บทที่ 3
ภาษาโคบอลและกระดาษเขียนโปรแกรม

ตัวอักขระ (character) เป็นหน่วยของข้อมูลที่เล็กที่สุดที่ใช้ในภาษาโคบอล ได้แก่ ตัวเลข (0-9) ตัวอักษร (A-Z) หรือสัญลักษณ์พิเศษต่าง ๆ ($ = + − * / , . เป็นต้น)

ตัวอักขระถ้านำมารวมเข้าด้วยกันก็เกิดเป็นฟิลด์ (fields) เช่น ฟิลด์ student number, student name, grade เป็นต้น

ถ้าฟิลด์นั้นมีมูลค่า (content) เป็นตัวเลขทั้งหมด เรียกว่า numeric field

ถ้าฟิลด์นั้นมีมูลค่าเป็นตัวอักษรทั้งหมด เรียก alphabetic field

ถ้าฟิลด์นั้นมีมูลค่าเป็นตัวอักขระหลายอย่างรวมกัน เราเรียกว่า alphameric field หรือ alphanumeric field

สำหรับคำสั่งให้คำนวณ (arithmetic statement) บวก ลบ คูณ หาร จะกระทำได้ (performed) เฉพาะฟิลด์ที่เป็นตัวเลขเท่านั้น

ถ้านำ data fields รวมเข้าด้วยกันก็จะเกิดเป็นเรคคอร์ด (record) ซึ่งเป็นหน่วยที่ใหญ่กว่าฟิลด์ ตัวอย่างเช่น ข้อมูลของนักศึกษาแต่ละคนในบัตรประวัตินักศึกษา (master card) หรือข้อมูลในบัตรกระบวนวิชา (course card) หนึ่งใบ ถือว่าเป็นหนึ่งเรคคอร์ด

จากเรคคอร์ดเหล่านี้ถ้านำมารวมเข้าด้วยกันก็เกิดเป็นแฟ้มข้อมูล (file) เช่น master-card file ของนักศึกษาที่ลงทะเบียนเรียนในภาค 1 ปีการศึกษา 2520 เป็นต้น

3.1 The character set

ภาษาโคบอลเกิดขึ้นจากกลุ่มของตัวอักขระทั้งหมด 51 ตัว โปรแกรมเมอร์เป็นคนเอามาประกอบเข้าด้วยกัน ภายใต้กฎเกณฑ์ที่กำหนดไว้ เพื่อให้เป็นชื่อ (names) และค่าจริง (values) ในโปรแกรม ซึ่งได้แก่

3.1.1 Numeric characters, ได้แก่ ตัวอักขระที่เป็นตัวเลขทั้งหมดคือ เลข 0 ถึงเลข 9 รวมทั้งหมด 10 ตัว

3.1.2 **Alphabetic characters,** ได้แก่ตัวอักษร ในภาษาอังกฤษ A ถึง Z รวมทั้งหมด 26 ตัว

3.1.3 **Special characters** มี 15 ตัว ได้แก่

	blank or space	$	dollar sign
+	plus sign	,	comma
-	minus sign or hyphen	.	period or decimal point
*	asterisk	"	quotation mark
/	slash	(	left parenthesis
=	equal sign	)	right parenthesis
<	less than	;	semi colon
>	greater than		

ตัวอักขระพิเศษ ทั้งหมดที่กล่าวข้างต้นแบ่งเป็น 4 กลุ่ม เพื่อใช้ในความหมายต่อไปนี้

a) ตัวอักขระที่ใช้แสดงความสัมพันธ์ (relation) ได้แก่

3	greater than
<	less than
=	equality

b) ตัวอักขระที่ใช้กำกับวรรคตอน (punctuation) ได้แก่

"	quotation mark	blank or space
(	left parenthesis	period
)	right parenthesis	comma
	semi colon	

c) ตัวอักขระที่ใช้เพื่อการบรรณาธิกรณ์ (editing) ได้แก่

$	dollar sign		actual decimal point
*	cheek protection	/	slash or stroke
,	comma	B	space
CR	credit	0	zero
DB	debit	+	plus
Z	zero suppression	-	minus

d) ตัวอักขระที่ใช้ในนิพจน์เลขคณิต (arithmetic expression) ได้แก่

+	addition	/	division
-	subtraction	**	exponentiation
*	multiplication		

3.2 Words

ได้แก่คำทุกคำที่ใช้บรรยาย การปฏิบัติงานในขั้นตอนต่าง ๆ แบ่งเป็น 3 ชนิดคือ

3.2.1 Programmer-specified words (หรือ programmer-supplied words หรือ user-defined word) หมายถึง ชื่อทั้งหมดที่โปรแกรมเมอร์ตั้งให้กับอีลิเม้นท์ต่าง ๆ ในโปรแกรม เช่น

ชื่อฟิลด์ (data names*)

ชื่อพารากราฟ (procedure names หรือ paragraph names), ชื่อเซคชัน (section name) (ชื่อพวกนี้อยู่ใน procedure division)

ชื่อมีเงื่อนไข (condition names)

ชื่อพิเศษ (special names)

ชื่อไฟล์ (file name), ชื่อโปรแกรม (program name) และชื่อเรคคอร์ด (record name)

3.2.2 กฎเกณฑ์ในการตั้งชื่อ (Rules for forming names)

ชื่อดังกล่าวข้างต้น เกิดขึ้นจากการรวมกันของกลุ่ม ตัวอักขระต่อไปนี้เท่านั้น

0,1,2 ..... , 9

A,B,C ..... , Z

– (hyphen)

1. ชื่อนั้นประกอบด้วยตัวอักขระตั้งแต่ 1 ตัวขึ้นไป มากที่สุด 30 ตัว สำหรับเครื่อง IBM ถ้าเป็นเครื่อง VAX ไม่เกิน 31 ตัว

2. เครื่องหมาย – (hyphen) นี้ จะอยู่ตรงตำแหน่งใดในชื่อนั้นก็ได้ แต่จะเป็นตัวแรกหรือตัวสุดท้ายของชื่อไม่ได้

---

* data-name แต่ละชื่อในภาษาโคบอล มีความหมายอย่างเดียวกับคำว่า ตัวแปร (variable) ในทางพีชคณิต คือ เป็นสัญลักษณ์ทั่วไปอย่างหนึ่ง หรือเป็นชื่อ ซึ่งอาจมีมูลค่า (content) เป็นไปได้หลายอย่าง มูลค่า ในที่นี้อาจเป็นตัวอักษร (alphabetic data) หรือตัวเลข (numeric data) หรือ ทั้งสองกลุ่มนี้รวมกัน (alphanumeric data) ก็ได้

ตัวอย่าง

QUANTITYONHAND จะอ่านง่ายขึ้นถ้าใช้เครื่องหมาย hyphen ช่วย เช่น QUANTITY-ON-HAND

แต่เครื่องหมาย – (hyphen) จะทำให้เกิดชื่อใหม่ขึ้นมา และนำไปใช้แทนชื่อเดิมไม่ได้ หมายความว่า ในโปรแกรมเดียวกันจะใช้ชื่อแรกบ้าง ชื่อหลังบ้าง ตามความสะดวกของโปรแกรมเมอร์ไม่ได้ ต้องเลือกเอาชื่อใดชื่อหนึ่ง

