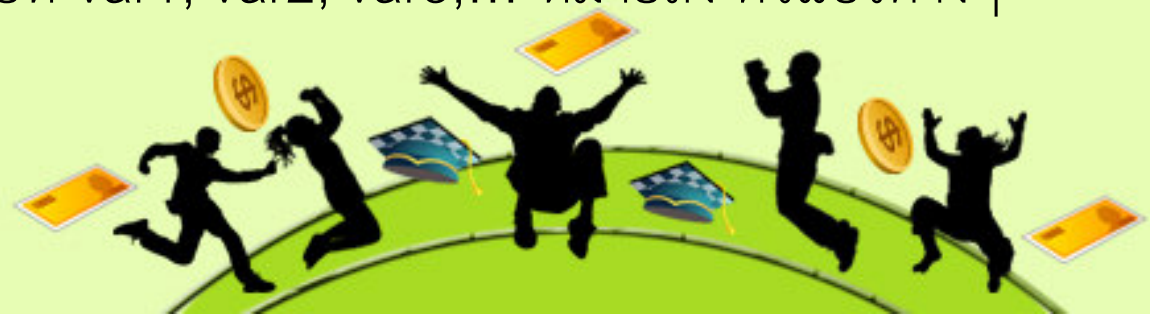
การรับข้อมูลนั้นนิยมใช้คำว่า Read หรือ Input และ การแสดงผลให้ใช้คำว่า Write โดยจะต้องตามด้วยชื่อตัวแปรที่ต้องการอ่านค่า ถ้าต้องการอ่านหรือแสดงค่าตัวแปรหลายตัวก็ใช้เครื่องหมาย คอมม่า (,) คั่นระหว่างชื่อตัวแปรแต่ละตัว

รูปแบบ

Read var1, var2, var3,…

Write var1, var2, var3,…

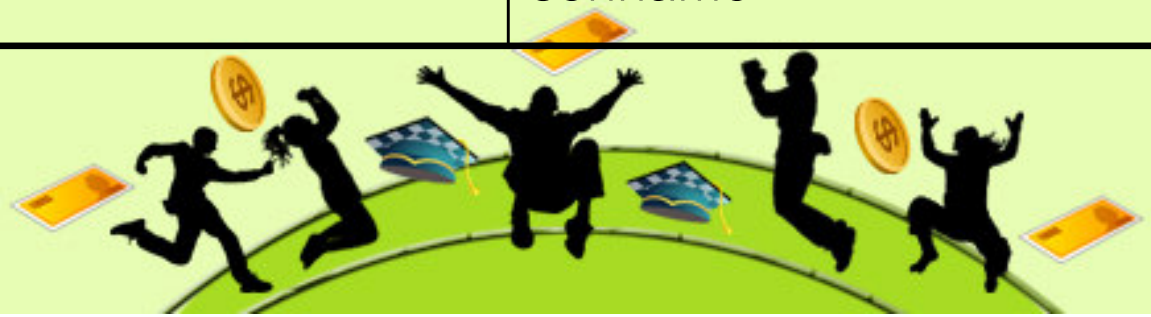
โดยที่ var1, var2, var3,… หมายถึง ตัวแปรต่างๆ



การเขียนคำสั่งเทียม (ต่อ)

ตัวอย่างที่ 5.1 การรับข้อมูลและการแสดงผล

ประโยคคำสั่ง	ความหมาย
Read id, name	อ่านข้อมูลสองตัวเก็บไว้ที่ตัวแปร id และ name ตามลำดับ
Read num1, num2, num3	อ่านค่าข้อมูลเก็บไว้ที่ตัวแปร num1, num2 และ num3 ตามลำดับ
Write “School”,schname	พิมพ์คำว่า “School” และค่าของตัวแปร schname



การเขียนคำสั่งเทียม (ต่อ)

1.2 การกำหนดค่าให้กับตัวแปร

การกำหนดค่าให้กับตัวแปร ใช้สัญลักษณ์ ← ตัวอย่างเช่น

name ← ‘Thanakorn’

