บทที่ 6

Procedure division

statements ต่าง ๆ ใน *division* นี้จะเป็นตัวบรรยาย operation ที่กระทำใน object program, statement เป็นประโยคที่ประกอบด้วย language element เหล่านี้ reserved words, programmer-specified names, literals และ symbols, จาก statements ถ้านำมา combined เข้าด้วยกัน ก็เป็น sentence คือจบด้วยหนึ่ง period เสมอ, ทุก ๆ sentence จะเป็นส่วนหนึ่งของ paragraph แต่ไม่จำเป็นต้องเป็นส่วนหนึ่งของ section หมายความว่า division นี้ จะไม่มี section ก็ได้ โดยมีรูปแบบดังนี้

PROCEDURE DIVISION.
[section-name-l SECTION:.]
paraagraph-name-1

[section-name-2 SECTION.]
paragraph-name-i

[section-name-k SECTION.]
paragraph,-name-n

ชนิดของคำสั่ง (categories of statements)

ภาษาโคบอลแบ่งคำสั่งออกเป็น 3 ชนิด คือ conditional statements, imperative statements และ compiler-directing statements

---

หมายเหตุ หน่วยของข้อมูลจากเล็กที่สุดไปยังหน่วยใหญ่สุดแสดงด้วยรูปภาพดังนี้

character → word → statement → sentence → paragraph → section → division → program

6.1 Imperative statements

imperative statement แต่ละคำสั่งจะประกอบด้วย verb หนึ่งตัว และ literals หรือ data names ซึ่งจะนำมาใช้เป็นตัวถูกกระทำ (operands) ใน operation ที่กำหนดด้วย verb ตัวนั้น, verb แต่ละตัวจะนำมาใช้ใน operation ข้างล่างนี้

คำนวณ ได้แก่

ADD
SUBTRACT
MULTIPLY
DIVIDE
COMPUTE

เคลื่อนย้ายและจัดระเบียบข้อมูล ได้แก่

MOVE
EXAMINE

ควบคุมขั้นตอนคำสั่ง ได้แก่

GO TO
ALTER
PERFORM
STOP

รับและแสดงผลข้อมูล ได้แก่

OPEN
READ
WRITE
CLOSE
ACCEPT
DISPLAY

6.1.1 Arithmetic statements

คำสั่ง ADD

คำสั่งนี้ ให้บวกค่าที่เป็นตัวเลข ตั้งแต่ 2 ตัวขึ้นไป ถึงมากที่สุด 9 ตัวเข้าด้วยกัน แล้วเก็บ ผลบวกไว้ใน item ตัวหนึ่ง โดยมีรูปแบบดังนี้

ADD { data-name-1 / literal-1 } [{ data-name-2 / literal-2 } ....] [**TO** { data-name-m / literal-m }]

[**GIVING** data-name-n]]**ROUNDED**] [ON **SIZE ERROR** any-imperative-statement (s)]

กฎเกณฑ์

1) ถ้ามี TO แต่ไม่มี GIVING หมายความว่า ให้บวก items และ literals ทุกตัว รวมทั้ง data-name-m เข้าด้วยกัน ผลลัพธ์เก็บไว้ที่ data-name-m (ตัวบวกตัวสุดท้ายต้องเป็น data-names item เสมอ เพราะจะเป็นตัวเก็บผลลัพธ์)

ตัวอย่าง

ADD 149 TO .FIELDONE.

ADD **NETPAY**, 50, **BONUSPAY** TO OVERTIME.

ADD A TO B ROUNDED.

ADD **FIELDX** TO **FIELDY** ON SIZE ERROR GO TO ERROR-ROUTINE.

ADD A, B, 17, 200.51, TAX TO D-ITEM.

2) ถ้ามี GIVING แต่ไม่มี TO หมายความว่า ให้บวก items และ literals ทุกตัว ที่อยู่หน้าคำ GIVING เข้าด้วยกัน ผลลัพธ์เก็บที่ data-name-n

ตัวอย่าง

ADD A, B GIVING C.

ADD FIELDM FIELDN, FIELD0 GIVING FIELDP ROUNDED ON SIZE ERROR GO TO SPECIAL-PARAGRAPH.

ADD SCORE-l, SCORE-2, SCORE-3 GIVING SCORE-T.

3) ถ้ามีทั้ง TO และ GIVING หมายความว่า ให้บวก item และ literals รวมทั้ง data-name-m/literal-m ทุกตัว เข้าด้วยกัน ผลลัพธ์เก็บที่ data-name-n

ตัวอย่าง

ADD A. B, C. D TO E GIVING F.

4) ถ้าไม่มี TO และไม่มี GIVING หมายความว่า ให้เอา item และ literals ทุกตัว บวกเข้าด้วยกัน ผลลัพธ์เก็บที่ data-name item ตัวสุดท้าย

ตัวอย่าง

ADD 11, 201, 5, 80, B, SUM.

(หมายเหตุ จะให้ผลลัพธ์เหมือนกับกฎข้อ (1) ตรงที่จะมี TO หรือไม่มี TO ก็ได้)

สำหรับเครื่อง IBM

รูปแบบที่ 1

```
ADD  {data-name-1} [data-name-2 ...]   TO data-name-m [ROUNDED]
     {literal-1  } [literal-2      ]
  [ON SIZE ERROR imperative-statement]
```

รูปแบบที่ 2

```
ADD  { data-name-1 } { data-name-2 } [data-name-3] . ..GIVING data-name-m
     { literal-1   } { literal-2   } [literal-3   ]
  [ROUNDED] [ON SIZE ERROR imperative-statement]
```

รูปแบบที่ 3

```
ADD  { CORR          }  data-name-1 TO data-name-2
     { CORRESPONDING }
  [ROUNDED] [ON SIZE ERROR imperative-statement]
```

ในที่นี้ data-name-1 และ data-name-2 ต้องเป็นข้อมูลกลุ่มทั้งคู่

ตัวอย่าง

```
01  REC-IN.
    03 FILLER PICTURE X(2).
    03 A PICTURE 9(2)V9(2).
```

```
    03 FILLER PICTURE X(5).
    03 B PICTURE 9(6).
01  REC-OUT.
    03 A PICTURE 9(2)V9(2).
    03 B PICTURE 9(6).
```

หลังจากนั้นใช้คำสั่ง

```
ADD CORR REC-IN TO REC-OUT.
```

คำสั่ง SUBTRACT

คำสั่งนี้ให้เอาผลรวมของค่าที่เป็นตัวเลข ตั้งแต่ 1 ตัวขึ้นไป มากที่สุด 9 ตัว ลบออกจาก ค่าที่เป็นตัวเลขอีกตัวหนึ่ง ผลลัพธ์เก็บที่ item ค่าหนึ่ง โดยมีรูปแบบดังนี้

SUBTRACT { data-name-1 / literal-1 } [{ data-name-2 / literal-2 } ....] **FROM** { data-name-m / literal-m }

[GIVING data-name-n] [ROUNDED] [ON SIZE ERROR any imperative statement (s)]

กฎเกณฑ์

1) ถ้ามีคำว่า GIVING ให้เอา literals และ data-name items ทุกตัวหน้าคำว่า FROM บวกเข้าด้วยกัน แล้วเอาไปลบจาก data-name-m หรือ literal-m ผลลัพธ์เก็บไว้ที่ data-name-n

ตัวอย่าง

```
SUBTRACT A, B, C, 100 FROM D GIVING DIFF
SUBTRACT A FROM, 100.00 GIVING ANSWER.
SUBTRACT NETPAY FROM GROSSPAY GIVING TAKEHOMEPAY.
```

2) ถ้าไม่มี GIVING, ให้เอาผลลัพธ์ ไปเก็บไว้ที่ data-name-m

ตัวอย่าง

```
SUBTRACT A FROM B.
SUBTRACT 5, TAX-DEDUCTION FROM TAKE-HOME-PAY ROUNDED.
```

คำสั่ง MULTIPLY

คำสั่งนี้ให้เอาค่าที่เป็นตัวเลข 2 จำนวนคูณเข้าด้วยกัน ผลลัพธ์เก็บใน item อีกตัวหนึ่ง โดยมีรูปแบบดังนี้

MULTIPLY { data-name-1 / literal-1 } **BY** { data-name-2 / literal-2 } [**GIVING** data-name-3]

[**ROUNDED**] [**ON SIZE ERROR** any - imparative-statement (s)]

กฎเกณฑ์

1) ถ้ามี GIVING ให้เอา numeric item ทั้งสองจำนวนนั้นคูณเข้าด้วยกัน ผลลัพธ์เก็บที่ data-name-3

ตัวอย่าง

```
MULTIPLY .025 BY TAX GIVING TAX-2.5.
MULTIPLY A BY B GIVING C.
```

2) ถ้าไม่มี GIVING ให้เก็บผลคูณไว้ที่ data-name-2

ตัวอย่าง

```
MULTIPLY .025 BY TAX.
MULTIPLY HOURS-WORKED BY HOURLY-RATE.
```

คำสั่ง DIVIDE

คำสั่งนี้ ให้เอาค่าที่เป็นตัวเลขจำนวนหนึ่งหารค่าที่เป็นตัวเลขอีกจำนวนหนึ่ง ผลหารเก็บที่ item อีกตัว โดยมีรูปแบบดังนี้

DIVIDE { data-name-1 / literal-1 } **INTO** { data-name-2 / literal-2 } [**GIVING** data-name-3]

[**ROUNDED**] [**REMAINDER** data-name-41 [**ON SIZE ERROR** any-imperative-statement (s)]

กฎเกณฑ์

1) ถ้ามี GIVING

data-name-2 หรือ literal-2 จะเป็นตัวตั้ง (divident)

data-name-l หรือ literal-l จะเป็นตัวหาร (divisor)

data-name-3 เป็นตัวเก็บผลหาร (quotient)

data-name-4 เป็นตัวเก็บเศษ

ตัวอย่าง

DIVIDE A INTO B GIVING C.

2) ถ้าไม่มี GIVING, ผลลัพธ์จากการหารเก็บใน data-name-2

ตัวอย่าง

DIVIDE 12 INTO T-M-TOTAL.

หมายเหตุ ถ้าตัวหารมีค่าเป็นศูนย์ จะเกิด size error ขึ้น

คำสั่ง COMPUTE

คำสั่งนี้กำหนดค่าของ data-name, literal หรือ arithmetic expression ให้กับ data-name อีก 1 ตัว

COMPUTE data-name-l [ROUNDED] = { data-name-2 / literal-l / arithmetic-expression }

[ON SIZE ERROR imperative-statement]

ในที่นี้ data-name-l อาจจะเป็น numerically edited data item

ตัวอย่าง

```
COMPUTE    A = 0.
COMPUTE    X-ITEM = A + B / C.
COMPUTE    I = I + 1.
```

สำหรับ operations ทางคณิตศาสตร์ ในภาษาโคบอลมี 5 ชนิด คือ

+	บวก
–	ลบ
*	คูณ
/	หาร
**	ยกกำลัง

และระหว่าง operator แต่ละตัวจะต้องมีตัวอักขระ blank อย่างน้อยหนึ่งตัวคั่นทั้งหน้าและหลัง

ตัวอย่าง

algebraic expression	COBOL arithmetic expression
a + b	A + B
a – b + (a – 5)c	A – B + (A – 5) * C
a^2 $\frac{b+c}{2}$	A ** 2 – (B t C) / 2

เมื่อมีเครื่องหมายวงเล็บกำกับ operation ภายในเครื่องหมายวงเล็บจะถูก execute เป็นอันดับแรก โดยเครื่องจะทำ expression ในวงเล็บในสุดก่อนเสมอ แล้วเริ่มจากซ้ายไปขวา ใน expression นั้น ในกรณีที่ไม่มีเครื่องหมายวงเล็บอยู่ด้วย เครื่องคอมพิวเตอร์จะทำตามระดับความสำคัญของตัว operator ดังนี้

