# บทที่ 5 Stream-oriented transmission ,

ในบทนี้จะกล่าวถึงรายละเอียดของคำสั่งรับข้อมูลเข้าเครื่องคอมพิวเตอร์และคำสั่งนำเอาข้อมูลออกจากเครื่องคอมพิวเตอร์ในลักษณะ stream-oriented transmission ซึ่งแบ่งออกได้เป็น 3 ชนิด คือ list-directed, data-directed, และ edit-directed transmission

## 5.1 List–directed transmission

ลักษณะนี้อนุญาตให้โปรแกรมเมอร์ อ่านและพิมพ์ข้อมูลโดยไม่ต้องกำหนดรูปแบบให้กับ items ใน stream นั้น

Input: โดยทั่วไป, data items ใน stream เป็น character strings ในรูปของ มูลค่าคงที่อาจจะมีเครื่องหมายกำกับได้ หรือในรูปของ expressions ซึ่งแทน complex constants, การกำหนดตัวแปรให้กับ data items ให้กำหนดโดย data list โดยที่ items เหล่านั้นแยกกันด้วยเครื่องหมาย comma หนึ่งตัว และ/หรือ เครื่องหมาย blank ตั้งแต่หนึ่งตัวขึ้นไป

**ตัวอย่าง**

```
GET LIST (A,B,C,D);          1, 3, 5, -12
                      หรือ   1 3 5 -12
```

Output: มูลค่าของข้อมูลซึ่งจะถูกเคลื่อนย้ายกำหนดโดยตัวแปร, มูลค่าคงที่ หรือ expression ซึ่งแทน data item, data item แต่ละตัวที่อยู่ใน stream เป็น a character string มีลักษณะเหมือนกันกับ attribute ของตัวแปร, items เหล่านั้นแยกกันด้วยเครื่องหมาย blank ตั้งแต่หนึ่งตัวขึ้นไป เลขศูนย์นำหน้าข้อมูลที่เป็นตัวเลขจะถูกตัดทิ้ง ข้อมูลฐานสอง (Binary items) จะถูกแทนด้วยเลขฐานสิบ

สำหรับ Print file, data items จะถูกกำหนดตำแหน่งให้โดยอัตโนมัติ เป็นตำแหน่งที่ (คอลัมน์) 1, 25, 49, 73, 97, และ 121 ตามลำดับ

**ตัวอย่าง**

```
                คอลัมน์
PUT LIST (A,B,C,D);
```

| 1 | 25 | 49 | 73 | 97 |
|---|---|---|---|---|
| 1 | 3 | 5 | -12 | |

## 5.2 Data–directed transmission

ลักษณะนี้อนุญาตให้โปรแกรมเมอร์ เคลื่อนย้ายข้อมูลโดยกำหนดเอง (self–identifying data)

Input: data item แต่ละตัวใน stream อยู่ในรูปแบบของการกำหนดทั้งมูลค่าและตัวแปรที่จะให้มูลค่านั้น รูปแบบทั่วไป มูลค่านั้นจะเป็นมูลค่าคงที่ items เหล่านั้นเขียนแยกกันด้วยเครื่องหมาย comma หนึ่งตัว และ/หรือ เครื่องหมาย blank ตั้งแต่หนึ่งตัวขึ้นไป ให้ใส่เครื่องหมาย semi colon 1 ตัว เมื่อจบแต่ละกลุ่มของ items นั้นซึ่งจะรับเข้าไปในเครื่องคอมพิวเตอร์ โดยใช้คำสั่ง GET 1 คำสั่ง และอาจจะไม่ต้องมี data list ในคำสั่ง GET ก็ได้ (optional) เนื่องจากเครื่องหมาย semi colon จะเป็นตัวกำหนดจำนวน items ที่ได้จาก stream แล้ว

**ตัวอย่าง**

```
GET DATA (A,B,C,D);
```

หรือ

```
GET DATA;
```

หรือ

```
A=1, B=3, C=5, D=-12;
```

หรือ

```
A=1 B=3 C=5 D=-12;
```

Output: มูลค่าของข้อมูลที่จะถูกนำออกมาจากเครื่องอาจจะมี data list หรือไม่มีก็ได้, data item แต่ละตัวใน stream มีรูปแบบของคำสั่ง assignment ไม่มีเครื่องหมาย semi colon, items เหล่านั้นแยกกันด้วยเครื่องหมาย blank ตั้งแต่ 1 ตัวขึ้นไป, item ตัวสุดท้ายซึ่งจะถูกเคลื่อนย้ายโดยคำสั่ง PUT จะต้องตามด้วยเครื่องหมาย semi colon 1 ตัว เลขศูนย์ที่นำหน้าข้อมูลที่เป็นตัวเลขจะถูกตัดทิ้ง, การแทน character ของมูลค่าแต่ละตัวมีลักษณะเหมือนกับ attributes ของตัวแปร ยกเว้นข้อมูลฐานสอง ปรากฏในลักษณะฐานสิบ