Age ← 25

Grade ← ‘A’

การเขียนคำสั่งเทียม (ต่อ)

1.3 การคำนวณ

การคำนวณจะใช้ชื่อตัวแปรที่ต้องการกำหนดค่า หรือ เปลี่ยนค่าไว้ด้านซ้ายแล้วตามด้วยเครื่องหมายเท่ากับ และนิพจน์ หรือค่าคงที่ ที่ต้องการ

รูปแบบ

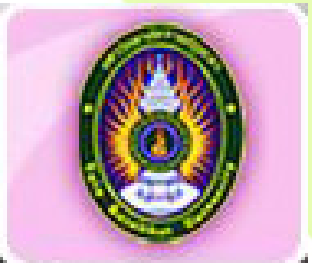
compute ตัวแปร ←นิพจน์

โดยที่นิพจน์ได้ แก่ ค่าคงที่ ตัวแปร ฟังก์ชัน และผลจากการดำเนินกรรมวิธีอย่างใดอย่างหนึ่ง

การเขียนคำสั่งเทียม (ต่อ)

ตัวอย่างที่ 5.2 การกำหนดค่าให้กับตัวแปร

ประโยคคำสั่ง	ความหมาย
Compute count ← count= count + 1	การนำค่า 1 รวมกับค่าที่มีอยู่เดิมแล้วเก็บไว้ที่ตัวแปร
Compute Total ← pr1 +pr2	คำนวณหาผลรวม pr1 บวก pr2 เก็บผลลัพธ์ไว้ที่ Total



การเขียนคำสั่งเทียม (ต่อ)

1.4 รูปแบบของคำสั่งเทียม

รูปแบบทั่วไปของคำสั่งเทียมแบ่งออกเป็นสองส่วน ดังนี้

1. ส่วนหัว (Header) : ชื่อของขั้นตอนวิธี (Algorithm Name)

2. ส่วนคำสั่ง(Algorithm Body) : ลำดับขั้นตอนวิธี

การเขียนคำสั่งเทียม (ต่อ)

ตัวอย่างที่ 5.6 เขียนคำสั่งเทียมคำนวณหาพื้นที่ของสี่เหลี่ยมผืนผ้าใดๆ

Rectangle_Area

Read Wind, Long

Compute Area ← Area= Wind*Long

Write Area

คำอธิบาย

1. ชื่อของคำสั่งเทียมคือ Rectangle_Area

2. ส่วนของคำสั่งมี 3 ประโยคคำสั่ง ดังนี้

2.1 อ่านข้อมูลความกว้าง และความยาว เก็บไว้ในตัวแปร Wide, Long ตามลำดับ

2.2 คำนวณหาพื้นที่สี่เหลี่ยมผืนผ้า โดย Area เท่ากับ Wide*Long

2.3 พิมพ์ค่าของตัวแปร Area

การเขียนคำสั่งเทียม (ต่อ)

ตัวอย่างที่ 5.7 เขียนคำสั่งเทียมคำนวณหาพื้นที่สามเหลี่ยมใดๆ

Triangle_Area

Input base, high

Compute area = 0.5 * base * high

Print area

End

คำอธิบาย

1 ชื่อของคำสั่งเทียม คือ Triangle_Area

2 ส่วนคำสั่งมี 3 ประโยคคำสั่งดังนี้

2.1 อ่านข้อมูล รับค่าฐาน ค่าความสูง ไว้ในตัวแปร **base, high** ตามลำดับ

2.2 คำนวณ พท. สามเหลี่ยม = 1/2 * ฐาน * สูง มาเก็บในตัวแปร area

2.3 แสดงผลพื้นที่สามเหลี่ยม

การเขียนคำสั่งเทียม (ต่อ)