3. ชื่อฟิลด์, ชื่อมีเงื่อนไข, ชื่อพิเศษ, ชื่อไฟล์, ชื่อเรคคอร์ด, และชื่อโปรแกรมจะต้องมีตัวอักษรอยู่ด้วยอย่างน้อยที่สุดหนึ่งตัว

4. ชื่อพารากราฟหรือชื่อ section ใน procedure division อาจจะเป็นตัวเลขทั้งหมดได้ แต่ชื่อพารากราฟที่เป็นตัวเลขทั้งหมดนี้จะใช้ในความหมายเดียวกันก็ต่อเมื่อประกอบด้วยจำนวนเลขเท่ากัน และมีค่าเดียวกันเท่านั้น เช่น ชื่อพารากราฟ 00234 จะไม่เหมือนกับชื่อพารากราฟ 234

5. **ชื่อทุกชื่อที่ตั้งขึ้นมานี้ จะเหมือนกับ reserved word ไม่ได้** (หน้า 293-295) ตาราง reserved words อยู่ในภาคผนวกตอนท้ายของหนังสือเล่มนี้

6. ชื่อแต่ละชื่อต้องมีความหมายอย่างเดียว (unique) หมายความว่าชื่อนั้นจะปรากฏใน data division มากกว่าหนึ่งครั้งไม่ได้ ยกเว้นในกรณีมีการใช้ data-name qualification เท่านั้น ซึ่งจะได้กล่าวถึงภายหลัง

ตัวอย่าง ชื่อที่ถูกต้องตามกฎเกณฑ์การตั้งชื่อ (Valid names)

A14	TOYOTA-CORONA
HOURS	TOTAL4
A3770875	ENDING-PROGRAM-COBOL
GROSS-PAY	REPORT-LINE-12
NEW-MASTER-FILE	431-C
OLD-MASTER-RECORD	CUSTOMER-NAME
ACCOUNT-1234	ACCUMULATED-DEPRECIATION

ตัวอย่าง ชื่อที่คอมพิวเตอร์ไม่ยอมรับ (Invalid names)

ADD	เป็น reserved word
ACCT-REC* CONTRA	ผิดที่ *
X52.3	ผิดที่ .

1234	ถ้าเป็นชื่อพารากราฟหรือชื่อ section ใน division ที่ 4 จึงจะใช้ได้
BLOCK	ผิดเพราะเป็น reserved word
LOW-VALUE	ผิดเพราะเป็น reserved word
ACCUM DEPREC	ผิดเพราะมี blank
– ABC	ผิดเพราะเครื่องหมาย – อยู่หน้าสุด
SUM-1234.	ผิดเพราะมี.
END OF YEAR BALANCE	ผิดเพราะมีตัวอักขระ blank
12-56	ผิดเพราะไม่มีตัวอักษร

เนื่องจาก reserved words มีประมาณ 300 กว่าคำ และเป็นคำที่ไม่มีตัวเลขปนอยู่เลย ดังนั้นวิธีง่ายที่สุดในการหลีกเลี่ยงกฎนี้ เมื่อนักศึกษาจะตั้งชื่อต่าง ๆ เพื่อใช้ในโปรแกรม ควรจะมีตัวเลขหรือ – (hyphen) ประกอบในชื่อนั้นด้วย

3.2.3 Literals

เป็นค่าจริง (actual values) หรือเป็นค่าคงที่ (constants*) ซึ่งจะไม่มีการเปลี่ยนแปลงค่าอีกในโปรแกรม, โปรแกรมเมอร์ เป็นคนเขียนขึ้นเพื่อใช้ในโปรแกรมแต่ literal ไม่เหมือนกับ programmer-specified names ตรงที่ literals มีความหมายในตัวเอง ไม่ต้องกำหนดรายละเอียดในการจองเนื้อที่หน่วยความจำ (defined) ก่อน ก็นำมาเขียนในโปรแกรมได้เลย แบ่งเป็นสองชนิดคือ

numeric literal

non-numeric literal

กฎเกณฑ์ในการสร้าง numeric literals

1. ประกอบด้วยตัวเลขทั้งหมดไม่เกิน 12 หลัก (ไม่นับเครื่องหมายและไม่นับจุดทศนิยม) ถ้าต้องการจะใส่เลข 0 เพิ่มเติมเลขตัวนั้น ทางซ้ายมือหรือทางขวามือก็ได้แต่เมื่อนับรวมแล้วต้องไม่เกิน 19 ตัว (สำหรับเครื่อง VAX และเครื่อง IBM ไม่เกิน 18 ตัว)

2. อาจจะมีเครื่องหมาย + หรือ เครื่องหมาย – ได้, แต่เครื่องหมายนี้จะต้องเป็นตัวอักขระ, ตัวซ้ายมือสุดและมีได้เพียงหนึ่งตัว ในกรณีที่ไม่มีเครื่องหมายกำกับ จะถือว่าเลข

---

* ค่าคงที่ในภาษาโคบอลแบ่งออกเป็น 3 ชนิดคือ numeric literal, non-numeric literal และ figurative constant

นั้นเป็นค่าบวกและถ้ามีเครื่องหมาย + หรือ – กำกับ จะมีตัวอักขระ blank ตามหลังเครื่องหมายนี้ไม่ได้

เช่น + 258.2 ผิด

+258.2 ถูก

3. ถ้ามีจุดทศนิยมจะมีมากกว่าหนึ่งจุดไม่ได้ จุดทศนิยมจะอยู่ตรงตำแหน่งใดก็ได้ยกเว้นตัวขวามือสุด ถ้าเลขจำนวนนั้นไม่มีจุดทศนิยม เครื่องจะถือว่าเป็นเลขจำนวนเต็ม (จุดทศนิยมตัวนี้เครื่องถือว่าเป็น**จุดทศนิยมสมมติ**)

ตัวอย่าง numeric literals ที่ถูกต้อง

–12572.6	1970
+25675	10
1435.89	1
2.00007	112.540
–1977	00386

ตัวอย่าง numeric literals ที่เครื่องไม่ยอมรับ

786.12-

23,468

421.