ถ้าไม่มี data list เครื่องจะถือว่า ตัวแปรทั้งหมดภายใน block ประกอบด้วยคำสั่งและมีลักษณะเป็น data directed output

สำหรับ print file เครื่องจะกำหนดตำแหน่งให้กับ data items อัตโนมัติเช่นเดียวกับชนิด List–directed transmission

**ตัวอย่าง**

```
PUT DATA (A,B,C,D);
```

| คอลัมน์ | 1 | 25 | 49 | 73 |
|---|---|---|---|---|
| | A=1 | B=3 | C=5 | D=-12; |

## 5.3 Edit–directed transmission

ลักษณะนี้อนุญาตให้โปรแกรมเมอร์ กำหนดมูลค่าข้อมูลให้กับตัวแปร หรือกำหนดข้อมูลซึ่งจะมีการเคลื่อนย้าย และกำหนดรูปแบบของ item แต่ละตัวซึ่งจะออกทางตัวกลางบันทึกข้อมูลภายนอก (external medium)

Input: ข้อมูลใน stream เป็น string ของ character ที่ต่อเนื่องกัน ไม่มีการแยก data item ตัวแปรที่กำหนดให้กับข้อมูล กำหนดโดย data list

Format items ใน format list กำหนดจำนวนของ character ที่ประกอบเป็นมูลค่าของตัวแปรแต่ละตัว และบอกลักษณะเฉพาะของข้อมูล (ตัวอย่างเช่น บอกตำแหน่งของทศนิยม)

Output: มูลค่าของข้อมูล ถูกเคลื่อนย้ายโดย data list ส่วนรูปแบบของข้อมูลใน stream ถูกกำหนดโดย format list

## Data Transmission Statements

การเคลื่อนย้ายข้อมูลชนิดนี้ (stream-oriented transmission) ใช้เฉพาะคำสั่งอ่านข้อมูลหนึ่งคำสั่ง (GET) และคำสั่งนำข้อมูลออกหนึ่งคำสั่ง (PUT) เท่านั้น

คำสั่ง GET หนึ่งคำสั่ง เอาชุดของ data items ถัดไปจาก stream และคำสั่ง PUT หนึ่งคำสั่ง ให้ชุดของ data items ที่กำหนด 1 ชุดไปไว้ใน stream, ตัวแปรที่กำหนดให้กับ data items และตัวแปร หรือ expressions ซึ่งจะถูกเคลื่อนย้าย จะกำหนดใน data list 1 ชุด ในคำสั่ง GET หรือคำสั่ง PUT

ในคำสั่งนั้นอาจจะมี options ซึ่งกำหนดมูลค่าเริ่มต้น หรือมูลค่าสุดท้ายของข้อมูล หรือชี้ให้เห็นว่ามันจะปรากฏใน stream ซ้ำกันเช่นเดียวกับข้อมูลข้างหน้า รูปแบบต่อไปนี้ เป็นการรวบรวมข้อสรุปของคำสั่ง stream-oriented data transmission และ options ของมัน

**Stream input:**

```
GET [{FILE (file-expression)} | {STRING (character-string-expression)}]
    [data-specification]
    [COPY [(file-expression)]]
    [SKIP [(expression)];]
```

**หมายเหตุ**

option COPY และ SKIP จะใช้ร่วมกับ option STRING ในคำสั่ง GET ไม่ได้

**Stream output:**

```
PUT [{FILE(file-expression)} | {STRING(character-string-variable)}]
    [data-specification]
    [SKIP [(expression)]];
```

**หมายเหตุ**

option SKIP จะใช้ร่วมกับ option STRING ในคำสั่ง PUT ไม่ได้

**Stream output print:**

PUT [FILE (file−expression)]

[data−specification]

[PAGE [LINE (expression)]
SKIP [(expression)]
LINE (expression);]

options เหล่านี้จะเรียงลำดับอันไหนก่อน อันไหนหลังก็ได้ ส่วน data−specification จะต้องอยู่ในรูปแบบใดรูปแบบหนึ่งต่อไปนี้

[LIST] (data−list)

DATA [ (data−list) ]

EDIT { (data−list) (format−list) }...