1.5 หลักการเขียนคำสั่งเทียม

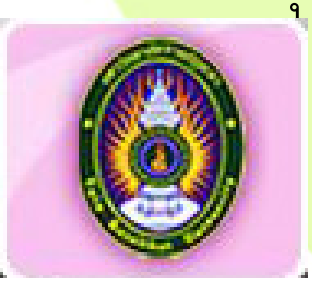
1. ประโยคคำสั่ง เขียนเป็นภาษาอังกฤษอย่างง่าย

2. ประโยคคำสั่งหนึ่ง ๆ จะเขียนต่อหนึ่งบรรทัดเท่านั้น

3. คำหลัก (key word) และการเขียนย่อหน้าใช้เพื่อแยกโครงสร้างควบคุม

4. คำสั่งถูกเขียนจากบนลงล่างโดยมีทางเข้า-ออก เพียงทางเดียว

5. กลุ่มของประโยคคำสั่งอาจถูกจัดอยู่ในรูปส่วนจำเพาะ(Module) และแต่ละกลุ่มต้องมีชื่อเรียก



การเขียนคำสั่งเทียม (ต่อ)

1.6 หลักการเขียนคำสั่งเทียม

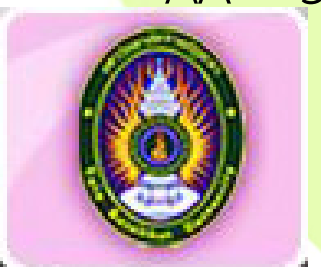
การเขียนรหัสเทียมจะมีคำที่ใช้ในการปฏิบัติการคอมพิวเตอร์ประกอบเป็นส่วนใหญ่ ซึ่งช่วยให้การเปลี่ยนรหัสเทียมเป็นภาษาคอมพิวเตอร์ ทำได้ง่ายขึ้นการปฏิบัติการของคอมพิวเตอร์แบ่งออกเป็น 6 แบบดังนี้

1. การกำหนดค่าให้กับตัวเก็บข้อมูล

1.1 กำหนดค่าเริ่มต้น คำที่ใช้ Initialize หรือ Set เช่น Set AA = 500

1.2 กำหนดค่าที่เกิดจากการประมวลผลไว้ที่ตัวเก็บจะใช้เครื่องหมาย = เช่น

AA = 500 + 1 หรือ BB = 100 หรือ CC = AA



การเขียนคำสั่งเทียม (ต่อ)

1.7 หลักการเขียนคำสั่งเทียม

2. การรับข้อมูล คำที่ใช้ Read หรือ Get เช่น Read AA

3. การแสดงข้อมูลออก คำที่ใช้ Print ,Write , Put , Display , Output เช่น Print “Hello Owen” หรือ Print AA

4. การปฏิบัติการทางคณิตศาสตร์ + , - , * , ( ) เช่น C = (F-32) * 5/9

5. การเปรียบเทียบและทำการเลือก คำที่ใช้ IF, THEN, ELSE

6. คอมพิวเตอร์สามารถปฏิบัติการซ้ำ คำที่ใช้คือDO WHILE และ END DO

คำสั่งโครงสร้างของคำสั่งเทียม

2 คำสั่งโครงสร้างของคำสั่งเทียม

2.1 โครงสร้างคำสั่งเทียมแบบลำดับ

จะทำงานเป็นลำดับก่อนหลังโดยไม่มีการข้ามคำสั่งใดๆ

อธิบายขั้นตอนการทำงาน	คำสั่งเทียม
1. เริ่มต้น	Begin
2. รับค่า base และ height	Read base, height
3. Area = 0.5 * base * height	Compute Area Area = 0.5 * base * height
4. แสดงค่า area	Write area
5. จบกาทำงาน	End

คำสั่งโครงสร้างของคำสั่งเทียม(ต่อ)

2.2 โครงสร้างคำสั่งเทียมแบบเลือกทำ มี 2 รูปแบบ ได้แก่

2.2.1 คำสั่งตรวจสอบเงื่อนไข เรียกว่า คำสั่ง IF มี 3 ลักษณะ

- NULL ELSE (IF-Then)