	เครื่องหมาย **	มีความสำคัญสูงสุด
	เครื่องหมาย * , /	มีความสำคัญรองลงมา
และ	เครื่องหมาย + , –	มีความสำคัญน้อยที่สุด

ตัวอย่าง

COBOL arithmetic expression	algebraic expression
A + B / C	$a + \frac{b}{c}$
(A + B) / C	$\frac{a+b}{c}$
A t (B / C)	$a + \frac{b}{c}$

ROUNDED option

ถ้าในการคำนวณ ปรากฏว่า จำนวนเลขหลังจุดทศนิยมของผลลัพธ์ มากกว่า จำนวนจุดทศนิยมที่กำหนดไว้ใน receiving item ส่วนที่เกินจะถูกตัดทิ้งไป ในระหว่างที่เครื่อง execute object program เรียกว่าเกิด normal truncation occurs

ตัวอย่าง **ADD 38.2519 TO 100.2 GIVING C.**

ถ้ากำหนด picture ของ C เป็น 999V99

```
  3X.2519
 100.2    +
 138.4519
```

ค่าที่ปรากฏใน item C จะเป็น 138.45 ถ้ามี ROUNDED อยู่ในคำสั่งนั้น เมื่อเกิดกรณีข้างต้น เครื่องจะปัดเศษให้ โดยยึดหลักเกณฑ์ดังนี้ ถ้าตำแหน่งจุดทศนิยม ตัวที่ถัดไปจากตำแหน่งจุดทศนิยมที่กำหนดไว้ใน receiving item เป็นเลข 5 หรือมากกว่าเลข 5 ให้ปัดเศษขึ้นมา 1 ถ้าน้อยกว่า 5 ปัดทิ้งไป

จากตัวอย่างเดิม

ADD 38.2519 TO 100.2 GIVING C ROUNDED

ผลลัพธ์ ใน item C ก็ยังคงเป็น 138.45

ตัวอย่าง ADD 5.3025 TO 11.183 GIVING C ROUNDED.

ให้ C มี picture เท่ากับ 99V999

5.3025

11.183 +

16.485 (5)

+ 1

ผลลัพธ์ ใน item C เท่ากับ 16.486

ON SIZE ERROR option

เกิดขึ้นเมื่อการคำนวณ ให้จำนวนเลขหน้าจุดทศนิยมของผลลัพธ์มากกว่าจำนวนเลขหน้าจุดทศนิยมที่กำหนดใน receiving item เรียกว่า เกิด size error เครื่องจะไม่ยอมรับผลจากการคำนวณครั้งนี้ ถ้าในคำสั่งนั้นมีคำว่า ON SIZE ERROR อยู่ด้วย, imperative statement (s) ที่ตามหลังจะถูก execute ถ้าไม่เกิด size error, imperative statement (s) นั้น ก็ไม่ถูก execute

ตัวอย่าง

SUBTRACT HOURS-SICK FROM 40 GIVING HOURS-WORKED ON SIZE ERROR GO TO ERROR-ROUTINE.

ตัวอย่างโปรแกรม

แผนกบริหารงานบุคคลของบริษัทแห่งหนึ่ง บันทึกจำนวนชั่วโมงทำงาน ภายในหนึ่งสัปดาห์ของพนักงานแต่ละคนไว้ในบัตร 80 คอลัมน์ 1 ใบโดยมีรายละเอียดดังนี้

คอลัมน์	รายการ
1 - 5	รหัสพนักงาน
7 - 36	ชื่อ นามสกุล
38 - 40	จำนวนชั่วโมงทำงานในวันจันทร์
41 - 43	จำนวนชั่วโมงทำงานในวันอังคาร
44 - 46	จำนวนชั่วโมงทำงานในวันพุธ
47 - 49	จำนวนชั่วโมงทำงานในวันพฤหัส
50 - 52	จำนวนชั่วโมงทำงานในวันศุกร์
53 - 55	จำนวนชั่วโมงทำงานในวันเสาร์

ให้นักศึกษาเขียน flow chart และโปรแกรมอ่านบัตรข้อมูลดังกล่าวข้างต้น จำนวนพนักงานมีไม่เกิน 100 คน แล้วคำนวณว่าพนักงานแต่ละคนใน 1 สัปดาห์นั้นทำงานทั้งหมดได้กี่ชั่วโมง และหาค่าเฉลี่ยว่าในหนึ่งวันพนักงานคนนั้นทำงานได้กี่ชั่วโมง กี่นาที จากนั้นให้พิมพ์ผลลัพธ์ดังนี้

คอลัมน์	
1 - 5	รหัสพนักงาน
7 - 36	ชื่อ นามสกุล
41 - 45	จำนวนชั่วโมงทำงานทั้งหมดใน 1 สัปดาห์
47 - 52	ค่าเฉลี่ยการทำงานในหนึ่งวัน
	คอลัมน์ 47 - 49 = จำนวนชั่วโมง
	คอลัมน์ 50 - 52 = จำนวนนาที

```
IDENTIFICATION DIVISION.
PROGRAM-ID.  STRADPROG.
ENVIRONMENT DIVISION.
CONFIGURATION SECTION.
SOURCE-COMPUTER.  IBM-370-138.
OBJECT-COMPUTER.  IBM-370-138.
INPUT-OUTPZJT SECTION.
FILE-CONTROL.
    SELECT INPUT-FILE ASSIGN TO UT-S-SYSIN.
    SELECT OUTPUT-FILE ASSIGN TO UT-1403-S-SYSPRINT.
DATA DIVISION.
FILE SECTION,,
FD  INPUT-FILE
    LABEL RECORD IS OMITTED
    DATA RECORD IS INPUT-REC.
01  INPUT-REC.
    05 EMPLOYEE-ID          PIC Y(5).
    05 FILLER               PIC x.
    05  EMPLOYEE-NAME       PIC X(30).
    05 FILLER               PIC x.
    05 HOUR                 PIC Y(3) OCCURS 6 TIMES.
    05 FILLER               PIC X(25).
FD  OUTPUT-FILE
    LABEL RECORD IS OMITTED
    DATA RECORD IS PRINT-OUT.
01  PRNT-OUT.
    05 FILLER               PIC X(132).
WORKING-STORAGE SECTION.
77 EOF            ' PIC X(3)      VALUE 'NO'.
77 HOUR-TOTAL       PIC Y(5)      VALUE ZEROS.
77 HOUR-MEAN        PIC 9(3)V9(2) VALUE ZEROS.
77 HOUR-TIME        PIC Y(3)      VALUE ZEROS,
77 HOUR-POINT       PIC V9(2)     VALUE ZEROS.
77 MIN-TIME:        PIC 9(3)      VALUE ZEROS.
01 HEADING-LINE-l,
    05 FILLER       PIC X(56)     VALUE SPACE.
    05 FILLER       PIC X(76)     VALUE 'EMPLOYEE-HOUR REPORT'.
01 HEADING.-LINE-2.
    OS FILLER       PIC X(28)     VALUE SPACE.
    05 FILLER       PIC X(26)     VALUE 'EMPLOYEE NO.'.
    05 FILLER       PIC X(27)     VALUE 'EMPLOYEE-NAME'
    05 FILLER       PIC X(15)     VALUE 'HOUR-TOTAL'.
    05 FILLER       PIC X(36)     VALUE 'AVERAGE:
.
```

```
01  DETAIL-LINE.
    05 FILLER          PIC X(31)     VALUE SPACE.
    05 EMP-ID          PIC 9(5).
    05 FILLER          PIC X(9)      VALUE SPACE.
    05 EMP-NAME        PIC X(30).
    05 FILLER          PIC X(X)      VALUE SPACE.
    05 H-TOTAL         PIC zzz99.
    05 FILLER          PIC X(8)      VALUE SPACE.
    05 Al              PIC zzz.
    05 FILLER          PIC x         VALUE
    05 A2              PIC 99.
    05 FILLER          PIC X(29)     VALUE SPACE.
PROCEDURE DIVISION.
INITIAL-RTN.
    PERFORM OPEN-FILES.
    PERFORM PROCESS-RTN UNTIL EOF = 'YES'.
    PERFORM END-RTN.
OPEN-FILES.
    OPEN INPUT INPUT-FILE
         OUTPUT OUTPUT-FILE.
    WRITE PRNT-OUT FROM HEADING-LINE-I
         AFTER ADVANCING 10 LINES.
    WRITE PRNT-OUT FROM HEADING-LINE-2
         AFTER ADVANCING 1 LINES.
    READ INPUT-FILE AT END MOVE 'YES' TO EOF.
PROCESS-RTN.
    ADD HOUR (I) HOUR (2) HOUR (3) HOUR (4) HOUR (5) HOUR (6) GIVING
         HOUR-TOTAL.
    DIVIDE HOUR-TOTAL BY 6 GIVING HOUR-MEAN ROUNDED.
    MOVE HOUR-MEAN TO HOUR-TIME HOURPOINT.
    MULTIPLY 60 BY HOUR-POINT GIVING MIN-TIME ROUNDED.
    MOVE EMPLOYEE-ID TO EMP-ID.
    MOVE EMPLOYEE-NAME TO EMP-NAME.
    MOVE HOUR-TOTAL TO H-TOTAL.
    MOVE HOUR-TIME TO Al.
    MOVE MIN-TIME TO A2.
    WRITE PRNT-OUT FROM DETAIL-LINE AFTER ADVANCING 1 LINES.
    READ INPUT-FILE AT END MOVE 'YES' TO EOF.
END-RTN.
    CLOSE INPUT-FILE OUTPUT-FILE,
    STOP RUN.
```

EMPLOYEE HOUR REPORT

EMPLOYEE-NO	EMPLOYEE-NAME	HOUR-TOTAL	AVERAGE
00101	MR. SOMCHAI	46	7.40
00102	MR. PAYUNG	48	8.
00103	MR. SOMCHAI	33	5.30
00104	MR. WANDEE	43	7.10
00105	MRS. MALEE	39	6.30
00106	MR. KARUN	44	7.20
00107	MR. TARDA	42	7.
00108	MR. PAWIT	33	5.30
00110	MISS. YUPIN	32	5.20
00111	MISS. SUCHADA	40	6.40

แบบฝึกหัด

1. กำหนดเนื้อที่ฟิลด์ต่าง ๆ ไว้ดังนี้

W	ใช้ picture	S999V99	มีค่าแรกเท่ากับ	01000
X	ใช้ picture	999V99	มีค่าแรกเท่ากับ	09000
Y	ใช้ picture	999V99	มีค่าแรกเท่ากับ	03000
Z	ใช้ picture	999	มีค่าแรกเท่ากับ	040

จงบอกค่าของ receiving items หลังจากเครื่องคอมพิวเตอร์ execute แต่ละคำสั่งต่อไปนี้

a) ADD 5, Y GIVING W.

b) ADD 100.25 TO Y ROUNDED ON SIZE ERROR MOVE ZERO TO Z.

c) DIVIDE Y INTO X.

d) DIVIDE X INTO Y.

e) DIVIDE 12.2 INTO Y GIVING Z REMAINDER X.

f) DIVIDE Z BY 12.2 GIVING Y ROUNDED REMAINDER X.