SNAP

FLOW

ALL [(character−string−expression)]

ถ้าหากว่าคำสั่ง GET หรือคำสั่ง PUT ซึ่งมี data list นั้นไม่มี keyword ตัวใดตัวหนึ่ง ใน LIST, DATA, หรือ EDIT นำหน้าเครื่อง จะถือราวกับว่ามี LIST นำหน้า ในคำสั่งเช่นนี้ data list ต้องตามหลัง keyword GET หรือ PUT ทันที ส่วน options อื่น ๆ ที่กำหนดขึ้นให้ตามหลัง data list อีกทีหนึ่ง

ส่วน options SNAP, FLOW และ ALL ใน data specification นั้น เป็นตัวนำข้อเท็จจริงเกี่ยวกับโปรแกรมให้เข้าไปอยู่ใน stream, options เหล่านี้จะใช้เฉพาะในคำสั่ง PUT เท่านั้น และต้องเป็นคำสั่ง PUT ที่ถูกประเมินผลด้วย PL/1 checkout compiler เท่านั้น ถ้าเป็นคำสั่ง PUT ที่ถูกประเมินผลด้วย PL/1 optimizing compiler เครื่องจะถือว่า options เหล่านี้ ถูกตรวจสอบแล้วเป็น snytax errors และไม่สนใจ (ignored)

data specification จะไม่มีก็ได้ถ้าหากว่ามี control option ตัวใดตัวหนึ่ง ใน PAGE, SKIP, หรือ LINE ปรากฏอยู่

ใน format lists อาจจะใช้ format item ตัวใดตัวหนึ่งต่อไปนี้

| | |
|---|---|
| A, B, C, E, F, P, R, X | ซึ่งใช้ใน STREAM หนึ่ง ๆ หรือ STRING option |
| SKIP [(w)], COLUMN (w) | ซึ่งอาจจะใช้กับ STREAM file หนึ่ง ๆ |
| PAGE, LINE (w) | ซึ่งอาจจะใช้กับ STREAM output print files |

## Options Of Transmission Statements

**FILE and STRING option**

option FIIE เป็นการกำหนดว่า จะเกิด operation กับ file นั้น, ส่วน option STRING ในคำสั่ง GET และคำสั่ง PUT ใช้สำหรับเคลื่อนย้ายข้อมูลระหว่างตำแหน่งของหน่วยความจำภายในเครื่องมากกว่าที่จะเป็นระหว่าง หน่วยความจำภายใน กับหน่วยความจำภายนอกเครื่อง ถ้าในคำสั่ง GET ไม่มีทั้ง option FILE และไม่มี option STRING ด้วยเครื่องจะถือว่าเป็น standard–input file SYSIN และถ้าไม่มี options ทั้งสอง ในคำสั่ง PUT เครื่องจะถือว่าเป็น standard output file SYSPRINT

**COPY option**

option นี้จะปรากฏเฉพาะในคำสั่ง GET FILE เท่านั้น เป็นการกำหนด stream ซึ่งจะให้พิมพ์ นั่นคือ อ่านข้อมูลจาก file แล้วเอาไปไว้ใน file ที่มีชื่ออยู่ใน COPY specification, ถ้าไม่กำหนดชื่อ file เครื่องจะถือว่าเป็น standard output file SYSPRINT

**ตัวอย่าง**

```
GET FILE (SYSIN) DATA(A,B,C) COPY (ABC);
```

คำสั่งข้างต้นนี้ไม่เพียงแต่เคลื่อนย้ายมูลค่าซึ่งกำหนดให้ตัวแปร A, B และ C ใน input stream เท่านั้น แต่ยังพิมพ์มูลค่าเหล่านี้ด้วยเหมือนกับที่ปรากฏใน input stream ลงใน file ABC ถ้าไม่กำหนดชื่อ file เครื่องจะ write on SYSPRINT file

**SKIP option**

option นี้กำหนด บรรทัดใหม่ (หรือ record ใหม่) ภายใน data set, expression ในเครื่องหมายวงเล็บ จะถูกเปลี่ยนรูปเป็นเลขจำนวนเต็ม w

data set นั้นจะถูก กำหนดตำแหน่งให้ไปเริ่มที่บรรทัด w (record w) จากบรรทัดปัจจุบัน (record)