$100.00

ตัวอย่าง จงขีดเส้นใต้ numeric literals ในประโยคต่อไปนี้

```
77 SUM-S PICTURE 99V9 VALUE 70.2.
MOVE NAME-IN (3) TO NAME-OUT (3).
ADD 1 TO TOTAL.
PERFORM ROUTINE-A 10 TIMES.
MULTIPLY SALES BY .0525 GIVING BONUS
IF NO-l IS GREATER THAN 15 GO TO LOST-ROUTINE,
DIVIDE AREA-AMOUNT INTO 3.1415 GIVING X.
```

กฎเกณฑ์ในการสร้าง non-numeric literals

1. ประกอบด้วยตัวอักขระอะไรก็ได้ใน COBOL set รวมทั้งตัวอักขระ blanks ยกเว้นเครื่องหมายคำพูด (quotation mark) เท่านั้น

2. จะต้องอยู่ภายในเครื่องหมายคำพูด แต่เครื่องหมายนี้ไม่นับรวมว่าเป็นส่วนหนึ่งของ non-numeric literal

3. มีความยาวไม่เกิน 120 ตัว (เครื่อง VAX ไม่เกิน 256 ตัว)

4. จะนำไปใช้ในคำสั่งที่มีการคำนวณ (arithmetic statement) ไม่ได้

ตัวอย่าง*

"BROWN"

"EXAMINE CLOCK NUMBER"

"PAGE 144 IS MISSING"

"–125.56"

"ITEM-DESCRIPTION"

"APRIL 28, 1977"

"ANY ERROR MASSAGE* T004"

แต่ "JOHN'S FATHER" ผิดตรงที่มี ' หน้า Sตามกฎข้อ 1.

ปกติ non-numeric literals จะเป็นข้อความเพื่อสื่อสารกันระหว่างตัวโปรแกรมกับพนักงานควบคุมเครื่องคอมพิวเตอร์ (operator) หรือให้พิมพ์หัวเรื่อง (heading) หรือให้ตรวจสอบข้อมูลหรือผลลัพธ์ในตำแหน่งต่าง ๆ ในโปรแกรม เช่น ถ้าส่วนนั้นของโปรแกรมให้คำตอบที่ไม่ถูกต้องก็ให้เครื่องพิมพ์ดีดบนคอนโซลพิมพ์ข้อความ (message) ออกมาให้พนักงานคุมเครื่องทราบเพื่อให้ทำ action อย่างใดอย่างหนึ่งต่อไป

ถ้าเราต้องการให้เครื่องหมายคำพูดเป็นส่วนหนึ่งของ non-numeric literal ก็สามารถกระทำได้ โดยใช้ figurative constant แทนเครื่องหมายนั้น

ตัวอย่าง

MOVE "JOHN QUOTE S FATHER" TO NAME.

ในที่นี้ QUOTE หมายถึงเครื่องหมาย quotation mark หนึ่งตัว (') ซึ่งทำให้มูลค่าของ NAME คือ JOHN'S FATHER

แบบฝึกหัด

1. คำต่าง ๆ ข้างล่างนี้เป็น data names มีบางตัวที่ใช้ไม่ได้ ให้บอกเหตุผลว่าทำไมจึงใช้ไม่ได้

   a) ITEM 1

   b) INPUT

---

* คำต่าง ๆ หรือข้อความทั้งหมดภายในเครื่องหมายคำพูดเปิด/ปิด ถือว่าเป็นหนึ่งคำ

c) RECORD-NO

d) FILE-A

e) FIELD 123

f) UNIT-14.8

g) ALPHA.-NAME

h) REC.

i) PRICE-$

j) 22A

k) -22

l) SPACE

m) END-OF-YEAR-BALANCE-DUE-ON-ACCOUNT

n) BALANCE-DUE-

2. คำต่อไปนี้เป็น numeric literals ที่ใช้ไม่ได้ ให้บอกเหตุผลว่าทำไมจึงใช้ไม่ได้

a) 1,234.56

b) 123456.

c) 123456-

d) $−1.2345

e) 123456.E08

3.2.4 COBOL reserved words

ได้แก่คำที่ใช้ได้เฉพาะในความหมายอย่างใดอย่างหนึ่งที่กำหนดให้เท่านั้น (specific use) ความหมายของคำประเภทนี้คอมไพเลอร์เข้าใจและยอมรับ มีอยู่ประมาณ 300 คำ reserved words อาจจะนำมาใช้เป็น non-numeric literals ได้ แต่จะมาใช้เป็น user-defined name เช่น ชื่อฟิลด์, ชื่อพารากราฟ, ชื่อมีเงื่อนไข, ชื่อไฟล์, ชื่อโปรแกรม หรือชื่อเรคคอร์ด ไม่ได้

Reserved words แบ่งออกเป็น 3 ชนิด คือ

optional word

key word

connectives

หมายเหตุ ให้ดูตารางของ reserved words ในภาคผนวก

1. Optional words

ใน source program อาจจะมีคำบางจำพวก ซึ่งประกอบอยู่ในคำสั่ง โดยมีวัตถุประสงค์เพื่อให้อ่านแล้วเหมือนประโยคภาษาอังกฤษทั่ว ๆ ไป โดยที่คำประเภทนี้แม้ว่าโปรแกรมเมอร์จะไม่เขียนไว้ความหมายของคำสั่งก็ไม่เปลี่ยนแปลง และคำพวกนี้ไม่จำเป็นต้องเอามาคอมไพล์ด้วย เราเรียกคำพวกนี้ว่า optional words

ตัวอย่าง นิพจน์ (expression) ทั้ง 4 บรรทัดข้างล่างนี้ถูกต้องและมีความหมายอย่างเดียวกันไม่ว่าจะมี optional words IS และ THAN หรือไม่ก็ตาม ในที่นี้ GREATER เป็น key word

A IS GREATER THAN B

A GREATER THAN B

A IS GREATER B

A GREATER B

2. Key words

Key words ใน COBOL set จำเป็นมากในการบอกความหมายของแต่ละส่วนของวลี (clause) หรือของประโยค (statement), key word ไม่เขียนไม่ได้ แบ่งออกเป็น 3 ชนิดคือ

verbs

'required words

functional words

Verbs เป็นคำในคำสั่งซึ่งบอกชนิดของ operation ที่จะสั่งให้กระทำ ได้แก่ verbs ที่ใช้ใน imperative statements ทั้งหมด เช่น ADD, SUBTRACT, MULTIPLY MOVE, OPEN, READ, WRITE,.....เป็นต้น

required words เป็นคำซึ่งต้องใช้ร่วมกับ verb เพื่อให้ความหมายของ verb หรือข้อความส่วนอื่นในประโยคได้ใจความสมบูรณ์ ตัวอย่างเช่น verb GO จะใช้โดด, ตามลำพังไม่ได้ ต้องมี required word TO เข้ามาประกอบด้วย จึงจะได้ความหมายครบถ้วน

functional words เป็นคำซึ่ง ไม่จำเป็นต้องกำหนดรูปแบบมาก่อน จะมีความหมายตามหน้าที่ของมันในประโยค

เช่น NEGATIVE หมายถึงค่าลบ

POSITIVE หมายถึงค่าบวก

3. Connectives เป็นคำที่บอกความต่อเนื่อง คำตั้งแต่สองคำขึ้นไป หรือประโยคตั้งแต่สองประโยคขึ้นไป แต่คำหรือตัวอักขระประเภทนี้อาจจะละเว้นไม่เขียนก็ได้ ได้แก่

OF และ IN	ใช้ใน qualified name
	(comma)
, AND	(comma space AND)
, OR	(comma space OR)
AND	
OR	
AND NOT	
OR NOT	

ถ้าต้องการให้อ่านง่ายขึ้นไปอีกให้ใช้ THEN แยกแต่ละคำสั่งภายใน sentence หนึ่ง ๆ ได้