2. จงเปลี่ยน COBOL arithmetic expression ต่อไปนี้ให้เป็น Algebraic expression ให้ถูกต้อง

a) ((A + (B * C)) / D) ** 2

b) (A + (B * C)) / D ** 2

c) A + (B * C) / D ** 2

d) A + B * C / D ** 2

3. จาก algebraic expression ข้างล่างนี้ จงเขียนให้อยู่ในรูปของ COBOL arithmetic expression ที่มีความหมายเหมือนกัน

a) $\left(\frac{a + bc}{d}\right)^2$

b) $\frac{a + bc}{d^2}$

c) $a + \frac{bc}{d^2}$

c) $\frac{abc}{d^2}$

4. จงเปลี่ยนสูตรต่อไปนี้ให้เป็นคำสั่งในภาษาโคบอล โดยใช้คำสั่งน้อยที่สุดเท่าที่จำเป็น ถ้ามีผลลัพธ์เกิดขึ้นในระหว่างการคำนวณให้ใช้ชื่อ TEMP

a) NEWINV = OLDINV ▪ SALES ▪ WASTE

b) TOT = OLDBAL + INT + DEP

c) COOKTIME = (15) (POUNDS) ⁻ 20

d) NWBAL = ODBAL + PURCHASES + (ODBAL ÷ 20)

5. จงพิจารณากลุ่มของคำสั่งทั้งหมดข้างล่างนี้ ซึ่งเป็นส่วนหนึ่งของโปรแกรมภาษาโคบอลแต่ละคำสั่งจะได้รับการ execute ตามลำดับ หลังจากที่เครื่อง execute แต่ละคำสั่งแล้ว จงบอกค่าปัจจุบันของ data-name แต่ละตัวในกรณีที่ไม่มีการกำหนดมูลค่า ให้นักศึกษาใส่เครื่องหมาย dash (–) 1 ตัว

	ACCT	DEP	INT
a) MOVE 200 TO ACCT.	-	-	-
h) DIVIDE 20 INTO ACCT GIVING INT.			
c) ADD INT, ACCT GIVING ACCT.	_		
d) MULTIPLY INT BY 5 GIVING DEP.			
e) ADD. DEP. ACCT GIVING ACCT.	_		
f) DIVIDE 20 INTO ACCT GIVING INT.		_ _ -	-
g) ADD ACCT. INT GIVING ACCT.	_		
h) SUBTRACT 73 FROM ACCT GIVING ACCT.	-	-	-

เฉลย

1. a) 03500

   b) 000

   c) 00300

   d) 00033

   e) 002

   f) 00328

2. a) $\left(\frac{a + bc}{d}\right)^2$

   a) $\left(\frac{a + bc}{d}\right)^2$

   b) $\frac{a + bc}{d^2}$

   c) $a + \frac{bc}{d^2}$

   d) $a + \frac{bc}{d^2}$

4. a) SUBTRACT SALES, WASTE FROM OLDINY GIVING NEWINV.

   b) ADD OLDBAL INT DEP GIVING TOT.

   c) MULTIPLY 15 BY POUNDS GIVING TEMP.

   SUBTRACT 20 FROM TEMP GIVING COOKTIME.

   d) DIVIDE 20 INTO ODBAL GIVING TEMP.

   ADD OLDBAL PURCHASES TEMP GIVING NWBAL.

5.

	ACCT	DEP	INT
a)	200	-	-
b)	200	-	10
c)	210	-	10
d)	210	50	10
e)	260	50	10
f)	260	so	13
g)	273	50	13
h)	200	50	13

6.1.2 Data manipulation statements

คำสั่ง MOVE

คำสั่งนี้ ให้ย้ายข้อมูลจากเนื้อที่หนึ่ง (item) ในหน่วยความจำไปยังเนื้อที่อื่น ๆ ในหน่วยความจำของคอมพิวเตอร์ ถ้าข้อมูลมีค่าเป็นตัวเลข และ picture ของ receiving item มี editing symbol อยู่ด้วย ระหว่างที่มีการเคลื่อนย้ายข้อมูล จะเกิดการอีดิท (edit) ขึ้น คำสั่ง move มี 2 รูปแบบ คือ

แบบที่ 1

```
MOVE { data-name-1 } TO data-name-2 [data-name-3]...
     { literal-1   }
```

แบบที่ 2

```
MOVE { CORRESPONDING } data-name-1 TO data-name-2
     { CORR          }
```

ในที่นี้ data-name-1 และ data-name-2 ต้องเป็นข้อมูลกลุ่มทั้งคู่

รูปแบบที่ 1 data-name-1 และ literal-1 หมายถึงเนื้อที่ของข้อมูลที่จะส่ง (sending area หรือ source item) ส่วน data-name-2 และ data-name-3 หมายถึง เนื้อที่รับข้อมูล (receiving area หรือ receiving items)

รูปแบบที่ 2 option CORRESPONDING ใช้ย้ายข้อมูลระหว่างเนื้อที่ต่าง ๆ ที่มีชื่อเหมือนกัน ซึ่งอยู่ในข้อมูลกลุ่ม

data-name ในรูปแบบที่ 2 นี้ จะมีเลขบอกระดับ 66, 77 และ 88 ไม่ได้

ข้อมูลจากแต่ละกลุ่มนั้นจะ CORRESPONDING กันเมื่อมันมีชื่อเหมือนกัน และ qualification เหมือนกัน แต่ไม่นับรวม data-name-1 และ data-name-2

ต้องมี data items อย่างน้อยที่สุด 1 คู่ที่เหมือนกัน และเป็นข้อมูลเดี่ยว

ข้อมูลย่อยแต่ละตัวในข้อมูลกลุ่มถ้ามี OCCURS, REDEFINES, หรือ RENAMES clause เครื่องจะไม่สนใจ

ตัวอย่าง

```
01 INSPECTION.
   03 TOTAL-QUANTITY
   03 REJECTED
   03 ACCEPTED
```

```
    03  QUANTITY-RATIO
01  QUANTITY-REPORT.
    03  TOTAL-QUANTITY
    03  QUANTITY-RATIO
```

คำสั่ง MOVE CORR INSPECTION TO QUANTITY REPORT.

มีความหมายว่า มูลค่าของฟิลด์ TOTAL-QUANTITY ของเรคคอร์ด INSPECTION จะถูกย้ายไปยังฟิลด์ TOTAL-QUANTITY ของเรคคอร์ด QUANTITY-REPORT ในทำนองเดียวกันฟิลด์ QUANTITY-RATIO ก็เช่นกัน

การเคลื่อนย้ายข้อมูลไม่ทำให้ข้อมูลเดิม (original data) หาย เพียงแต่เป็นการก๊อบปี้ (copy) ค่าเดิมขึ้นใหม่ในเนื้อที่ซึ่งกำหนดให้ data-name-1 หรือ literal-1 เป็น source item, data-name-2, data-name-3 และอื่น ๆ เป็น receiving items ทั้ง source item และ receiving item อาจจะเป็นข้อมูลเดี่ยวหรือข้อมูลกลุ่มก็ได้ (ในกรณี literal ถือว่าเป็นข้อมูลเดี่ยว)

กฎทั่วไปของคำสั่ง MOVE

1. ในกรณีย้าย source item และ receiving item ที่เป็นข้อมูลเดี่ยวทั้งคู่ เรียกว่า elementary move

elementary item → elementary item

ข้อมูลเดียวแต่ละตัว ต้องเป็นตัวใดตัวหนึ่งใน numeric, alphabetic, alphanumeric, numeric edited หรือ alphanumeric edited (ดูรายละเอียดของ PICTURE clause ใน Data division)

สำหรับ numeric literals ถือว่าเป็น numeric ส่วน non-numeric literals ถือว่าเป็น alphanumeric

2. เมื่อ receiving item เป็น alphanumeric edited, alphanumeric, หรือ alphabetic item

a) ถ้านิยามด้วย JUSTIFIED clause ตำแหน่งของตัวอักขระที่ไม่ใช้จะเป็น blanks ตามหลัก justification

b) ถ้าขนาดของ source item มากกว่าขนาดของ receiving item ตัวอักขระส่วนที่เกินจะถูกตัดทิ้ง หลังจาก receiving item เต็ม

ตัวอย่าง

Source data	Picture	Receiving item
A B C D	X(4)	A B C D
A B C 1 2 3	X(8)	A B C 1 2 3 ␣ ␣
A B C D E	X(4)	A B C D
A B 1 2 3	X(3)	A B 1

c) ถ้า source item มีเครื่องหมายกำกับ, เครื่องจะใช้ค่าสัมบูรณ์ (absolute value)

3. เมื่อ receiving item เป็น numeric หรือ numeric edited

a) ให้ยึดจุดทศนิยมเป็นหลัก ตำแหน่งของตัวอักขระที่ไม่ใช้เครื่องจะใส่เลขศูนย์ ยกเว้น เมื่อมีการแทนที่เลขศูนย์ อันเนื่องมาจากการ edit

b) ค่าสัมบูรณ์ของ source item จะนำมาใช้ ถ้า receiving item ไม่มีเครื่องหมาย (no operational sign)

c) ถ้า source item มีตัวเลขทางซ้ายมือของจุดทศนิยมหรือตัวเลขทางขวามือของจุดทศนิยม มากกว่าที่กำหนดใน receiving item, ตัวเลขส่วนเกินเครื่องจะตัดทิ้ง

ตัวอย่าง

Source data	Picture	Receiving item
1 2 3 4 5	$**9.99	$ 1 2 3 . 4 5
1 2 3 4 5	999.9	1 2 3 . 4
0 0 0 1 2	$**9.99	$ * * 0 . 1 2

d) ถ้า source item มีตัวอักขระชนิด non-numeric อยู่ด้วย ผลลัพธ์ขณะ execute โปรแกรมจะไม่ทราบค่า (unpredictable)

4. การเปลี่ยนรูปข้อมูลใด ๆ จากรูปแบบหนึ่งที่เก็บภายในหน่วยความจำไปเป็นอีกรูปแบบหนึ่งระหว่างย้ายข้อมูล เป็นไปตามการ edit ตำแหน่งใน receiving itemg

5. การย้ายใด ๆ ที่ไม่ใช่ elementary move* เรียกว่า alphanumeric elementary move

---

* elementary move หมายถึงทั้ง source item และ receiving item เป็นข้อมูลเดี่ยวทั้งคู่

elementary item → group

group → elementary item

และ group → group

ยกเว้นไม่มีการเปลี่ยนรูปข้อมูลจากรูปแบบหนึ่งที่เก็บภายในหน่วยความจำไปเป็นอีกรูปแบบหนึ่ง ในการย้ายข้อมูลเช่นนี้ เนื้อที่ของ receiving จะใส่ข้อมูลโดยไม่สนใจว่าเป็นข้อมูลเดี่ยวหรือข้อมูลกลุ่ม ทั้ง source item หรือ receiving item

6. เมื่อตัวถูกกระทำ source และตัวถูกกระทำ receiving ของคำสั่ง MOVE ใช้เนื้อที่บางส่วนในหน่วยความจำร่วมกัน (นั่นคือ ตัวถูกกระทำ overlap กัน) ผลลัพธ์ของการ execute จะไม่ทราบค่า (unperdictable)

ตัวอย่าง

Source data	Picture of receiving item	Receiving item
'ABCD'	X(4)	A B C D
'ABC-123'	X(7)	A B C - 1 2 3
'123'	X(2)	1 2
ALL QUOTES	X(6)	" " " " " " "

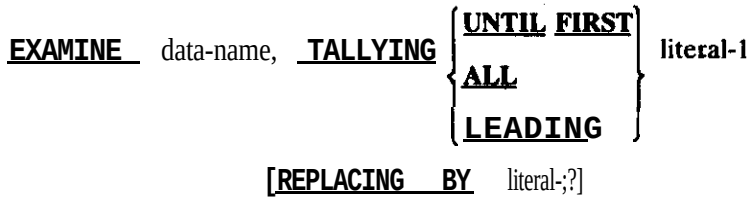
ตัวอย่าง

```
MOVE FIELDA TO PACKED-FIELD.
MOVE NET-PAY TO PRINT-NET-PAY.
MOVE SPACES TO XX1, XX2, XX3.
MOVE ZEROS TO AVERAGE, TOTAL, DIFF.
```