รูปแบบคำสั่ง IF

จริง

IF <เงื่อนไข> THEN

กลุ่มคำสั่ง

ENDIF

ตัวอย่าง

```
IF a = 1 THEN
    WRITE 'a = 1'
ENDIF
```

คำสั่งโครงสร้างของคำสั่งเทียม(ต่อ)

- แบบ IF – Then-Else

รูปแบบคำสั่ง **จริง**

```
IF <เงื่อนไข> THEN
    กลุ่มคำสั่งที่ 1
ELSE
    กลุ่มคำสั่งที่ 2
ENDIF
```

ตัวอย่าง

```
IF Sex = 'M' THEN
    WRITE "Male"
ELSE
    WRITE "Female"
ENDIF
```

คำสั่งโครงสร้างของคำสั่งเทียม(ต่อ)

- Nested IF (IF-Then-Else-IF)

รูปแบบคำสั่ง **จริง**

```
IF <เงื่อนไขที่ 1 > THEN
    กลุ่มคำสั่งที่ 1
ELSE
    IF <เงื่อนไขที่ 2 > THEN
        กลุ่มคำสั่งที่ 2
    ELSE
        .....
    ENDIF
ENDIF
```

คำสั่งโครงสร้างของคำสั่งเทียม(ต่อ)

ตัวอย่าง

```
IF  Score > 80  THEN
        Grade = 'A'
ELSE
        IF  Score > 70  THEN
                Grade = 'B'
        ELSE
                IF  Score > 60  THEN
                        Grade = 'C'
                ELSE
                        IF  Score > 50  THEN
                                Grade = 'D'
                        ELSE
                                Grade = 'E'
                        ENDIF
                ENDIF
        ENDIF
ENDIF
```

ตัวอย่าง

```
IF  Score > 80  THEN
        Grade = 'A'
ELSE
        IF  Score > 70  THEN
                Grade = 'B'
        ELSE
                IF  Score > 60  THEN
                        Grade = 'C'
                ELSE
                        IF  Score > 50  THEN
                                Grade = 'D'
                        ELSE
                                Grade = 'E'
                        ENDIF
                ENDIF
        ENDIF
ENDIF
```

4 3 2 1

คำสั่งโครงสร้างของคำสั่งเทียม(ต่อ)

2.2.2 แบบ Case

รูปแบบคำสั่งทางเลือกแบบ CASE

```
CASE <Value> OF
        Value 1  :  คำสั่งที่ 1
        Value 2  :  คำสั่งที่ 2
                 :
        Value n  :  คำสั่งที่ n
   Other Value   :  คำสั่ง Other
ENDCASE
```