Figurative constants เป็น reserved words ใช้แทนค่าคงที่ ฉะนั้นค่าคงที่ใด ๆ ก็ตาม ซึ่งชื่อเป็น reserved words จึงเรียกว่า figurative constants ชื่อเหล่านี้ทั้งรูปเอกพจน์ และพหูพจน์ มีความหมายอย่างเดียวกัน ใช้แทนกันได้ ได้แก่คำต่อไปนี้

ZERO ZEROS ZEROES	ทั้ง 3 รูปแบบนี้ มีความหมายเหมือนกัน หมายถึงมีค่าเป็นศูนย์ อาจจะถือเป็น numeric หรือ non-numeric literals ก็ได้ถ้าใช้ในประโยคที่มีการคำนวณ หมายถึง numeric มีค่าเป็นเลข 0, ถ้าเป็นอย่างอื่น ใช้แทนตัวเลข 0 จำนวนหนึ่งขึ้นอยู่กับขนาดของข้อมูล

ตัวอย่าง

```
MOVE ZEROS TO AMOUNT.
```

คำสั่งนี้จะทำให้เนื้อที่หน่วยความจำชื่อ AMOUNT ประกอบด้วยเลข 0 เท่ากับจำนวนตำแหน่งที่ได้มีการจองที่ไว้

SPACE SPACES	ทั้ง 2 รูปแบบนี้มีความหมายอย่างเดียวกัน ใช้แทนตัวอักขระ blanks จำนวนหนึ่ง ขึ้นอยู่กับขนาดของข้อมูล
QUOTE QUOTES	ใช้แทนเครื่องหมายคำพูด (") จำนวนหนึ่ง แต่คำว่า QUOTE จะนำไปใช้แทนสัญลักษณ์ " เพื่อบอกขอบเขตของ non-numeric literal ไม่ได้

ตัวอย่าง

non-numeric literal "ABC"

จะเขียนเป็น QUOTE ABC QUOTE ไม่ได้

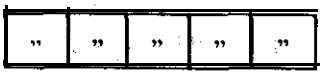
HIGH-VALUE HIGH-VALUES	หมายถึงค่าสูงสุดในลำดับตัวเลขของคอมพิวเตอร์เครื่องนั้น
LOW-VALUE LOW-VALUES	หมายถึงค่าต่ำสุด ในลำดับตัวเลขของคอมพิวเตอร์เครื่องนั้น

ALL any literal คำว่า ALL ตามหลังด้วย figurative constant หรือ non-numeric literal จะมีความหมายว่า แทน ตัวอักขระ ที่ประกอบขึ้นเป็น literal นั้นจำนวนหนึ่ง

ตัวอย่าง การใช้ figurative constants

1) MOVE QUOTES TO AREA-A.

สมมติว่าเนื้อที่ชื่อ AREA-A กำหนดไว้สำหรับใส่ตัวอักขระ 5 ตัว หลังจาก execute คำสั่งนี้แล้วจะได้ผลลัพธ์ ดังนี้



AREA-A

2) DISPLAY QUOTE "NAME" QUOTE.

ผลลัพธ์ พิมพ์ "NAME!'

3) MOVE SPACES TO TITLE.

เนื้อที่ชื่อ TITLE จะมีค่าเป็นตัวอักขระ blanks ทั้งหมด

4) MOVE ALL "4" TO COUNT-FILES.

ผลลัพธ์ให้ใส่เลข 4 ลงในทุก ๆ ตำแหน่งในเนื้อที่ชื่อ COUNT-FILES

5) MOVE ALL' ZEROS TO REGISTER. และคำสั่ง MOVE ZEROS TO REGISTER.

ทั้งสองคำสั่งนี้มีความหมายเหมือนกัน คือให้ใส่เลข 0 ลงในทุกตำแหน่งในเนื้อที่ชื่อ REGISTER

6) MOVE ALL "20 MAY 1977" TO DATE.

สมมติเนื้อที่ชื่อ DATE กำหนดไว้สำหรับใส่ตัวอักขระ 16 ตัว

หลังจากเครื่อง execute คำสั่งข้างต้นจะได้ผลลัพธ์ดังนี้

---

* ALL ตามด้วย figurative constant จะมีความหมายอย่างเดียวกับการใช้ figurative constant เพียงตัวเดียว

2	0		M	A	Y		I	Y	7	7	2	0		M	A

DATE

การใช้เครื่องหมายกำกับวรรคตอน (Punctuation)

โดยปกติภาษาโคบอล ไม่นิยมใช้เครื่องหมายกำกับวรรคตอน แต่ถ้าโปรแกรมเมอร์ต้องการเขียน มีกฎเกณฑ์ ดังนี้

1. การใช้เครื่องหมาย period (.) หรือ comma (,) หรือ semicolon (;) ต้องเว้นไว้หนึ่งที่ก่อนเขียนตัวอักขระตัวแรกของคำถัดไป

2. ตัวอักขระซึ่งอยู่หน้าเครื่องหมาย period (.) หรือ comma (,) หรือ semicolon (;) จะเป็น blank ไม่ได้

ตัวอย่าง

IDENTIFICATION DIVISION . (ผิด)

ADD ONE , TWO , THREE TO FOUR. (ผิด)

ADD ONE, TWO, THREE TO FOUR. (ถูก)

3. ตัวอักขระหลังวงเล็บเปิดต้องไม่ใช่ blank และตัวอักขระหน้าวงเล็บปิดต้องไม่ใช่ blank เช่นกัน

ตัวอย่าง

(A + B *C)

4. ระหว่างคำสองคำ หรือนิพจน์ ที่อยู่ในวงเล็บ สองชุด หรือ literal สองตัว จะต้องเว้นไว้อย่างน้อย หนึ่ง blank

5. **ตัวปฏิบัติการเลขคณิต** (arithmetic operator) **หรือเครื่องหมายเท่ากับจะต้องมี** blank หนึ่งตัวข้างหน้า และตามหลังด้วย blank อีกหนึ่งตัว สำหรับ unary operator ต้องอยู่หน้าวงเล็บเปิด

6. เครื่องหมาย comma ใช้คั่นระหว่างตัวถูกกระทำ หลาย ๆ ตัว (operands) ในคำสั่งนั้น

ตัวอย่าง

ADD A, B, C, D TO E.

7. เครื่องหมาย comma หรือ semicolon ใช้คั่นกลุ่มต่าง ๆ ของ clauses

ตัวอย่าง

RECORD CONTAINS 80 CHARACTERS, DATA RECORD IS CARD-REX

8. เครื่องหมาย **semicolon** ใช้คั่นกลุ่มของคำสั่ง **(statements)**

ตัวอย่าง

ADD A TO B; SUBTRACT B FROM C.

9. คำว่า THEN ใช้คั่นกลุ่มของคำสั่ง

ตัวอย่าง

ADD A TO B THEN SUBTRACT B FROM C.