สำหรับ non–PRINT files, ถ้าไม่มี expression หรือถ้า w ไม่มากกว่าศูนย์, เครื่องจะถือว่ามีมูลค่าเท่ากับ 1

สำหรับ PRINT file ถ้า w มีมูลค่าน้อยกว่าหรือเท่ากับศูนย์ ผลลัพธ์ก็คือเครื่องจะเลื่อนกลับเพื่อพิมพ์บรรทัดเดิม (current line) แต่ถ้าไม่มี expression นี้ เครื่องจะถือว่ามีมูลค่าเท่ากับ 1

option SKIP ให้ผลลัพธ์ก่อนที่จะมีการเคลื่อนย้ายข้อมูลใด ๆ ซึ่งกำหนดใน data specification ถึงแม้ว่าจะเขียนทีหลัง data specification ก็ตาม

**ตัวอย่าง**

```
PUT LIST (P,Q,R,) SKIP (2);
```

คำสั่งข้างต้นนี้ เครื่องจะพิมพ์มูลค่าของ ตัวแปร P, Q, และ R ออกมาทาง standard output file SYSPRINT ลงในบรรทัดที่สองถัดจากบรรทัดปัจจุบัน

แต่ถ้าให้พิมพ์ออกทางเทอร์มินัล (terminal), SKIP (w) ถ้า w มีมูลค่ามากกว่า 3 จะมีความหมายอย่างเดียวกับ SKIP (3) หมายความว่าเครื่องจะข้ามให้ไม่เกิน 3 บรรทัดเท่านั้น

**PAGE option**

option นี้กำหนดได้เฉพาะ PRINT file ผลก็คือเครื่องจะขึ้นกระดาษแผ่นใหม่ ให้กับกลุ่มข้อมูลนั้นก่อนที่จะเคลื่อนย้ายมูลค่าใด ๆ ซึ่งกำหนดใน data specification (ถ้ามี) ถึงแม้ว่า option นี้จะปรากฏทีหลัง data specification ก็ตาม

แต่ถ้าให้พิมพ์ออกมาทาง เทอร์มินัล option นี้จะข้ามให้ 3 บรรทัดเท่านั้น

**LINE option**

option นี้กำหนดได้เฉพาะ PRINT file ผลก็คือ ใส่ blank file และบรรทัดถัดไปก็คือบรรทัดที่ w จาก กระดาษแผ่นนั้น เมื่อ w เป็นมูลค่าของ expression ในวงเล็บซึ่งเครื่องจะเปลี่ยนรูปให้เป็นเลขจำนวนเต็ม, option นี้จะให้ผลก่อนที่เครื่องจะเคลื่อนย้ายมูลค่าใด ๆ ที่กำหนดใน data specification (ถ้ามี) ถึงแม้ว่า option นี้จะปรากฏทีหลัง data specification ก็ตาม

ถ้าทั้ง PAGE option และ LINE option ปรากฏในคำสั่งเดียวกัน เครื่องจะทำ PAGE option ให้ก่อน

**ตัวอย่าง**

```
PUT FILE (LIST) DATA (P,Q,R,) LINE (34) PAGE;
```

คำสั่งข้างต้นนี้ มูลค่าของตัวแปร P, Q, และ R พิมพ์ในรูปแบบของ data-directed บนกระดาษแผ่นใหม่บรรทัดที่ 34

แต่ถ้าให้พิมพ์ออกมาทาง เทอร์มินัล ตามตัวอย่างข้างต้นนี้ เครื่องจะพิมพ์ให้บรรทัด 3

**Data specification**

data specification ซึ่งอยู่ในคำสั่ง GET และคำสั่ง PUT ให้กำหนดมูลค่าซึ่งจะให้เคลื่อนย้าย

**DATA lists**

ใน List-directed และ edit-directed, data specification ต้องมี data list เพื่อกำหนด data item ซึ่งจะทำการเคลื่อนย้าย ส่วนใน data-directed, data-specification อาจจะไม่ต้องมี data list ก็ได้

โดยมีรูปแบบดังนี้

(data–list)

เมื่อ data–list หมายถึง

element [, element] .....