ตัวอย่าง การเขียนรหัสเทียมด้วยคำสั่ง CASE

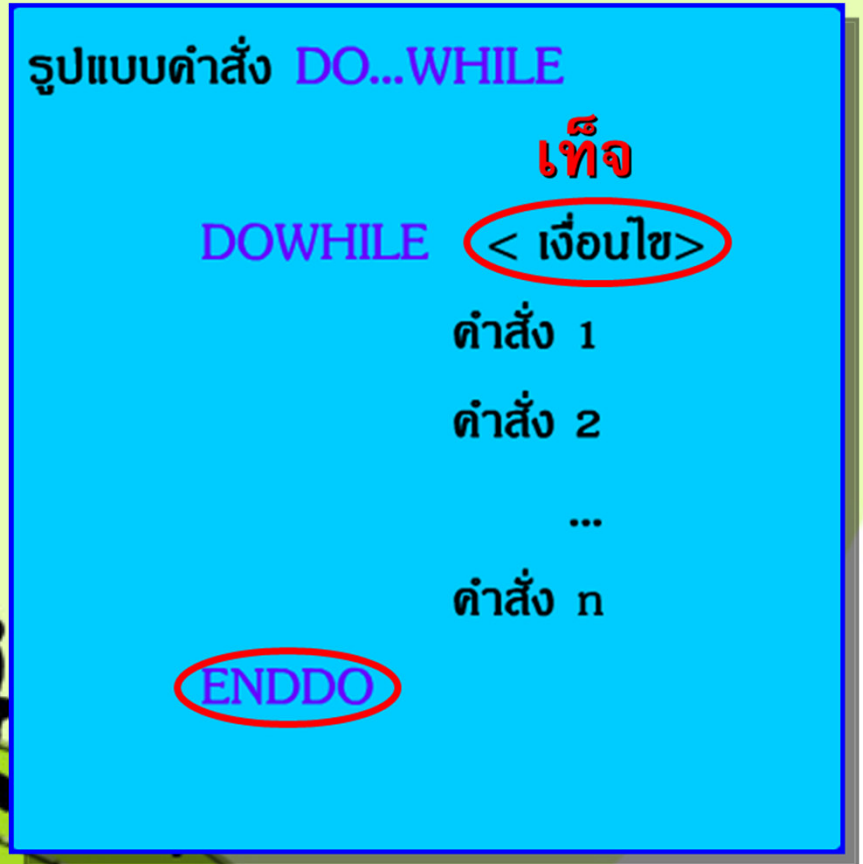
```
CASE  Score OF
        80..100     :   Grade = 'A'
        70..79      :   Grade = 'B'
        60..69      :   Grade = 'C'
        50..59      :   Grade = 'D'
        0..49       :   Grade = 'E'
OTHER
        WRITE 'Score is Invalid !!!'
ENDCASE
```

คำสั่งโครงสร้างของคำสั่งเทียม(ต่อ)

2.3 คำสั่งโครงสร้างของคำสั่งเทียมแบบทำซ้ำ

โครงสร้างคำสั่งเทียมแบบทำซ้ำมี 3 รูปแบบ ได้แก่

2.3.1 Do...WHILE



คำสั่งโครงสร้างของคำสั่งเทียม(ต่อ)

ตัวอย่างคำสั่งการทำซ้ำหรือการวนรอบ

DO...WHILE

ตัวอย่างคำสั่ง DO...WHILE

```
BEGIN
        a = 0, Sum = 0
    DOWHILE a<11
        Sum = Sum + a
        a = a + 1
    ENDDO
        WRITE Sum
END
```

จริง

ตัวอย่างคำสั่ง DO...WHILE

```
BEGIN
        a = 0, Sum = 0
    DOWHILE a<11
        Sum = Sum + a
        a = a + 1
    ENDDO
        WRITE Sum
END
```

เท็จ

คำสั่งโครงสร้างของคำสั่งเทียม(ต่อ)

2.3.2 REPEAT...UNTIL

คำสั่งการทำซ้ำหรือการวนรอบ

รูปแบบคำสั่ง REPEAT...UNTIL

```
REPEAT
        คำสั่งที่ 1
        คำสั่งที่ 2
        ....
        คำสั่งที่ N
UNTIL   <เงื่อนไข>
```

คำสั่งโครงสร้างของคำสั่งเทียม(ต่อ)

ตัวอย่างคำสั่งการทำซ้ำหรือการวนรอบ REPEAT...UNTIL

ตัวอย่างคำสั่ง REPEAT...UNTIL

```
a = 0, Sum = 0
REPEAT
    Sum = Sum + a
    a = a + 1
UNTIL a > 10   เท็จ
WRITE Sum
```

ตัวอย่างคำสั่ง REPEAT...UNTIL

```
a = 0, Sum = 0
REPEAT
    Sum = Sum + a
    a = a + 1
UNTIL a > 10   จริง
WRITE Sum
```

คำสั่งโครงสร้างของคำสั่งเทียม(ต่อ)

2.3.3 FOR...NEXT

รูปแบบคำสั่ง FOR...NEXT

```
FOR  <กำหนดรอบการทำงาน>  DO
        คำสั่งที่ 1
        คำสั่งที่ 2
              ...
        คำสั่งที่ N
NEXT
```