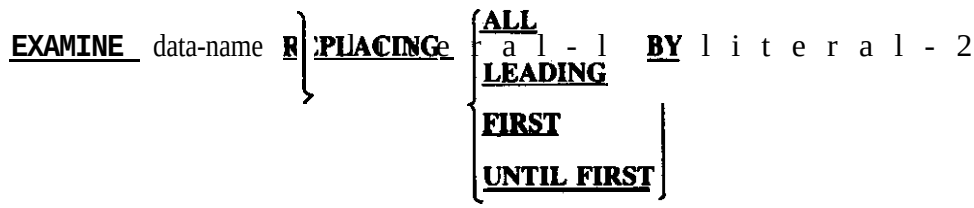
คำสั่ง EXAMINE

คำสั่งนี้ใช้ในการนับจำนวนตัวอักขระตัวใดตัวหนึ่งที่กำหนดไว้ว่าปรากฏใน data item นั้นกี่ตัว และ/หรือ ใช้แทนที่ตัวอักขระตัวหนึ่งด้วยตัวอักขระอีกตัวหนึ่ง มี 2 รูปแบบด้วยกันคือ

รูปแบบที่ 1



รูปแบบที่ 2



ในทุกกรณี รายละเอียดของ data-name ต้องมีการใช้เป็นแสดงผล (usage is display)

เมื่อ data-name แทน non-mumeric data item การตรวจสอบเริ่มจากตัวอักขระซ้ายมือสุดไปทางขวามือ ตัวอักขระทุกตัวในข้อมูลนั้น (data item) จะได้รับการตรวจสอบลักษณะนี้

เมื่อ data-name แทน numeric data item ข้อมูลนี้จะต้องมีแต่ตัวเลขเท่านั้น และอาจจะเป็นเครื่องหมายกำกับ การตรวจสอบเริ่มจากตัวอักขระซ้ายมือสุดไปทางขวามือ ตัวอักขระทุกตัวได้รับการตรวจสอบ

ถ้าในรายละเอียดของ picture clause ของข้อมูลตัวนั้นมี ตัวอักษร S ซึ่งหมายถึงเครื่องหมายกำกับตัวเลข (an operational sign) ในคำสั่ง EXAMINE เครื่องจะไม่สนใจ (ignored) เครื่องหมายนี้

literal-1 และ literal-2 ต้องเป็นตัวอักขระเพียงหนึ่งตัวซึ่งอยู่ในชั้น (class) เดียวกับ data-name นอกจากนี้แล้ว literal แต่ละตัวอาจจะเป็น figurative constant ตัวไหนก็ได้ยกเว้นเป็น ALL ไม่ได้ ถ้า data-name เป็น numeric, literal-1 และ literal-2 ต้องเป็นเลขจำนวนเต็มไม่มีเครื่องหมายกำกับ (unsigned integer) หรือ figurative constant ZERO (ZEROS, ZEROES)

ในรูปแบบที่ 1 จำนวนที่นับได้นั้น จะไปแทนที่ค่าของ special register ชื่อ TALLY ซึ่งรายละเอียดของมันเป็นเลขจำนวนเต็มไม่มีเครื่องหมาย 5 หลัก

1. เมื่อใช้ option ALL, จำนวนที่นับได้นี้ แทนจำนวนการเกิดของ literal-1

2. เมื่อใช้ option LEADING, จำนวนที่นับได้นี้ แทนจำนวนการเกิดของ literal-1 ก่อนจะถึงตัวอักขระตัวอื่นที่ไม่ใช่ literal-1

3. เมื่อใช้ option UNTIL FIRST, จำนวนที่นับได้นี้แทนจำนวนตัวอักขระทั้งหมดที่นับได้ก่อนการเกิดของ literal-1

สำหรับ option REPLACING ทั้งในรูปแบบที่ 1 และรูปแบบที่ 2 มีกฎการแทนที่อย่างเดียวกัน ดังนี้

1. เมื่อใช้ option ALL, literal-2 จะไปแทนที่การเกิดของ literal-1 ทุกตัว

2. เมื่อใช้ option LEADING, การแทนที่ของ literal-2 สำหรับการเกิดแต่ละตัวของ literal-1 หยุดเมื่อตัวอักขระตัวนั้นไม่ใช่ literal-1 หรือพบขอบเขตทางขวามือ (right-hand boundry) ของ data item นั้น

3. เมื่อใช้ option UNTIL FIRST, การแทนที่ของ literal-2 หยุดเมื่อพบ literal-1 หรือพบขอบเขตทางขวามือของข้อมูลตัวนั้น

4. เมื่อใช้ option FIRST, การเกิดครั้งแรกของ literal-1 ถูกแทนด้วย literal-2

ตัวอย่าง

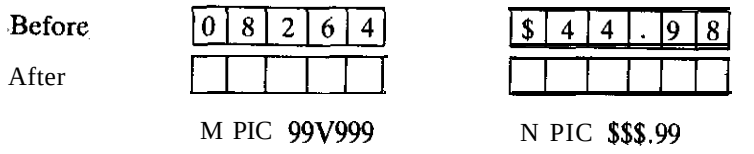
การตรวจสอบข้อมูล

คำสั่ง EXAMINE	ITEM-1 Data (before)	(after)	ค่าของ TALLY
EXAMINE ITEM-l TALLYING ALL 0	101010	101010	3
EXAMINE ITEM-1 TALLYING ALL 1 REPLACING BY 0	101010	000000	3
EXAMINE ITEM-l REPLACING LEADING "*" BY SPACE	**7000	-7000	+
EXAMINE ITEM-l REPLACING FIRST "*" BY "$"	**1.94	$ * 1.941	+

+ หมายถึง unchanged

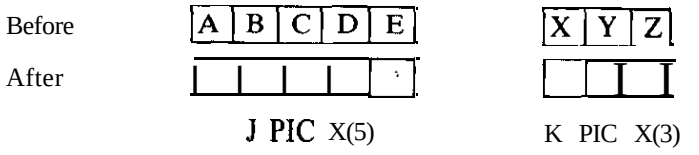
แบบฝึกหัด จงเติมมูลค่าของ item แต่ละตัวลงในบล็อกที่ว่างให้ถูกต้อง

1. คำสั่ง MOVE M TO N.

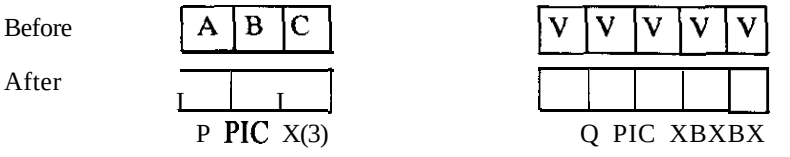


2. คำสั่ง MOVE C TO D.

3. คำสั่ง MOVE J TO K



4. คำสั่ง MOVE P TO Q.



5. จากตารางข้างล่างนี้ จงเติมผลลัพธ์ลงใน receiving items ให้ถูกต้อง

	Picture of source field	Content of source field,	Move to field with picture	Printed result
1	99V9	344	ZZ.9CR	
2	S9(4)	-2550	9999CR	
3	S9V99	-055	Z.ZZDB	
4	999	123	9909	
5	9999	1234	9,999	
6	99	12	9B9	

	Picture of source field	Content of source field	Move to field with picture	Printed result
7	9(7)	1234567	$Z,ZZZ,Z99	
8	9(3)V9(2)	00454	$ZZZ.ZZ	
9	9(3)V9(2)	00433	***.**	
10	S9(4)	+0133	-9(4)	

6. จงใส่มูลค่าของ report items ในตารางข้างล่างนี้

	Source data	Picture	Report item
a)	2 8 7 5	99.99+	
b)	1 2 3	999DB	
c)	1 2 3	$9.99BCR	
d)	4 4 3 8	-9(4)	
e)	1 2 3 4 5 6	9(2)B(2)9(4)	
f)	0 0 0 1 2 3	$$$,$$$.99	

7. ใน Data division ถ้าเราเตรียมเนื้อที่ไว้ดังนี้

01 AMOUNT PICTURE 9(4).

01 DAYS-ELAPSED PICTURE X(9).

จงพิจารณาคำสั่งใน Proceduredivision ข้างล่างนี้ว่าถูกต้องหรือไม่

a) MOVE “ZERO” TO AMOUNT

b) MOVE SPACES TO AMOUNT

c) ADD 50 TO DAY-ELAPSED.

d) MOVE ZERO TO DAY-ELAPSED.

e) MOVE “ZERO” TO DAY-ELAPSED,

8. กำหนดให้เครื่องคอมพิวเตอร์ execute 2 คำสั่งข้างล่างนี้

MOVE A TO B.

MOVE B TO C.

เมื่อนิยาม C ด้วย picture X(11) จงบอกมูลค่าของ C ในแต่ละกรณี

	มูลค่าของ A	picture ของ B	มูลค่าของ c
1)	10.125	9999V99	
2)	10000.00	Z,ZZZ.ZZ	
3)	900.15	$$,$$$.99	
4)	0.08	$$,$ZZ.ZZ	
5)	5 0 . 5 0	$***.99DB	
6)	-25.25	+,+ + +.99	
7)	25.25	$$$$.99	
8)	WELCOME	XXXBXXX	
9)	061087	99/99/99	
10)	ALL QUOTES	X(5)	

9. กำหนดเนื้อที่ในหน่วยความจำชื่อ X, Y และ Z ซึ่งมีค่าแรกอยู่ในบรรทัดที่หนึ่งของตารางข้างล่างนี้ หลังจาก execute แต่ละคำสั่งให้บอกค่าปัจจุบันของ data-name ทั้ง 3 ตัวนี้

คำสั่ง	X	Y	Z
ค่าแรก	10	18	20
1) MOVE X TO Y.			
2) MOVE X TO 15.			
3) MOVE 15 TO X.			
4) MOVE X TO Y, Z.			
5) MOVE Y TO X, Z.			
6) MOVE ZEROS TO X.			
7) ADD Y TO X.			
8) SUBTRACT Y FROM Z.			
9) DIVIDE X INTO Z GIVING Y.			
10) MULTIPLY X BY Y.			

6.1.3 Sequence control statement

ได้แก่ คำสั่ง GO TO, ALTER, PERFORM และ STOP โดยปกติเครื่องคอมพิวเตอร์จะทำการ execute statements, sentences, และ paragraphs ตามลำดับคำสั่งที่ปรากฏทีละบรรทัด ยกเว้นในกรณีที่มี verb ดังกล่าวข้างต้น จะทำให้การทำงานไม่เป็นไปตามลำดับขั้นเหมือนเดิม

คำสั่ง GO TO

คำสั่งนี้ บอกให้ transfer permanent control ไปยังตำแหน่งใดตำแหน่งหนึ่งในโปรแกรม แบ่งออกเป็นสองชนิด โดยมีรูปแบบดังนี้

รูปแบบที่ 1

GO TO [procedure-name]

ถ้ามีชื่อพารากราฟตามหลัง verb GO TO, คำสั่งนี้หมายถึงให้ย้าย control อย่างไม่มีเงื่อนไข ไปยังจุดเริ่มต้นของพารากราฟ procedure-name หรือ section ซึ่งกรณีนี้จะทำให้คำสั่งอื่น ๆ ที่อยู่ในลำดับถัดไปในพารากราฟเดิมไม่ถูก execute

ตัวอย่าง

```
PARA-2.
    READ INPUT-FILES AT END GO TO PARA-3.


    GO TO PARA-2
PARA-3.
    CLOSE INPUT-FILES.
    . . .
```

ถ้าไม่มี procedure-name ตามหลัง verb GO TO, procedure-name นี้ จะได้จากคำสั่ง ALTER ซึ่งอยู่ในลำดับก่อนหน้าที่จะมา execute คำสั่ง GO TO นี้

รูปแบบที่ 2

GO TO procedure-name-1 [procedure-name-2....procedure-name-n]