10. ตัวอักขระ blank ที่อยู่ใน non-numeric literal ถือว่าเป็นส่วนหนึ่งของ literal และไม่ได้มีความหมายว่าแบ่งคำนั้น

ตัวอย่าง

"THIS IS A COBOL SOURCE PROGRAM"

ข้อความทั้งหมดในเครื่องหมายคำพูดถือว่าเป็น 1 คำประกอบด้วยตัวอักขระ 30 ตัว

11. สำหรับ division header, section header และ paragraph header ทั้งหมดนี้ต้องตามด้วย period

ตัวอย่าง

IDENTIFICATION DIVISION.

FILE SECTION.

BEGIN. (เป็นชื่อพารากราฟในโปรแกรมตัวอย่างหน้า 16)

FINISH. (เป็นชื่อพารากราฟในโปรแกรมตัวอย่างหน้า 16)

ใน data division, file description entry และ record description enry ทุกชุดต้องจบด้วย period

ใน procedure division, paragraphs และ sentences บางอย่าง (เช่น conditional sentences) จะต้องจบด้วย period

ใน environment และ identification divisions แต่ละพารากราฟต้องจบด้วย period

12. เครื่องหมายวงเล็บเปิด/ปิด (parentheses) มีการใช้ 2 อย่างคือ

a) subscripting เมื่อ subcript number หรือ data items ซึ่งเขียนอยู่ภายในเครื่องหมายวงเล็บตามหลังชื่อบอกชนิดของ item นั้น จะบอกลำดับอีลิเม้นท์ตัวที่ของชื่อนั้น (ในที่นี้หน้าวงเล็บเปิดต้องเป็น data-name)

ตัวอย่าง

NAME-IN (3)	หมายถึง อีลิเม้นท์ ชื่อ NAME-IN ตัวที่ 3
REC-F (1)	หมายถึง อีลิเม้นท์ ชื่อ REC-F ตัวที่ 1

TABLE (4,3) หมายถึง อีลิเม้นท์ตัวซึ่งอยู่แถวที่ 4 คอลัมน์ที่ 3 ของอะเรย์ 2 มิติชื่อ TABLE

b) เครื่องหมายวงเล็บที่ใช้ใน PICTURE clause หมายถึงมีตัวอักขระเหมือนกับตัวอักขระที่อยู่หน้าวงเล็บเปิด เท่ากับจำนวนเลขในวงเล็บนั้น

ตัวอย่าง

PICTURE Y(X)	หมายถึง PICTURE YYYYYYYY
PICTURE AA9(2)X	หมายถึง PICTURE AAYYX
PICTURE 9(3)X(5)	หมายถึง PICTURE YYYXXXXX
PICTURE A(S)	หมายถึง PICTURE AAAAA
PICTURE 9(3)V9(2)	หมายถึง PICTURE YYYVYY
PICTURE $(2), $(3).9(2)	หมายถึง PICTURE $$,$$$.99

แบบฝึกหัด

1. จงบอกว่าคำต่อไปนี้ เป็น numeric literal หรือ non-numeric literal หรือ figurative constant หรือไม่ใช่ทั้ง 3 ชนิดที่กล่าวข้างต้น

a) "DEPRECIATION SCHEDULE?'
b) "12%"
c) 237
d) "111"
e) INTEREST-DUE
f) 12.532
g) SPACES
h) "SPACES"
i) "QUOTE"
j) 459
k) Z E R O
I) HIGH-VALUE
m) TOTAL
n) SUM

2. จงใส่เครื่องหมาย √ หน้าข้อความที่เห็นว่าถูกต้อง
และใส่เครื่องหมาย × หน้าข้อความที่เห็นว่าผิด

a) All the capital letters of the English alphabet from A through Z are legal characters in COBOL

b) The numerals from 0 through 9 may be used in COBOL.

c) A COBOL data-name may contain a maximum of 20 characters.

d) A hyphen may be used in a COBOL data-name.

e) The COBOL statement

MULTIPLY HENS BY NUMBER-OF-EGGS GIVING CHICKS.

contains eight COBOL words !

f) A programmer may invent words to represent the quantities in his problem.

g) A COBOL data-name may not begin with a number.

h) -AREA- is a legal COBOL data-name.

i) A reserved word may be used as a data-name.

j) XADD is a legal COBOL data-name.

3. สำหรับข้อความแต่ละชุดทางซ้ายมือ จงหาคำตอบทางขวามือที่มีความหมายถูกต้องที่สุด แล้วใส่เลขหน้าช่องว่างทางซ้ายมือ, แต่ละคำตอบจะถูกใช้เพียงครั้งเดียวเท่านั้น

_____ a) data-name	1) DIVIDE A INTO B GIVING C.
__ b) reserved word	2) 3
_____ c) ADD statement	3) HO ! HO !
__ d) illegal character	4) ADD A, B BIVING C.
__ e) SUBTRACT statement	5) SUBTRACT C FROM B GIVING A.
__ f) illegal data-name	6) X
_____ g) C = A + B	7) (blank character)
_____ h) alphabetic character	8) HO-HO
_____ i) C = A/B	9) ?
_____ j) DIVIDE statement	10) ADD C, A GIVING B.
_____ k) punctuation character	11) INTO
_____ l) A = B − C	12) SUBTRACT B FROM C GIVING A.
__ m) numeric character	13) DIVIDE B INTO A GIVING C.

3.3 รูปแบบของกระดาษเขียนโปรแกรม (Format for program writing coding sheet)

ในการเขียนโปรแกรมด้วยภาษาใด ๆ ก็ตาม จำเป็นต้องเขียนลงในกระดาษตามแบบฟอร์ม ซึ่งแต่ละภาษากำหนดเอาไว้ ภาษาโคบอลก็เช่นเดียวกัน เราต้องเขียนโปรแกรมลงในกระดาษสำหรับเขียนโปรแกรมโดยเฉพาะ

กระดาษพิเศษที่ใช้เขียนโปรแกรมภาษาโคบอลนี้เรียกว่า COBOL coding form รูปแบบนี้เหมือนกับรูปแบบมาตรฐานของบัตร 80 คอลัมน์ คือ ในกระดาษหนึ่งบรรทัด แบ่งออกเป็น 80 คอลัมน์เช่นเดียวกัน และกระดาษเขียนโปรแกรมหนึ่งแผ่นจะมีประมาณ 20 บรรทัด

อย่างไรก็ตาม ทุกวันนี้เรามักจะส่งโปรแกรมเข้าเครื่องคอมพิวเตอร์ทางจอภาพ (a visual-display keyboard terminal) มากกว่า

กระดาษเขียนโปรแกรม

รายละเอียดของหัวกระดาษตอนบน

PROGRAM ให้ใส่ชื่อโปรแกรม

ROUTINE หมายถึงโปรแกรมย่อย ให้ใส่ชื่อโปรแกรมย่อย

NAME ใส่ชื่อโปรแกรมเมอร์

DATE ใส่ วัน เดือน ปี ที่เขียนโปรแกรม

PAGE OF ให้ใส่ตัวเลขเมื่อตัวเลขหลัง OF หมายถึงจำนวน coding sheet ทั้งหมดที่ใช้เขียนโปรแกรมนี้ และตัวเลขหลัง PAGE หมายถึงเลขบอกหน้าของ coding sheet แผ่นนั้น