**กฎเกณฑ์ของ syntax**

ลักษณะของอีลิเม้นต์เหล่านี้ขึ้นอยู่กับว่า data list นั้นใช้เป็น input หรือ output โดยมีกฎเกณฑ์ต่อไปนี้

1. ใน input, data-list element แต่ละตัวใน edit-directed และ list-directed transmission อาจจะเป็นตัวหนึ่งตัวใดใน an element, array หรือ structure variable, a pseudovariable หรือ a repetitive specification (เหมือนกับ a repetitive specification ใน do group) เกี่ยวกับอีลิเม้นต์ตัวใดตัวหนึ่งของอีลิเม้นต์เหล่านี้

สำหรับ a data-directed data specification, data-list element แต่ละตัวอาจจะเป็น an element, array หรือ structure variable แต่ทั้งนี้ชื่อทั้งหมดใน a data–directed data list จะมี subscripted, locator–qualified หรือ iSUB–defined ไม่ได้แต่เป็น qualified (นั่นคือ structure member), simple–defined, หรือ string–overlay–defined names ได้

2. ใน output, data–list element แต่ละตัวใน edit–directed และ list–directed data specification อาจจะเป็นตัวใดตัวหนึ่งใน an element expression, an array expression, a structure expression หรือ a repetitive specification เกี่ยวกับอีลิเม้นต์ตัวใดตัวหนึ่งในอีลิเม้นต์เหล่านี้

สำหรับ a data–directed data specification, data–list element แต่ละตัวอาจจะเป็น an element, array หรือ structure variable, หรือ a repetitive specification เกี่ยวกับอีลิเม้นต์ตัวใดตัวหนึ่งของอีลิเม้นต์เหล่านี้ แต่จะเป็น locator–qualified หรือ iSUB–defined ไม่ได้ แต่เป็น qualified (นั่นคือ สมาชิกตัวหนึ่งของ a structure), หรือ simple–defined หรือ string–overlay–defined หรือ subscript ก็ได้

3. อีลิเม้นต์ของ data list แต่ละชุดอาจจะเป็น

| | | |
|---|---|---|
| Input : | Problem data: | Arithmetic |
| | | String |
| Output : | Problem data: | Arithmetic |
| | | String |

Problem control data:

Area

Entry

Event

File

Label

Offset

Pointer

Task

แต่จะเป็น entry และ label constants ไม่ได้

data list ซึ่งกำหนด problem control data นั้นจะใช้ได้เฉพาะในคำสั่ง PUT DATA หรือ PUT LIST ซึ่งประมวลผลด้วย checkout compiler หรือคำสั่ง PUT DATA ซึ่งประมวลผลด้วย optimizing compiler เท่านั้น, ในกรณีหลังนี้ ชื่อของตัวแปรจะถูกเคลื่อนย้าย ไม่ใช่เคลื่อนย้ายมูลค่าของมัน

4. data list ต้องอยู่ภายในเครื่องหมายวงเล็บ

5. ใน output, data list จะประกอบด้วย data items ที่เป็น expressions ได้ไม่เกิน 60 ตัว

**Repetitive Specification**

มีรูปแบบทั่วไปดังนี้

(element[,element]...DO { variable / pseudovariable } = specification...)

specification หนึ่ง ๆ มีรูปแบบดังนี้

expression 1 [ TO expression 2 [BY expression 3] / BY expression 3 [TO expression 2] / REPEAT expression 6 ] [WHILE (expression 4)] [UNTIL (expression 5)]

เมื่อ WHILE option และ UNTIL option จะเรียงอันดับไหนก่อน อันไหนหลังก็ได้

**กฎเกณฑ์ของ syntax**

1. อีลิเม้นต์แต่ละตัวใน element list ของ repetitive specification อาจจะเป็นอีลิเม้นต์ตัวใดตัวหนึ่งในอีลิเม้นต์ของ data-list element ที่กล่าวมาแล้วข้างต้น

2. expression ใน specification มีความหมายอย่างเดียวกับในคำสั่ง DO กล่าวคือ

a) expression แต่ละชุดใน specification ต้องเป็น an element expression

b) ใน specification, expression 1 แทนมูลค่าเริ่มต้นของตัวแปรควบคุมหรือตัวแปรเทียม expression 3 แทนมูลค่าเพิ่มซึ่งจะเอาไปบวกกับตัวแปรควบคุมภายหลังจากแต่ละ repetition ของ data-list elements ใน repetitive specification expression 2 แทนมูลค่าสุดท้ายของตัวแปรควบคุม, expression 6 เป็น an expression ซึ่งจะถูกประเมินผล และกำหนดให้เป็นมูลค่าของตัวแปรควบคุมภายหลังจากแต่ละ repetition, expression 4 และ expression 5 แทนเงื่อนไขต่อไป เพื่อควบคุมจำนวนครั้งที่จะให้เครื่องทำซ้ำ ๆ กัน, ความหมายของ specification เหมือนกับ specification ในคำสั่ง DO เมื่อเครื่องทำ specification สุดท้ายเสร็จเรียบร้อยแล้ว control จะถูกส่งไปยังอีลิเม้นต์ตัวถัดไปใน data list