DEPENDING ON data-name

รูปแบบนี้เป็นการ transfer control ไปยังพารากราฟใดพารากราฟหนึ่งใน procedure-names ทั้งหลายที่กำหนดไว้ ทั้งนี้ขึ้นอยู่กับค่าของ data-name item ซึ่งคำนวณได้ระหว่าง execute

data-name ต้องเป็นเลขจำนวนเต็มบวก มีค่าเท่ากับ 1,2,3,....,n ถ้า data-name มีค่า

เท่ากับ 1 ให้ transfer control ไปยังพารากราฟ procedure-name-1

ถ้า data-name = 2 ให้ทำพารากราฟ procedure-name-2

⋮

ถ้า data-name = n ให้ทำพารากราฟ procedure-name-n

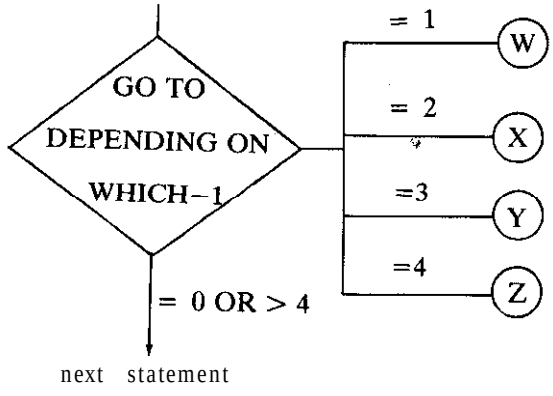
แต่ถ้ามูลค่าของ data-name item ไม่ใช่เลขจำนวนเต็ม (integer) ระหว่าง 1 ถึง n จะไม่มีการ transfer ใด ๆ เกิดขึ้น control ก็จะไปทำคำสั่งถัดไป

ตัวอย่าง

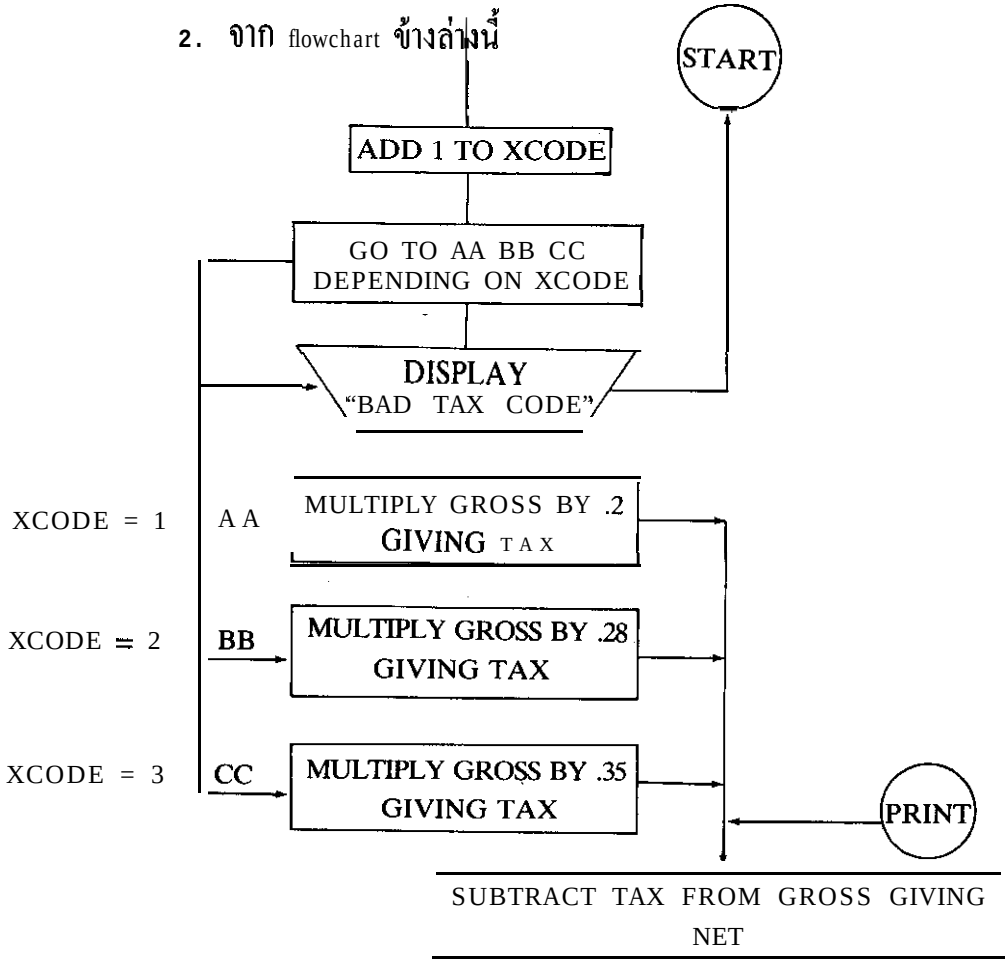
```
READ CARD-FILE AT END GO TO END-OF-JOB.
GO TO BEGIN, PRINT, FINISH DEPENDING ON DATA-I.
```

แบบฝึกหัด

1. จงเขียนคำสั่งจาก flow chart ข้างล่างนี้ เมื่อ W, X, Y และ Z เป็นชื่อพารากราฟ



2. จาก flowchart ข้างล่างนี้



สามารถเขียนเป็น conditional GO TO statements ได้ดังนี้

บรรทัดที่ 1 ADD 1 TO XCODE.

2 GO TO AA BB CC DEPENDING ON XCODE.

3 DISPLAY "BAD TAX CODE".

4 GO TO START.

5AA.MULTIPLY GROSS BY .2 GIVING TAX.

6 GO TO PRINTS.

7BB.MULTIPLY GROSS BY .28 GIVING TAX.

8 GO TO PRINTS.

9CC.MULTIPLY GROSS BY .35 GIVING TAX.

10PRINTS. SUBTRACT TAX FROM GROSS GIVING NET.

บรรทัดที่

a) ถ้า XCODE มีค่าเท่ากับ 0, บรรทัดที่จะถูก execute คือ .....

b) ถ้า XCODE มีค่าเท่ากับ 1, บรรทัดที่จะถูก excute คือ .....

c) ถ้า XCODE มีค่าเท่ากับ 2, บรรทัดที่จะถูก execute คือ .....

d) ถ้า XCODE ไม่ใช่ 0 หรือ 1 หรือ 2 บรรทัดที่จะถูก execute คือ .....

เฉลย

a) 1, 2, 5, 6, 10

b) 1, 2, 7, 8, 10

c) 1, 2, 9, 10

d) 1, 2, 3, 4

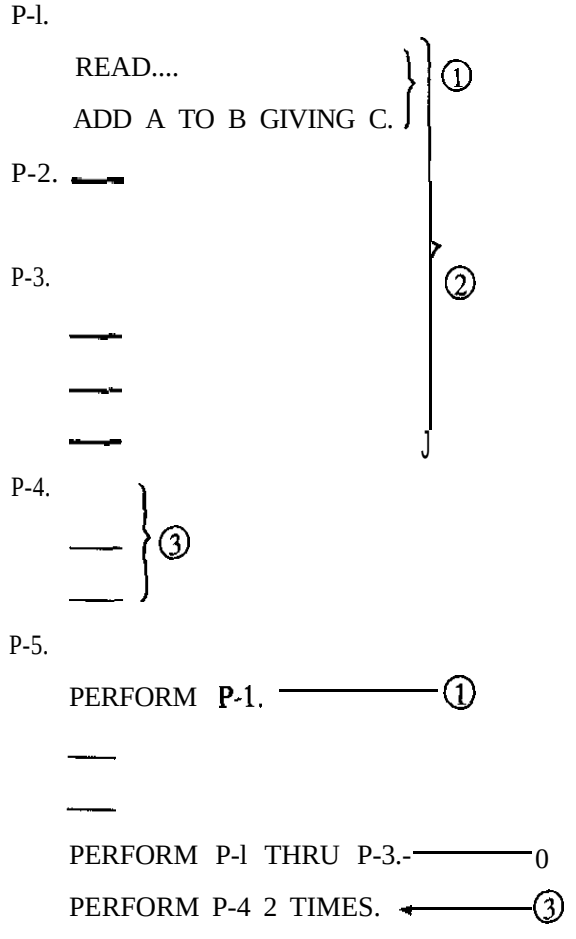
คำสั่ง PERFORM

คำสั่งนี้ให้ย้าย control ชั่วคราวไป execute พารากราฟใดพารากราฟหนึ่งเป็นจำนวนครั้งเท่าที่กำหนดไว้ เมื่อทำงานเสร็จแล้วให้ย้อนกลับ (return) มายังคำสั่งที่อยู่หลังคำสั่ง PERFORM แต่ถ้าคำสั่งสุดท้ายที่ทำการ execute ก่อนที่ control จะกลับมายังคำสั่งหลังคำสั่ง PERFORM เป็นคำสั่ง GO TO, control จะกลับมายังคำสั่งหลัง PERFORM ไม่ได้ นั่นหมายความว่า คำสั่งสุดท้ายที่กล่าวถึงนี้จะเป็นคำสั่ง GO TO ไม่ได้

มีรูปแบบดังนี้

PERFORM procedure-name-1 [**THRU** procedure-name-2] [{data-name-1 / integer-1} **TIMES**]

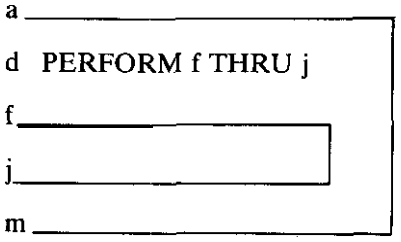
ตัวอย่าง



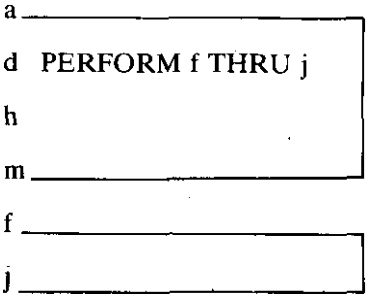
ตัวอย่าง

ลักษณะของคำสั่ง PERFORM ที่ถูกต้อง (Valid)

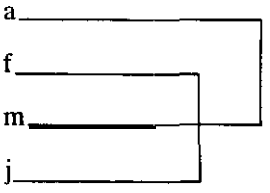
x PERFORM a THRU m



x PERFORM a THRU m



x PERFORM a THRU m

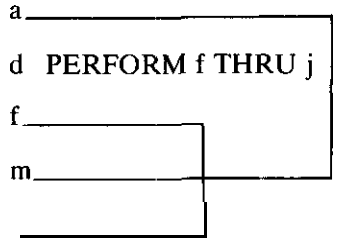


d PERFORM f THRU j

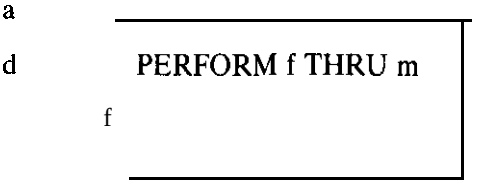
ตัวอย่าง

ลักษณะของคำสั่ง PERFORM ที่ใช้ไม่ได้ (Invalid)

x PERFORM a THRU m

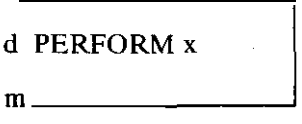


x PERFORM a THRU m

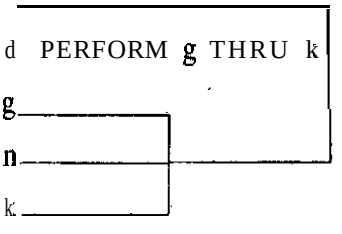


k

x PERFORM a THRU m



x PERFORM b THRU n



สำหรับเครื่อง IBM **หรือเครื่อง** VAX

มีรูปแบบดังนี้

(1) PERFORM procedure-name-l [**THRU** procedure-name-21

(2) PERFORM procedure-name-l [**THRU** procedure-name-2]

$$\left\{ \begin{array}{l} \text{data-name-1} \\ \text{integer-1} \end{array} \right\} \textbf{TIMES}$$

(3) PERFORM procedure-name-l [**THRU** procedure-name-21

UNTIL condition-1

ถ้า condition-1 **เป็นจริงเมื่อไหร่ เครื่องจะกลับมาทำคำสั่งซึ่งอยู่ถัดจากคำสั่ง PERFORM** (next sentence)

รูปแบบที่ 3 เงื่อนไข UNTIL จะถูกตรวจสอบเป็นอันดับแรก ก่อน procedure(s) จะถูก execute ถ้าเงื่อนไขเป็นจริง procedure(s) เหล่านั้นจะไม่ได้รับการ execute เลย เขียน flowchart ดังนี้

ตัวอย่าง

ถ้าเรานิยามฟิลด์ A, B, N **และ** I **ใน Working-storage** section **ดังนี้**

77 A PIC 9(2) VALUE 0.

77 B PIC 9(2) VALUE 0.

77 N PIC 9(2) VALUE 5.

77 I PIC 9(2) VALUE 0.

ให้บอกว่าโปรแกรมข้างล่างนี้ทำงานเกี่ยวกับอะไร

PERFORM ADD-AB UNTIL I = 5.