คำสั่งโครงสร้างของคำสั่งเทียม(ต่อ)

ตัวอย่างคำสั่งการทำซ้ำหรือการวนรอบ FOR...NEXT

ตัวอย่างคำสั่ง FOR...NEXT

```
Sum = 0
FOR i = 1 to 10 DO
    Sum = Sum + i
NEXT
WRITE Sum
```

คำสั่งโครงสร้างของคำสั่งเทียม(ต่อ)

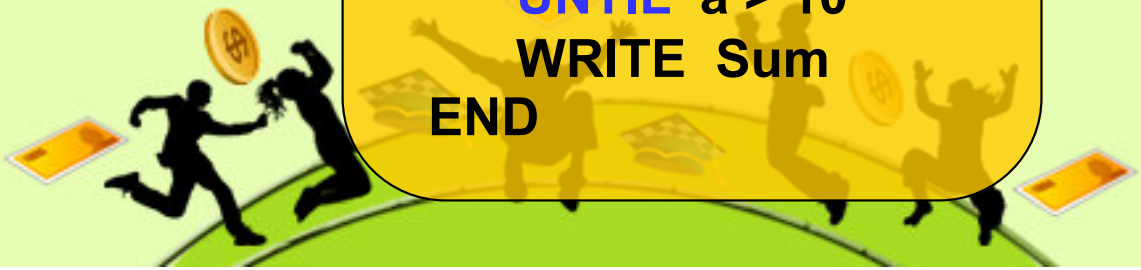
ตัวอย่างคำสั่งการทำซ้ำหรือการวนรอบ 3 แบบ

การหาผลรวมของเลข 1 ถึง 10 บวกสะสมไว้ที่ตัวแปร Sum

```
BEGIN
      a  = 1,  Sum  =  0
      DOWHILE  a<11
            Sum  =  Sum + a
            a = a + 1
      ENDDO
      WRITE  Sum
END
```

```
BEGIN
      a  = 1,  Sum  =  0
      REPEAT
            Sum  =  Sum + a
            a = a + 1
      UNTIL  a > 10
      WRITE  Sum
END
```

```
BEGIN
      Sum  =  0
      FOR  i  = 1  to 10  DO
            Sum  =  Sum + i
      NEXT
      WRITE  Sum
END
```



ภาษาออกแบบโปรแกรม

ภาษาออกแบบโปรแกรมนั้นเป็นการพัฒนาคำสั่งเทียมให้มีฟอร์มมาตรฐานขึ้น โดยกำหนดกฎเกณฑ์บางอย่างขึ้นมา ในหนังสือบางเล่มเรียกภาษาออกแบบโปรแกรมนี้เป็นอีกชื่อหนึ่งของคำสั่งเทียม กล่าวคือไม่มีความแตกต่างกันระหว่างคำสั่งเทียมและภาษาออกแบบโปรแกรม ในการออกแบบโปรแกรมในรายละเอียด ถ้าหากว่าใช้คำสั่งเทียม หรือ PLD นั้น

ภาษาออกแบบโปรแกรม(ต่อ)

ถาวร อนุภาพไตรรงค์(2528,หน้า 18-22) กล่าวว่าควรประกอบด้วยส่วนของคำหลักเพื่อการออกแบบดังต่อไปนี้

1 ส่วนสำหรับกำหนดลักษณะของตัวแปรที่ใช้ในการแปรโปรแกรม ตัวอย่างเช่น

DECLARE customer_record

Customer_no CHAR

Cutomer_name CHAR

Address CHAR

ภาษาออกแบบโปรแกรม(ต่อ)

2.การออกแบบโปรแกรม สามารถรวมกลุ่มของกลุ่มคำสั่ง (Block) โดยใช้คำหลักคำว่า BEGIN และ END ดังตัวอย่าง

BEGIN (bock-name)

:

Pseudo code statements

:

END

ภาษาออกแบบโปรแกรม(ต่อ)

3. คำหลัก สำหรับการเปรียบเทียบโดยใช้ IF-THEN-ELSE ดังตัวอย่าง

IF(condition-description)

THEN (block-or- Pseudocode-statement)

ELSE (block-or- Pseudocode-statement)

EndIF

ภาษาออกแบบโปรแกรม(ต่อ)

ตัวอย่าง

```
IF gross_earning <10000
        THEN BEGIN
                Compute tax = 0.07* gross_earning
                Compute net = gross_earning - tax
        ELSE Skip
EndIF
```

ภาษาออกแบบโปรแกรม(ต่อ)

4.คำสั่งสำหรับการเปรียบเทียบที่มีให้เลือกได้ หลายๆ อย่าง จะใช้การเลือกแบบ Case ดังนี้

CASE OF (Case Variable-name)

WHEN (condition1) SELECT (block or statement)

WHEN (condition2) SELECT (block or statement)

:

ENDCASE

ภาษาออกแบบโปรแกรม(ต่อ)

5.คำสั่งสำหรับการทำงานซ้ำ อาจทำได้ 3 แบบคือ

5.1 DOWHILE (condition)

:

(block-or- Pseudocode-statement)

:

ENDDO

ภาษาออกแบบโปรแกรม(ต่อ)

5.2 DOUNTIL (condition)

:

(block-or- Pseudocode-statement)

:

ENDDO

ภาษาออกแบบโปรแกรม(ต่อ)

5.3 DOFOR I = a TO b By c

:

(block-or- Pseudocode-statement)

:

ENDDO

ในการทำงานซ้ำๆ ดังกล่าวในบางครั้งจำเป็นต้องออกจากการทำซ้ำ เมื่อมีเงื่อนไขบางอย่างเกิดขึ้น อาจทำได้โดยใช้คำหลักว่า EXIT หรือ NEXT

ภาษาออกแบบโปรแกรม(ต่อ)

6 สำหรับโครงสร้างของโปรแกรมย่อย อาจใช้โครงสร้างดังนี้

PROCEDURE (subprogram_name) (attribute)

:

block-or- Pseudocode-statement

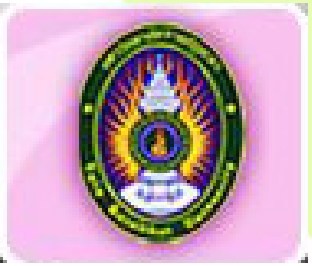
:

RETURE

END

สรุป

คำสั่งเทียมและภาษาออกแบบโปรแกรม หรือ PDL เป็นเครื่องมือที่ใช้สำหรับในการออกแบบโปรแกรมอีกแบบหนึ่ง โดยมีลักษณะเป็นการใช้ภาษาอังกฤษธรรมดา ที่มีอิสระหรือ คล่องตัวกว่าการใช้ภาษาคอมพิวเตอร์ เพราะว่าคำสั่งเทียมนั้นไม่ขึ้นอยู่กับไวยากรณ์ของภาษาใดภาษาหนึ่ง ซึ่งง่ายต่อการเปลี่ยนแปลงแก้ไขตรรกวิทยาของโปรแกรมมากกว่าภาษาคอมพิวเตอร์ ส่วนภาษาออกแบบโปรแกรมนั้นเป็นการพัฒนาคำสั่งเทียมให้มีแบบฟอร์มให้เป็นมาตรฐานขึ้น โดยกำหนดกฎเกณฑ์บางอย่างขึ้น ในปัจจุบันคำสั่งเทียมหรือภาษาออกแบบโปรแกรมได้รับความนิยมแพร่หลาย เพราะภาษาคอมพิวเตอร์สมัยใหม่จะสามารถใช้คำสั่งเทียมหรือภาษาออกแบบโปรแกรมได้ดีและชัดเจนกว่า



จบบทที่ 5