ตัวอย่าง

PAGE 1 OF 3

หมายถึง coding sheet นี้เป็นแผ่นที่ 1 ของ coding sheet ทั้งหมด 3 แผ่นที่ใช้เขียนโปรแกรม

PAGE 3 OF 3

หมายถึง coding sheet แผ่นที่ 3 มีทั้งหมด 3 แผ่น นั่นคือ แผ่นนี้เป็นกระดาษใบสุดท้ายในการเขียนโปรแกรม

ตัวอย่าง ถ้าโปรแกรมของเราใช้ coding sheet ทั้งหมด 5 แผ่น แต่ละแผ่นจะเขียนตามลำดับดังนี้

แผ่นที่ 1 เขียน PAGE 1 OF 5

แผ่นที่ 2 เขียน PAGE 2 OF 5

แผ่นที่ 3 เขียน PAGE 3 OF 5

แผ่นที่ 4 เขียน PAGE 4 OF 5

แผ่นที่ 5 เขียน PAGE 5 OF 5

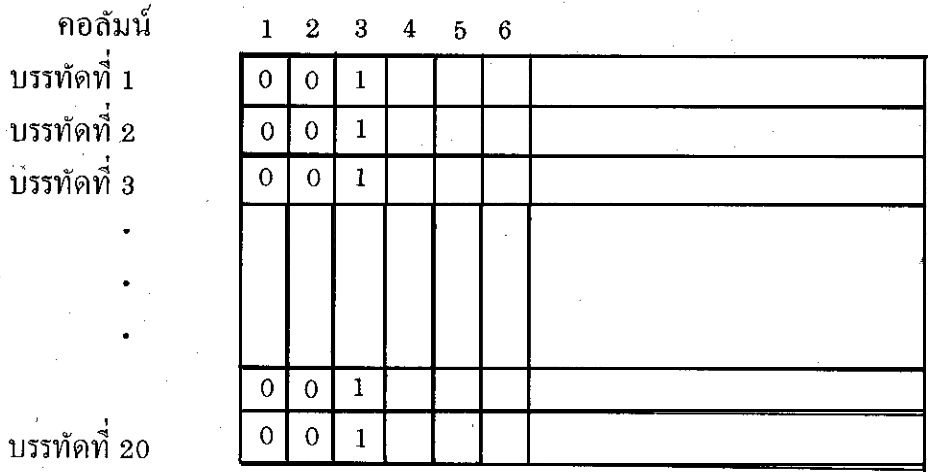
coding sheet จะมีจำนวนคอลัมน์เท่ากับจำนวนคอลัมน์ในบัตรเหมือนกัน คือ 80 คอลัมน์ แต่ละบรรทัดหมายถึงบัตรแต่ละใบ ใช้ในวัตถุประสงค์ต่าง ๆ กันดังนี้

1. Sequence number (คอลัมน์ 1-6)

แบ่งออกเป็น 2 ส่วน คือ

1.1 page number (คอลัมน์ 1–3) สำหรับใส่ตัวเลขบอกหน้าของ coding sheet แผ่นที่กำลังเขียนอยู่ มีประโยชน์ตรงที่ใช้ตรวจทาน (verified) บัตรที่เจาะโปรแกรมกับคำสั่งที่เขียนในกระดาษเขียนโปรแกรม

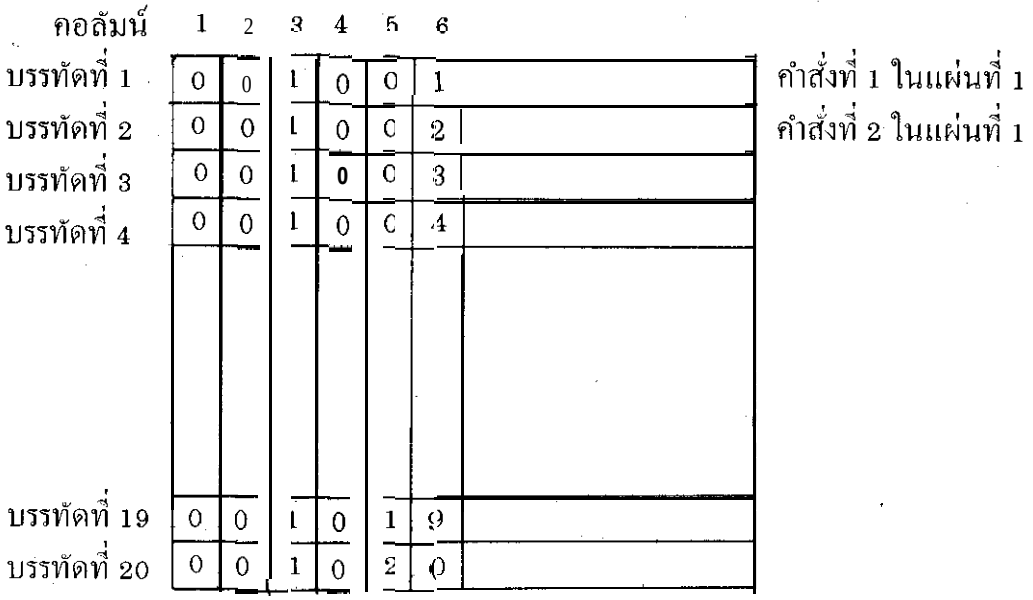
ตัวอย่าง



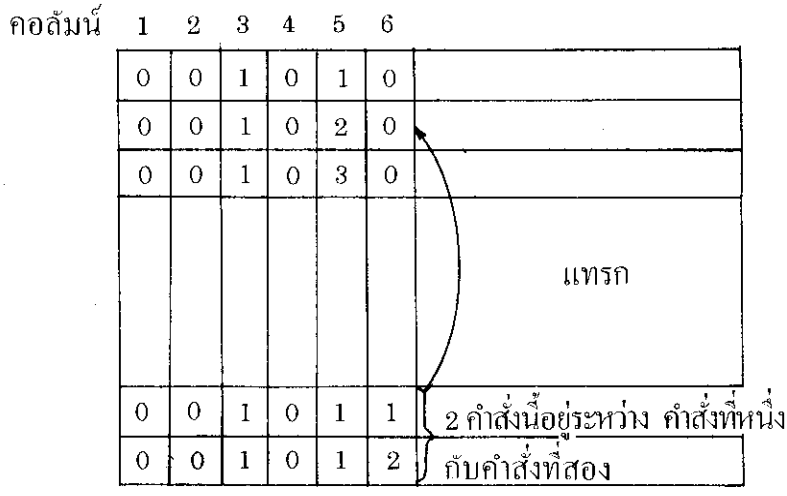
coding sheet แผ่นที่ 1

นั่นคือทำให้ทราบว่า บัตรที่เจาะโปรแกรมใบนั้นเขียนอยู่ในกระดาษเขียนโปรแกรมแผ่นที่เท่าไหร่