หรือ PERFORM ADD-AB 5 TIMES.

หรือ PERFORM ADD-AB VARYING I FROM 1 BY 1 UNTIL I > 5.

COMPUTE MEAN = B I N.

GO TO PARA-4.

ADD-AB.

ADD 1 TO A.

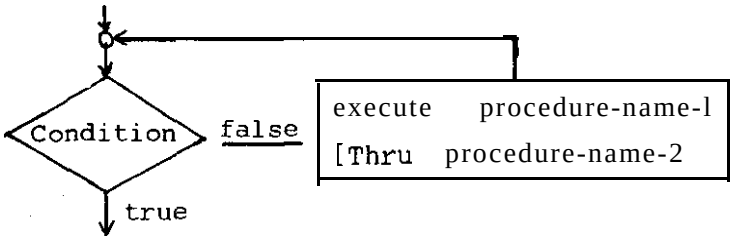
ADD A TO B.

ADD 1 TO I.

PARA-4.

ขั้นตอนการทำงาน

a	iii-	N	I
0	0	5	0
1	1		1
2	3		2
3	6		3
4	10		4
5	15		5



MEN = $\frac{15}{5}$

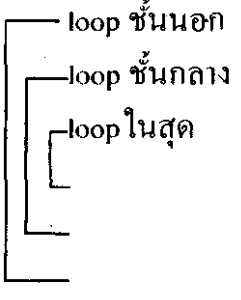
(4) **PERFORM** procedure-name-l [**THRU** procedure-name-21

* VARYING data-name-l **FROM** {data-name-2 / literal-2} **B**{literal-3 / data-name-3} **UNTIL** condition-1

** [**AFTER** data-name-4 **FROM**{data-name-5 / literal-5} **I**{literal-6 / data-name-6} **UNTIL** condition-2]

* [**AFTER** data-name-7 **FROM**{data-name-8 / literal-8} **I**{literal-9 / data-name-9} **UNTIL** condition-3]

รูปแบบข้างต้น มีลักษณะดังนี้



ตัวอย่าง 1

PP-3.

PERFORM **TT-RTN** VARYING I FROM 1 BY 1 UNTIL I > 5.

---

* บรรทัดนี้แทน loop ชั้นนอก

** บรรทัดนี้แทน loop ชั้นกลาง

*** บรรทัดนี้แทน loop ชั้นในสุด

PERFORM RR-RTN VARYING J FROM 1 BY 1 UNTIL J > 6.

BB-RTN.

TT-RTN.

MOVE 0 TO ACT (I).

RR-RTN.

MOVE 0 TO MO (J).

จากกลุ่มของคำสั่งข้างต้นนี้เป็นการกำหนดให้ data-name ACT (1), ACT (2)..... ACT (5) ทุกตัวมีมูลค่าเท่ากับ 0 และมูลค่าของ data-name MO (1), MO (2) ... MO (6) อีก 6 ตัวให้เท่ากับ 0 เช่นกัน

ตัวอย่าง 2

PP-2.

PERFORM MOVE-ZERO VARYING I FROM 1 BY 1 UNTIL I > 5

AFTER J FROM 1 BY 1 UNTIL J > 6.

PP-3.

MOVE -ZERO.

MOVE 0 TO BAL-1 (I J), BAL-2 (I J), T-BAL (I J).

ขั้นตอนการทำงาน

เครื่องจะทำคำสั่งทั้งหมดภายใน loop ในสุด ให้หมดก่อนแล้วจึงจะทำ loop นอก ดังนี้

I = 1, J = 1, 2, 3, 4, 5, 6,

I = 2, J = 1, 2, 3, 4, 5, 6,

I = 5, J = 1, 2, 3, 4, 5, 6,

ตัวอย่าง 3

READ CARD-FILE RECORD AT END MOVE "YES" TO END-DATA.

PERFORM READ-PRINT UNTIL END-DATA = "YES".

READ-PRINT.

คำสั่ง STOP

คำสั่งนี้สั่งให้เครื่องหยุดชั่วคราว หรือหยุด execute งานนี้ ขณะที่กำลัง execute object program โดยมีรูปแบบดังนี้

$$\underline{\text{STOP}} \left\{ \begin{matrix} \text{literal} \\ \underline{\text{RUN}} \end{matrix} \right\}$$

1) ถ้ามี literal ตามหลัง STOP เครื่องคอมพิวเตอร์จะหยุด execute แล้วพิมพ์ literal ออกมาทางเครื่องพิมพ์ดีดบนคอนโซลหรือเทอร์มินัล จากนั้นก็หยุดจนกว่าพนักงานควบคุมเครื่องหรือโปรแกรมเมอร์จะมากดสวิทช์ GO, หรือ Return คีย์ เครื่องก็จะทำ next sentence ต่อไป

2) ถ้ามี RUN ตามหลัง STOP หมายความว่า โปรแกรมนั้นหมดแล้ว ให้หยุด execute, STOP RUN จึงใช้เป็นคำสั่งสุดท้ายของ sequence ในโปรแกรมเท่านั้น

ตัวอย่าง

```
        IF M-CHECK IS NOT EQUAL TO ZERO, STOP "ERROR ON
            MASTER CARD COLUMN 1".



CLOSE-FILES.
        CLOSE M-FILE, P-FILE.
        STOP RUN.
```

แบบฝึกหัด

1. จงเขียนชุดของคำสั่งข้างล่างนี้ใหม่ โดยใช้คำสัง PERFORM...UNTIL... หนึ่งคำสั่ง

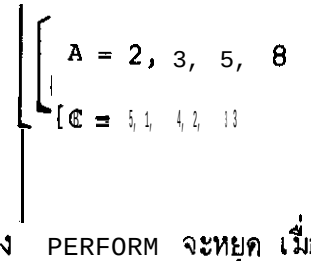
```
PERFORM    GET-INPUT.
PERFORM    GET-INPUT.
PERFORM    GET-INPUT.
PERFORM    GET-INPUT.
```

2. จงบอกจำนวนครั้งที่เครื่องจะ performed พารากราฟ PARA-A จากชุดของคำสั่งข้างล่างนี้

```
MOVE 11 TO TIMES-COUNTER.
PERFORM PARA-A UNTIL TIMES-COUNTER GREATER THAN IO.
```

3. จงบอกค่าของ A, B และ C เมื่อเครื่อง execute พารากราฟ TEST-ABC และคำสั่ง PERFORM จะหยุดเมื่อ A, B และ C มีค่าเท่าไร

```
PERFORM TEST-ABC
    VARYING A FROM 2 BY 3
        UNTIL A GREATER THAN 7
    AFTER B FROM 5 BY -1
        UNTIL B LESS THAN 3
    AFTER C FROM 1 BY 1
        UNTIL C GREATER THAN 3.
```



คำสั่ง PERFORM จะหยุด เมื่อ A=8, B=5, C=1

6.1.4 Input/Output statements

การเคลื่อนที่ของข้อมูลในเครื่องคอมพิวเตอร์ถูกควบคุมโดยโปรแกรมควบคุมระบบ (DOS) คำสั่งภาษาโคบอลที่จะกล่าวถึงในหัวข้อนี้ใช้เพื่อเริ่มต้นเคลื่อนที่ข้อมูลภายในแฟ้มข้อมูลไปยังตัวกลางบันทึกข้อมูลภายนอก และเคลื่อนข้อมูลจากตัวกลางบันทึกข้อมูลภายนอกเข้ามายังคอมพิวเตอร์ และการควบคุมข้อมูลที่มีจำนวนน้อย (low-volume information) ที่ได้มาจากหรือส่งไปยัง input/output device เช่น เครื่องอ่านบัตร หรือ เครื่องพิมพ์ดีดบนคอนโซล

Disk Operating System เป็นระบบการประมวลผลเรคคอร์ด นั่นคือ หน่วยของข้อมูลที่จะอ่านโดยคำสั่ง READ หรือบันทึกโดยคำสั่ง WRITE คือ **เรคคอร์ด** ผู้ใช้ภาษาโคบอลจำเป็นต้องเกี่ยวข้องเฉพาะกับการใช้เรคคอร์ดของแต่ละคนเท่านั้น การจัดหาจะเป็นไปอย่างอัตโนมัติสำหรับการปฏิบัติการ เช่น การเคลื่อนย้ายข้อมูลไปไว้ในบัฟเฟอร์ และ/หรือ ภายในหน่วยความจำหลัก, การตรวจสอบความถูกต้อง (validity checking), การแก้ไขที่ผิด (error correction), การไม่บล็อกและการบล็อก (unblocking and blocking), และวิธีการสวิทช์ volume (volume switching procedures)

คำศัพท์ที่ใช้อธิบายรายละเอียดในหัวข้อนี้มี volume และ reel คำว่า volume หมายถึง all input/output devices คำว่า reel หมายถึง เฉพาะ tape devices เท่านั้น

คำสั่ง OPEN

คำสั่งนี้เริ่มต้นการประมวลผลของแฟ้มข้อมูลชนิด input, output, และ input - output และยังทำการตรวจสอบ และ/หรือ บันทึก label และทำการปฏิบัติการ input/output อื่น ๆ

มีรูปแบบดังนี้

```
OPEN [ INPUT { file-name [ REVERSED          ] } ... ]
                           [ WITH NO REWIND  ]
     [OUTPUT {file-name WITH NO REWIND} ...]
     [ I-O {file-name}
```

คำสั่ง OPEN จะต้องมี option INPUT, OUTPUT หรือ I-O อย่างน้อย 1 อย่าง ในแต่ละ option อาจจะมี file-name มากกว่า 1 ชื่อได้ แต่ในคำสั่งนั้นจะมี option แต่ละอันมากกว่า 1 ครั้งไม่ได้ ทั้งสาม options เรียงลำดับอย่างไรก็ได้

เมื่อ file - name ต้อง นิยามมาแล้วโดย file description entry ใน Data division

คำสั่ง OPEN สำหรับแฟ้มข้อมูลแต่ละชุดต้องถูก execute ก่อนคำสั่ง READ คำสั่งแรก หรือคำสั่ง WRITE คำสั่งแรกของแฟ้มข้อมูลนั้น คำสั่ง OPEN คำสั่งที่สองสำหรับแฟ้มข้อมูลเดียวกันนั้นไม่สามารถ execute ได้ก่อนการ execute ของคำสั่ง CLOSE ของแฟ้มข้อมูลนั้น คำสั่ง OPEN ไม่ได้เข้าถึงหรือปล่อยข้อมูลเรคคอร์ดแรก คำสั่ง READ หรือคำสั่ง WRITE ต้องได้รับการ execute ในการเข้าถึงหรือปล่อยตามลำดับ สำหรับข้อมูลเรคคอร์ดแรก

เมื่อมีการตรวจสอบ หรือการบันทึก label แรก คำสั่ง OPEN ทำให้การเริ่มต้น label subroutine ของผู้ใช้เครื่องได้รับการ execute

option REVERSED และ option NO REWIND ใช้ได้เฉพาะกับ sequential single reel files

option REVERSED ใช้ได้กับแฟ้มข้อมูลซึ่งประกอบด้วย fixed-lenght (F mode) เรค-คอร์ดเท่านั้น

สำหรับเทปไฟล์ให้ใช้กฎต่อไปนี้

1. ถ้าไม่มี option REVERSED และไม่มี option NO REWIND การ execute ของคำสั่ง OPEN ทำให้แฟ้มข้อมูลนั้นไปอยู่ที่ตำแหน่งต้นม้วน

2. เมื่อมีการใช้ option REVERSED หรือ option NO REWIND การ execute ของคำสั่ง OPEN ไม่ทำให้แฟ้มข้อมูลนั้นเปลี่ยนตำแหน่ง เมื่อมี option REVERSED แฟ้มข้อมูลนั้นต้องกำหนดตำแหน่งไปที่ปลายม้วน และเมื่อมี option NO REWIND แฟ้มข้อมูลนั้นต้องกำหนดตำแหน่งไปที่ต้นม้วน

เมื่อกำหนด option REVERSED คำสั่ง READ ของแฟ้มข้อมูลนั้นทำให้ data เรคคอร์ดของแฟ้มข้อมูลใช้ได้ (available) ในลักษณะย้อนกลับ (reversed order) นั่นคือเริ่มต้นที่เรคคอร์ดสุดท้าย

option I-O ทำให้มีการเปิดของ a mass storage file ทั้ง input และ output operation เนื่องจาก option นี้ implies ว่ามีแฟ้มข้อมูลอยู่แล้ว และจะใช้ไม่ได้ถ้า mass storage file ที่กำลังเพิ่งเริ่มสร้าง

เมื่อใช้ option I-O การ execute ของคำสั่ง OPEN จะรวมขั้นตอนต่อไปนี้

1. ตรวจสอบ label
2. label subroutine ของ user จะได้รับการ execute ถ้ามีการกำหนดไว้ใน USE sentence
3. บันทึก label

ตัวอย่าง 1

```
OPEN INPUT RECEIVABLE, NEW-ACCOUNTS.
OPEN OUTPUT NEW-PAYROLL-FILE
```

ตัวอย่าง 2

```
OPEN INPUT RECEIVABLE, NEW-ACCOUNTS, OUTPUT NEW-
PAYROLL-FILE.
```

คำสั่ง READ

คำสั่งนี้มีฟังก์ชันดังนี้

1. สำหรับการประมวลผล sequential file เพื่อทำให้ next logical record เอาไปใช้ได้จาก input file และให้ control กับ imperative-statement ที่ตามหลัง ในกรณีที่พบ end-of-file

2. สำหรับการประมวลผล random file เพื่อทำให้ เรคคอร์ดที่กำหนดเอาไปใช้ได้จาก mass storage file และให้ control กับ imperative-statement ที่ตามหลังถ้าพบว่าค่าของ ACTUAL KEY นั้นใช้ไม่ได้ มีรูปแบบดังนี้

```
READ file-name RECORD [INTO data-name]
    { AT END          }
    { INVALID KEY     } imperative-statement
```

คำสั่ง OPEN ต้อง execute ก่อนการ execute ของคำสั่ง READ ในแฟ้มข้อมูล

เมื่อคำสั่ง READ ได้รับการ execute, next logical record ในแฟ้มข้อมูลชื่อนี้จะเข้าถึงใน input area ที่ นิยามแล้วใน record description entry ชุดนี้

เรคคอร์ดยังคงอยู่ใน input area จนกระทั่ง input/output statement ถัดไปของแฟ้มข้อมูลได้รับการ execute

เมื่อแฟ้มข้อมูลนั้นประกอบด้วย logical record มากกว่า 1 ชนิด เรคคอร์ดเหล่านี้จะ share เนื้อที่หน่วยความจำที่เดียวกันโดยอัตโนมัติ

option INTO data-name ทำคำสั่ง READ ให้มีความหมายอย่างเดียวกับคำสั่ง READ บวกกับคำสั่ง MOVE, data-name ตัวนี้ต้อง นิยามใน Working-storage section หรือ Linkage section entry, หรือ output record ของแฟ้มข้อมูลที่เปิดแล้ว เมื่อใช้ option นี้ เรคคอร์ดปัจจุบันจะใช้ได้ (available) ใน input area เช่นเดียวกับในเนื้อที่ซึ่งกำหนดโดย data-name ข้อมูลจะถูกย้ายไปไว้ใน data-name ตามกฎของคำสั่ง MOVE ในโคบอล โดยไม่ต้องใช้ option CORRESPONDING

option AT END ต้องมีสำหรับแฟ้มข้อมูลทุกชุดที่มีการเข้าถึงแบบ sequential mode ถ้าระหว่างการ execute ของคำสั่ง READ พบ logical end ของแฟ้มข้อมูลนั้น control จะถูกส่งไปยัง imperative-statement ที่กำหนดใน AT END phrase หลังจากการ execute imperative-statement ที่ตามหลัง AT END phrase แล้ว คำสั่ง READ ของแฟ้มข้อมูลนั้นจะใช้ไม่ได้ถ้าไม่มีการ execute คำสั่ง CLOSE และคำสั่ง OPEN ของแฟ้มข้อมูลนั้นก่อน

ตัวอย่าง 1

```
READ FORECAST-FILE AT END, CLOSE FORECAST-FILE STOP RUN.
```

ตัวอย่าง 2

```
READ CD-FILE INTO INPUT-AREA AT END MOVE "YES" TO END-FILE.
IF END-FILE = "NO" WRITE OUT-RECORD.
```

คำสั่ง WRITE

คำสั่งนี้ สั่งให้นำ 1 logical record จาก output area ในหน่วยความจำ บันทึกออกมาทาง external unit โดยมีรูปแบบดังนี้

WRITE record-name [**FROM** data-name]

recore-name ในที่นี้ จะต้องเป็นเรคคอร์ด ซึ่ง นิยามใน file section เท่านั้น และก่อนที่จะเขียนคำสั่ง write นี้ แฟ้มข้อมูลซึ่งมีเรคคอร์ดนั้นอยู่จะต้อง open ก่อนเสมอ

ถ้ามีคำว่า FROM อยู่ในคำสั่ง หมายความว่า ให้เอาข้อมูลภายใต้ชื่อ data-name ซึ่งเป็นเนื้อที่ใด ๆ ในหน่วยความจำ write ออกมาทาง external unit โดยผ่าน record-name

ตัวอย่าง

```
WRITE PRINT-REC.
WRITE PRINT-REC FROM HEADING-L
WRITE PRINT-REC FROM HEADING-2.
```

คำสั่งแรก พิมพ์มูลค่า (content) ของ PRINT-REC
คำสั่งที่สอง พิมพ์มูลค่าของ HEADING-l
คำสั่งที่สาม พิมพ์มูลค่าของ HEADING-2

Carriage control เป็นการบังคับหรือ ควบคุมให้พรินเตอร์ หรือ เครื่องพิมพ์ดีด พิมพ์เอ้าท์พุท เรคคอร์ด ออกมาในลักษณะที่โปรแกรมเมอร์ต้องการ เช่น เว้น 1 บรรทัด, เว้น 2 บรรทัด พิมพ์ซ้ำบรรทัดเดิม หรือ ขึ้นหน้าใหม่ เป็นต้น โดยที่ตัวอักขระตัวแรกของเอ้าท์พุทเรคคอร์ดจะถูกนำไปใช้เป็น carriage control ดังนี้

character	Action before print	Action after print
+	no space	no space
blank (space)	space 1 line	no space
0 (zero)	space 2 lines	no space
-	space 3 lines	no space
1	skip to next page	no space

การกำหนด carriage control character ในการ defined เอ้าท์พุทเรคคอร์ดของเครื่อง CDC

```
DATA DIVISION.
FILE SECTION.
FD PRINT-FILE LABEL RECORD IS OMITTED DATA RECORD IS P-REC.
01 P-REC PICTURE X(136).
WORKING-STORAGE SECTION.
01 ABC.
    03 FILLER PICTURE X(40)     VALUE SPACES.
    03 A-ITEM PICTURE 9(10).
    03 FILLER PICTURE X(5)      VALUE SPACES.
    03 B-ITEM PICTURE 9(10).
    03 FILLER PICTURE X(5)      VALUE SPACES.
    03 C-ITEM PICTURE 9(10).
    03 FILLER PICTURE X(56)     VALUE SPACES.
```

ภายในหน่วยความจำของเครื่องคอมพิวเตอร์ เรคคอร์ด A B C จะใช้เนื้อที่ทั้งหมด 136 คอลัมน์ มีลักษณะดังนี้

1 40	41 50	51 55	56 65	66 70	71 80	81 136
	A-ITEM		B-ITEM		C-ITEM	

นั่นคือ คอลัมน์ 1 - 40 เป็นเนื้อที่ว่างทั้งหมด

41 - 50 เป็นเนื้อที่สำหรับใส่มูลค่าของ A–ITEM

51 - 55 เป็นเนื้อที่ว่าง

56 - 65 เป็นเนื้อที่สำหรับใส่มูลค่าของ B–ITEM

66 - 70 เป็นเนื้อที่ว่าง

71 - 80 เป็นเนื้อที่สำหรับใส่มูลค่าของ C–ITEM

81 - 136 เป็นเนื้อที่ว่างทั้งหมด

จะเห็นว่า ตัวอักขระ ตัวแรกของเอ้าท์พุทเรคคอร์ดชื่อ ABC เป็น blank ดังนั้น เมื่อเครื่อง execute คำสั่ง

```
WRITE P-REC FROM ABC.
```

เครื่องจะเว้นให้ 1 บรรทัดก่อนพิมพ์เรคคอร์ด A B C แต่ถ้าโปรแกรมเมอร์ต้องการให้เครื่องคอมพิวเตอร์เว้น 2 บรรทัดก่อนพิมพ์เรคคอร์ด ABC จะต้อง defined ให้ ตัวอักขระตัวแรกเป็น 0 ดังนี้

```
01 ABC.
   03 FILLER PICTURE X(40)    VALUE "0".
   03 A-ITEM PICTURE 9(10).
   03 FILLER PICTURE X(5)     VALUE SPACES,
   03 B-ITEM PICTURE 9(10).
   03 FILLER PICTURE X(5)     VALUE SPACES.
   03 C-ITEM PICTURE 9(10).
   03 FILLER PICTURE X(56)    VALUE SPACES.
```

เมื่อเครื่อง execute คำสั่ง

```
WRITE P-REC FROM ABC.
```

เครื่องจะเว้นให้ 2 บรรทัดก่อนพิมพ์ เรคคอร์ด A B C และถ้าต้องการให้เว้น 3 บรรทัดก่อนพิมพ์เรคคอร์ด ABC ต้อง defined, ดังนี้

```
01 ABC.
   03 FILLER PICTURE X(40) VALUE "-".
   03 A-ITEM PICTURE 9(10).
```

ถ้าต้องการให้พรินเตอร์ขึ้นกระดาษแผ่นใหม่ก่อนพิมพ์เรคคอร์ด ABC ต้อง defined ดังนี้

```
01 ABC
    03 FILLER PICTURE X(40)    VALUE " 1" .
    03 A-ITEM PICTURE 9(10)
```

แต่ถ้าต้องการ defined เรคคอร์ด ABC เพียงครั้งเดียวแล้วค่าของตัวอักขระตัวแรกของเรคคอร์ดนี้กำหนดเอาใน procedure division เราเพียงแต่กำหนดชื่อให้กับเนื้อที่นี้ ดังนี้

```
01 ABC.
    03 FLAG     PICTURE X(1).
    03 FILLER   PICTURE X(39)     VALUE SPACES.
    03 A-ITEM PICTURE 9(3).
```