1.2 line number หรือ serial number (คอลัมน์ 4–6) สำหรับใส่เลขที่ของคำสั่งว่าเป็นคำสั่งที่เท่าไหร่



แต่นิยมเขียนตามรูปแบบข้างล่างนี้มากกว่าเพื่อความสะดวกเมื่อมีคำสั่งแทรกเพิ่มเติมภายหลัง คือ คอลัมน์ 6 ให้ใส่เลข 0 เมื่อมีคำสั่งแทรกในบรรทัดใดก็ตามก็ยังสามารถให้เลขซึ่งเรียงลำดับกันได้ในคอลัมน์ 6 โดยเพิ่มได้อีก 9 คำสั่ง



หมายเหตุ ข้อมูลหรือตัวเลขในคอลัมน์ 1–6 นั้น ไม่มีผลกระทบ (effect) ต่อคำสั่งในตัวโปรแกรมใด ๆ ทั้งสิ้น ฉะนั้นโปรแกรมเมอร์จะปล่อยให้ว่างไว้เฉย ๆ ก็ได้

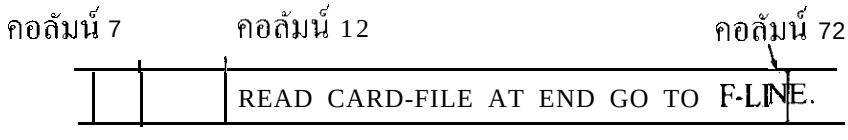
แต่ในกรณีที่มีตัวเลขอยู่ เครื่องคอมพิวเตอร์จะทำการตรวจสอบเลขเหล่านั้นว่าเรียงลำดับกันจากน้อยไปหามากถูกต้องหรือไม่ ถ้าเรียงลำดับไม่ถูกต้อง เครื่องจะพิมพ์ error message ออกมาให้ระหว่างที่มีการคอมไพล์โปรแกรม

2. Continuous indicator (คอลัมน์ 7)

คอลัมน์นี้สำหรับใส่เครื่องหมาย – (hyphen หรือ dash) ซึ่งเป็นตัวอักขระใช้บอกความต่อเนื่องของ word หรือ literal

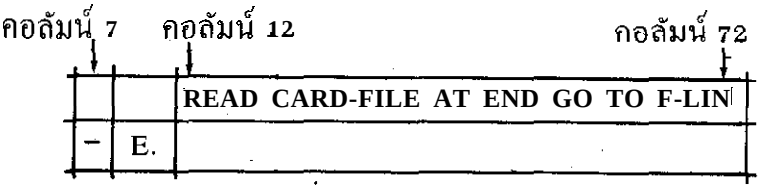
ในกรณีที่คำ (word) หรือ literal ไม่สามารถจบได้ในบรรทัดที่กำลังเขียนนั้น ในการต่อให้ขึ้นบรรทัดใหม่ โดยเริ่มที่คอลัมน์ 12 หรือทางขวามือของคอลัมน์ 12 แล้วใส่เครื่องหมาย hyphen (–) ที่คอลัมน์ 7 ของบรรทัดที่ต่อเพื่อบอกให้รู้ว่าเป็นบัตรต่อคำสั่ง

ตัวอย่าง

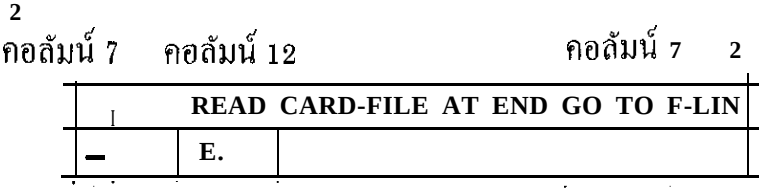


แต่คำสั่งใน source program จะเขียนได้ถึงคอลัมน์ 72 เท่านั้น เขียนเกินคอลัมน์ 72 ไม่ได้ ฉะนั้น จากตัวอย่างข้างต้น จึงต้องใช้สองบรรทัดในการเขียนหนึ่งคำสั่งดังกล่าว

แบบที่ 1



แบบที่ 2

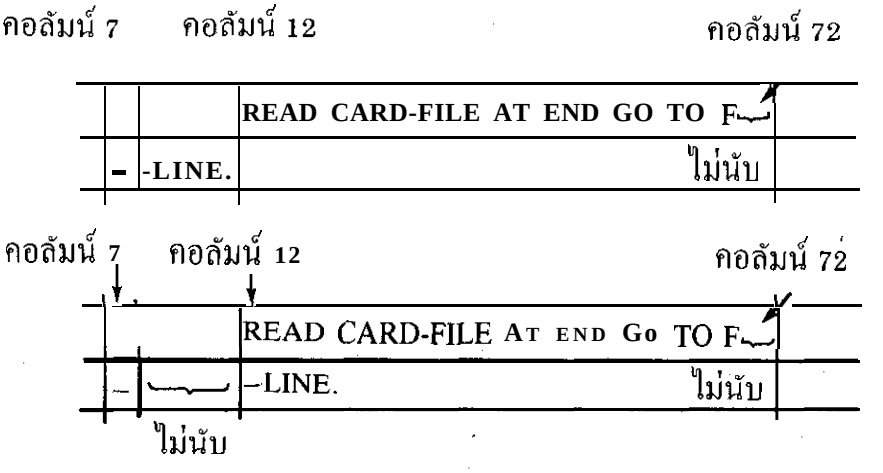


เราแยกประเภทของการต่อคำ (word) หรือ literal ดังนี้

a) ถ้าเป็น word หรือ numeric literal จำเป็นต้องเขียนใน 2 บรรทัด ให้ใส่เครื่องหมาย – (hyphen) ที่คอลัมน์ 7 ของบรรทัดที่สอง บอกให้รู้ว่าเป็นบัตรต่อเนื่องคำหรือต่อ literal หนึ่งตัว

ในกรณีเขียนไม่ถึงคอลัมน์ 72 เครื่องหมาย blanks ตอนท้ายที่เหลือของบรรทัดแรกจะไม่นับรวมว่าเป็นส่วนหนึ่งของ word หรือ numeric literal ซึ่งจะมาต่อกับตัวอักขระตัวแรกที่ไม่ใช่ blank ในบรรทัดที่สอง

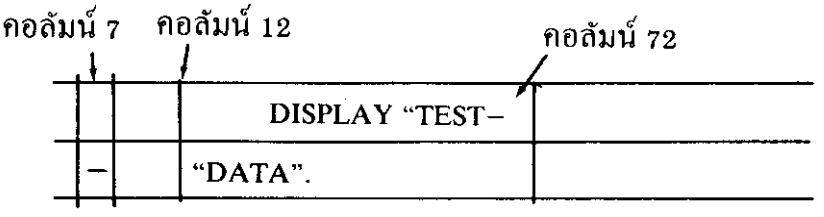
ตัวอย่าง จากตัวอย่างข้างต้น อาจเขียนได้อีก 2 แบบ คือ



b) ถ้าเป็น non-numeric literal เขียนแยกในสองบรรทัด นอกจากเครื่องหมาย – (hyphen) ในคอลัมน์ 7 แล้ว ยังต้องเพิ่มเครื่องหมายคำพูดอีกหนึ่งตัวในคอลัมน์ 12 หรือทางขวามือของคอลัมน์ 12 ในบรรทัดที่สอง, blanks ทั้งหมดตอนท้ายจนถึงคอลัมน์ 72 ของบรรทัดแรกให้ถือเป็นส่วนหนึ่งของ non-numeric literal ซึ่งจะต่อเนื่องทันที

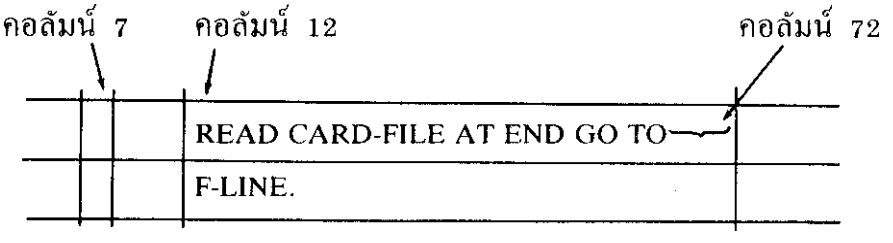
ตัวอย่าง

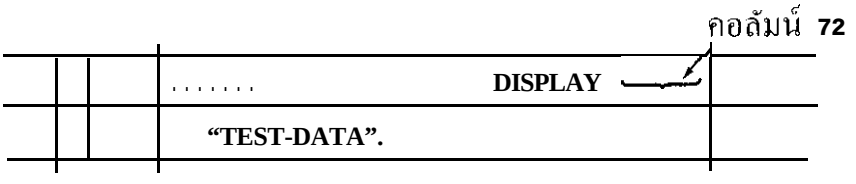
ให้เขียนต่อเนื่องคำดังนี้



หมายเหตุ

แต่แบบที่นิยมเขียนกันมากที่สุดคือ ถ้าคำสุดท้ายในบรรทัดแรกเขียนเกินคอลัมน์ 72 ให้ยกทั้งคำนั้นมาเขียนในบรรทัดที่สองเลย ที่เหลือตอนท้ายของบรรทัดแรกให้ว่างไว้ และไม่จำเป็นต้องมี hyphen ที่คอลัมน์ 7 จากตัวอย่างเดิม เขียนดังนี้





นอกจากนี้แล้วคอลัมน์ 7 ยังใช้สำหรับบอกให้ทราบว่าข้อความในบรรทัดนั้นเป็นคำอธิบาย (comment) โดยการใส่เครื่องหมาย * (asterisk) ในคอลัมน์นี้ ข้อความทั้งหมด

จะปรากฏใน listing ของ source program แต่ส่วนนี้เครื่องจะไม่คอมไพล์และไม่มีผลกระทบต่อ object program แต่อย่างใด มีไว้เพียงเพื่ออธิบายหรือเตือนความจำของโปรแกรมเมอร์ หรือเพื่อประโยชน์อื่น ๆ หรือเพื่อความสวยงามของตัวโปรแกรมเท่านั้น

3. **คอลัมน์ 8-72 ใช้สำหรับเขียนตัวโปรแกรม** (program instructions)

แบ่งออกเป็น 2 ส่วน คือ

a) margin A (คอลัมน์ 8-11)

ใช้สำหรับเขียน division name, section name, paragraph name, level number 01, level indicator (FD) ทั้งหมดนี้ต้องเขียนที่ มาร์จิน A คือเริ่มเขียนได้ตั้งแต่คอลัมน์ 8 เป็นต้นไป, ส่วน level number 77 จะเขียนที่มาร์จิน A หรือ มาร์จิน B ก็ได้ (สำหรับเครื่อง IBM, level number 77 ต้องเขียนที่ มาร์จิน A เสมอ)

b) margin B (คอลัมน์ 12-72)

ใช้สำหรับเขียนส่วนอื่น ๆ นอกไปจากส่วนที่กำหนดไว้ว่าต้องเขียนที่ มาร์จิน A ได้แก่ พวก paragraph content เป็นต้น

4. **Identification** (คอลัมน์ 73-80)

มีไว้สำหรับเขียน identification code หรือชื่อโปรแกรม แต่ข้อมูลส่วนนี้ไม่มีผลต่อการคำนวณแต่อย่างใด โปรแกรมเมอร์จะทิ้งว่างไว้ก็ได้ หรือจะใช้เขียนคอมเม้นท์อย่างอื่นนอกเหนือไปจากชื่อโปรแกรมก็ได้

เครื่องหมายและความหมายที่ใช้ในการกำหนดรูปแบบข้อความ หรือคำสั่งในภาษาโคบอล (COBOL format specifications/Meta-language element)

1. [--------] หมายถึง ข้อความหรือคำ (word) ที่อยู่ภายในเครื่องหมายวงเล็บนี้ จะเขียนหรือจะไม่เขียนก็ได้ ให้อยู่ในดุลยพินิจของโปรแกรมเมอร์
2. {--------} หมายถึง ข้อความหรือคำต่าง ๆ ที่อยู่ภายในเครื่องหมายวงเล็บนี้ให้เลือกเขียนเพียงหนึ่งอย่าง
3. COBOL word ทุกคำที่เขียนด้วยตัวพิมพ์ใหญ่ (capital/upper case letters) หมายถึง reserved word คือคำที่คอมไพเลอร์ทราบความหมายแล้ว และจะนำไปใช้ในความหมายอื่นไม่ได้
   a) ถ้าขีดเส้นใต้ที่คำนั้น หมายถึง key words จำเป็นต้องเขียนไว้และจะสะกดคำผิดไม่ได้เลย
   b) ถ้าไม่ได้ขีดเส้นใต้ที่คำนั้น หมายถึง optional words มีไว้เพื่อประกอบให้เป็นประโยค

ที่อ่านแล้วเข้าใจง่าย, โปรแกรมเมอร์จะเขียนหรือไม่เขียนก็ได้ ถ้าเขียนแล้วจะสะกดคำผิดไม่ได้เช่นกัน

4. COBOL word ทุกคำที่เขียนด้วยตัวพิมพ์เล็ก (small/lower case letters) หมายถึง ข้อความหรือคำที่โปรแกรมเมอร์ จะต้องเป็นคนกำหนดเองให้ถูกต้อง ตามกฎเกณฑ์และความหมายที่วางไว้ ตัวอย่างเช่น เลขบอกระดับ, ชื่อฟิลด์, ชื่อไฟล์, ชื่อเรคอร์ด เป็นต้น
5. เครื่องหมาย ..... หมายถึง อาจจะมีข้อความ หรือคำในคำสั่งนั้นตามมาอีกจำนวนหนึ่งก็ได้
6. สำหรับเครื่องหมาย +, –, <, > และ = เมื่อปรากฏอยู่ในรูปแบบต่าง ๆ จำเป็นต้องเขียนไว้ถึงแม้จะไม่ได้ขีดเส้นใต้ไว้ก็ตาม