เมื่อ	FLAG มีค่าเป็น	blank	เครื่องจะเว้น	1	บรรทัด
	FLAG มีค่าเป็น	0	เครื่องจะเว้น	2	บรรทัด
	FLAG มีค่าเป็น	–	เครื่องจะเว้น	3	บรรทัด
	FLAG มีค่าเป็น	1	เครื่องจะเว้น ขึ้นกระดาษแผ่นใหม่		
	FLAG มีค่าเป็น	+	เครื่องจะพิมพ์ซ้ำบรรทัดเดิม		

ตัวอย่างที่ 1

โปรแกรมข้างล่างนี้อ่านบัตรข้อมูลจำนวนหนึ่งในบัตรข้อมูลแต่ละใบ บันทึกรหัสนักศึกษา, ชื่อ, นามสกุล, และรายได้ต่อเดือน จากนั้นให้พิมพ์ข้อมูลทั้งหมดโดยเรียงลำดับจากนักศึกษาที่มีรายได้สูงสุดไปจนถึงรายได้ต่ำสุด แล้วคำนวณว่านักศึกษากลุ่มนี้จะมีรายได้เฉลี่ยต่อเดือนเท่ากับเท่าไร ?

```
       IDENTIFICATION DIVISION.
       PROGRAM-ID. NONSEE.
       ENVIRONMENT DIVISION.
       CONFIGURATION SECTION.
       SOURCE-COMPUTER. 3100 MEMORY SIZE IS 32K WORDS.
       OBJECT-COMPUTER. 3100 MEMORY SIZE IS 32K WORDS.
       INPUT-OUTPUT SECTION.
       FILE-CONTROL.
           SELECT CP-22 ASSIGN TO CARD-READER 60.
           SELECT BM-7 ASSIGN TO PRINTER 61.
       DATA DIVISION.
       FILE SECTION.
       FD CP-22 LABEL RECORD IS OMITTED DATA RECORD IS BI-REC.
       01 BI-REC.
           05 FILLER      PICTURE   X(4).
           05 MA-1.       PICTURE   999X9(5).
           05 FILLER      PICTURE   X(5).
           05 NAME-S      PICTURE   X(25).
           05 FILLER      PICTURE   X(5).
           05 MONEY-M     PICTURE   9(4).
           05 FILLER      PICTURE   X(28).
       FD BM-7 LABEL RECORD IS OMITTED DATA RECORD IS M-REC.
       01 M-REC PICTURE X(136).
       WORKING-STORAGE SECTION.
       77 A PICTURE 99      VALUE ZEROS.
       77 B PICTURE 99      VALUE ZEROS.
       77 C PICTURE 99      VALUE ZEROS.
       77 D PICTURE X(80)   VALUE SPACES.
       77 E PICTURE 9(6)    VALUE ZEROS.
       01 HEADING.
           05 FILLER      PICTURE X(66) VALUE SPACES.
           05 FILLER      PICTURE X(13) VALUE
                          "INCOME REPORT".
           05 FILLER      PICTURE X(63) VALUE SPACES.
       01 L-1.
           05 FILLER      PICTURE X(60) VALUE SPACES.
           05 IN-1        PICTURE X(13).
```

```
    05 FILLER      PICTURE X(63) VALUES SPACES.
01 L-2.
    05 FILLER      PICTURE X(136) VALUE SPACES.
01 L-3.
    05 FILLER PICTURE X(35) VALUE "0".
    05 IN-2   PICTURE X(66).
    05 FILLER PICTURE X(35) VALUE SPACES.
01 L-4.
    05 FILLER PICTURE X(38) VALUE "0".
    05 FILLER PICTURE X(4) VALUE "RANK".
    05 FILLER PICTURE X(5) VALUE SPACES.
    05 FILLER PICTURE X(11) VALUE "STUDENT NO".
    OS FILLER PICTURE X(11) VALUE SPACES.
    05 FILLER PICTURE X(12) VALUE "STUDENT NAME".
    05 FILLER PICTURE X(10) VALUE SPACES.
    05 FILLER PICTURE X(6) VALUE "INCOME".
    05 FILLER PICTURE X(39) VALUE SPACES.
01 L-5.
    05 FILLER              PICTURE X(39) VALUE SPACES.
    05 IN-3                PICTURE 99.
    05 FILLER              PICTURE X(3) VALUE SPACES.
    05 IN-4                PICTURE X(80).
    05 FILLER              PICTURE X(12) VALUE SPACES.
oi L-6.
    05 REC.1 OCCURS 10 TIMES.
       07 FILLER           PICTURE X(4).
       07 STD-1            PICTURE 999X9(5).
       07 FILLER           PICTURE X(5).
       07 STD-2            PICTURE X(25).
       07 FILLER           PICTURE X(5).
       07 SCR-3            PICTURE 9999.
       07 FILLER           PICTURE X(28).
01 L-7.
       05 FILLER                PICI'URE X(62) VALUE "0"
       05 FILLER PICTURE X(7) VALUE "MEAN ="
       05 IN-5 PICTURE 9(4).99.
       05 FILLER PICTURE X(60) VALUE SPACES.
```

```
PROCEDURE DIVISION.
PRO-l.
        OPEN INPUT CP-22,
        OUTPUT BM-7.
        MOVE SPACES TO M-REC, L-6.
        WRITE M-REC FROM HEADING.
PRO-2.
        MOVE ALL "*" TO IN-l.
        WRITE M-REC FROM L-l.
PRO-3.
        MOVE ALL "*" TO IN-2.
        WRITE M-REC FROM L-3.
PRO-4.
        WRITE M-REC FROM L-4.
PRO-S.
        PERFORM PRO-3.
PRO-6.
        WRITE M-REC FROM L-2.
PRO-7.
        READ CP-22 AT END GO TO PRO-8.
        ADD MONEY-M TO E.
        ADD 1 TO A.
        MOVE BI-REC TO REC-1 (A).
        GO TO PRO-7.
PRO-8.
        ADD 1 TO B.
        MOVE B TO C.
PRO-9.
        ADD 1 TO C.
        IF SCR-3 (B) GR SCR-3 (C)
        GO TO PRO-IO.
        MOVE REC-l(B) TO D.
        MOVE REC-I(C) TO REC-I(B).
        MOVE D TO REC-l(C).
PRO- 10.
        IF B EQ 9 GO TO PRO-11.
        IF C EQ 10 MOVE ZEROS TO C
                   GO TO PRO-8.
        GO TO PRO-9.
```

```
PRO-II.
        MOVE ZEROS TO B.
PRO- 12.
        ADD 1 TO B.
        MOVE B TO IN-3.
        MOVE REC-I(B) TO IN-4.
        WRITE M-REC FROM L-5.
PRO- 13.
        PERFORM PRO-12 9 TIMES.
PRO- 14.
        DIVIDE 10 INTO E GIVING IN-5
        WRITE M-REC FROM L-7.
PRO-15.
        CLOSE CP-22, BM-7.
        STOP RUN.
```

VER 4.2 FOR MSOS NONSEE
OF DIAGNOSTICS 0000
L SYMBOLS

```
        BCDBOXS
        TRANSMIT
        MGET
        MPUT
        MOPENF
        MCLOSEF
        EDITCOBL
        DIVIDE
        ABNORMAL
        SUBSCRIP1
        SUBSCRP2
        SUBSCRP3
```

INCOME REPORT

****

RANK.	STUDENT NO.	STUDENT NAME	INCOME
01	222-22182	APHICHAI UEMSATHIENPORN	8000
02	202-30594	WEERA APIBOON	7000
03	202-40399	WANPEN KANCHANASAI	3500
04	227-01947	SORASAK PONGAM	2100
05	192-30208	CHATCHAI TAN	2000
06	202-20677	AMORNRAT PLUNGKLANG	1700
07	212-45152	NATNAPA RUNGROADCHAIPORN	1500
08	202-45626	VIJOJ AROONREUNGSIRILERT	1250
09	232-22314	• NIVIT MTIKETI	800
10	231-08081	NUTT TOVICHIT	600

MEAN = 2845.00

สำหรับเครื่อง IBM คำสั่ง write มี 3 รูปแบบดังนี้

1) **WRITE** record-name **[FROM** data-name-l]

2) **WRITE** record-name **[FROM** data-name-l]

ในที่นี้ data-name-2 และ integer ต้องเป็น nonnegative (คือไม่มีค่าเป็นลบ) และมีมูลค่าน้อยกว่า 100 เสมอ

ถ้าใช้ option BEFORE ADVANCING เครื่องพิมพ์จะพิมพ์เรคคอร์ดก่อนแล้วจึงเว้นบรรทัด

ถ้าใช้ option AFTER ADVANCING เครื่องพิมพ์จะเว้นบรรทัดก่อนแล้วจึงพิมพ์เรคคอร์ด

ถ้าไม่มีการใช้ทั้ง option BEFORE และ option AFTER เครื่องพิมพ์จะเว้น 1 บรรทัดอัตโนมัติ

ตัวอย่าง

```
WRITE REPORT-LINE AFTER ADVANCING 3 LINES.
```

ตัวอย่าง

```
ENVIRONMENT DIVISION.
CONFIGURATION SECTION.
SOURCE-COMPUTER. IBM-370.
OBJECT-COMPUTER. IBM-370.
SPECIAL-NAMES.
    CO1 'IS TOP-OF-PAGE.
INPUT-OUTPUT SECTION.

PROCEDURE DIVISION.

    WRITE P-RECORD AFTER ADVANCING TOP-OF-PAGE
```

3) WRITE record-name [FROM data-name-l]

AFTER POSITIONING { data-name-2 / intger } **LINES**

ใน option AFTER POSITIONING, integer ต้องไม่มีเครื่องหมายและมีค่าใดค่าหนึ่งใน 0, 1, 2, หรือ 3

ส่วน data-name-2 ต้องนิยามมาแล้วด้วย picture X(1)

integer	Value of data-name-2	ความหมาย
1	blank	เว้น 1 บรรทัด
2	0	เว้น 2 บรรทัด
3	–	เว้น 3 บรรทัด
	+	พิมพ์ซ้ำบรรทัดเดิม
0	1	ขึ้นกระดาษแผ่นใหม่

หมายเหตุ ในแฟ้มข้อมูล 1 ชุด ทั้ง 3 รูปแบบนี้ จะใช้พร้อม ๆ กันไม่ได้ ต้องเลือกเอาแบบใดแบบหนึ่ง

คำสั่ง CLOSE

แฟ้มข้อมูลทุกชุดเมื่อทำการประมวลผลเสร็จเรียบร้อยแล้วจะต้องปิดให้หมด โดยมีรูปแบบดังนี้

CLOSE file-name-1 [file-name-2.....]

file name ในที่นี้ต้อง defined ใน FD entry เท่านั้น และจะต้องเปิดมาแล้ว คำสั่ง CLOSE หนึ่งคำสั่ง ใช้กับแฟ้มข้อมูล 1 ชุด แต่เราอาจจะใช้คำสั่ง CLOSE หนึ่งคำสั่งกับแฟ้มข้อมูลหลาย ๆ ชุดได้

ตัวอย่าง

```
CLOSE CARD-FILE, PRINT-FILE, TAPE-FILE.
```

คำสั่ง DISPLAY

คำสั่งนี้สั่งให้ย้ายข้อมูลจากเครื่องคอมพิวเตอร์ไปยัง console typewriter หรือ output device อื่น ๆ โดยมีรูปแบบดังนี้

DISPLAY { data-name-1 / literal-1 } [{ data-name-2 / literal-2 } ..] [**UPON** mnemonic-name]

1) ถ้าไม่มีคำ UPON อยู่ด้วย, ตัวอักขระเหล่านี้จะพิมพ์ออกมาทางเครื่องพิมพ์ดีดที่อยู่บนคอนโซลหรือออกมาทางเทอร์มินัล ในกรณีที่เราใช้เทอร์มินัล