3. แต่ละ repetitive specification ต้องอยู่ในวงเล็บเปิด/ปิด เหมือนที่แสดงในรูปแบบทั่วไป มีข้อสังเกตว่าถ้า repetitive specification 1 ชุดนั้น เป็นเพียงอีลิเม้นต์ชุดเดียวใน data list ต้องมีเครื่องหมายวงเล็บข้างนอกซ้อนกัน 2 ชุด เพราะว่า data list ต้องมีเครื่องหมายวงเล็บ 1 ชุด และ repetitive specification ก็จะต้องมีเครื่องหมายวงเล็บอีก 1 ชุดต่างหาก

4. จากรูปแสดงรูปแบบของ specification ส่วนของ repetitive specification หนึ่ง ๆ อาจจะทำซ้ำหลายครั้ง ดังตัวอย่างข้างล่างนี้

```
DO I = 1 TO 4, 6 TO 10;
```

Repetitive specification อาจจะมีซ้อนกันได้ (nested) นั่นคือ อีลิเม้นต์หนึ่งตัวของ repetitive specification โดยตัวมันเองก็เป็น a repetitive specification ด้วย, DO แต่ละชุดต้องจบด้วยเครื่องหมายวงเล็บปิดขวามือ (ซึ่งคู่กับวงเล็บเปิดซ้ายมือ)

**ตัวอย่าง**

```
GET LIST (((A(I,J) DO I = 1 TO 2) DO J = 3 TO 4));
```

คำสั่งข้างต้นนี้มีเครื่องหมายวงเล็บ 3 ชุด และชุดในสุด อีก 1 ชุด เป็นตัวกำหนดขอบเขตของ subscript

| | | |
|---|---|---|
| | วงเล็บชุดนอกสุด | เป็นชุดที่กำหนดdata list |
| | วงเล็บชุดที่สอง | เป็นชุดที่กำหนด repetitive specification อันนอก |
| ส่วน | วงเล็บชุดที่สาม | เป็นชุดที่กำหนด repetitive specification ชุดใน |

คำสั่งดังกล่าวข้างต้นจึงมีความหมายเหมือนกับ nested do-group ต่อไปนี้

```
DO J = 3 TO 4;
    DO I = 1 TO 2;
    GET LIST (A(I,J) );
    END;
END;
```

ซึ่งจะกำหนดมูลค่าของอีลิเม้นต์ของ array A ตามลำดับ คือ A(1,3),A,(2,3),A(1,4),A(2,4)

สำหรับ optimizing compiler ระดับของ nesting กำหนดให้มากที่สุดเพียง 50 ชุด ส่วน check-out compiler กำหนดได้ไม่มีข้อจำกัด

**หมายเหตุ**

ถึงแม้ว่า keyword DO จะใช้ใน repetitive specification แต่ไม่ต้องใช้คำสั่ง END คู่กัน

**Transmission of Data–List Elements**

ถ้า data–list element แต่ละตัวเป็น an array variable อีลิเม้นต์ของ array จะถูกเคลื่อนย้ายในลักษณะของ row–major order หมายความว่า subscript ตัวขวามือสุดของ array นั้นจะถูกเปลี่ยนก่อน

ถ้า data–list element แต่ละตัวเป็นตัวแปรโครงสร้าง (a structure variable) อีลิเม้นต์ของโครงสร้างจะถูกเคลื่อนย้ายในลักษณะที่ปรากฏในการ declare โครงสร้างนั้น

**ตัวอย่าง**

```
        DCL 1 A(10),2B,2C;
คำสั่ง  PUT FILE (X) LIST (A);
        ผลลัพธ์ใน output จะปรากฏดังนี้
        A.B(1) A.C(1) A.B(2) A.C(2) A.B(3) A.C(3)...
ถ้า     DCL 1 A, 2B(10), 2C(10);
        ใช้คำสั่ง PUT เหมือนเดิมจะปรากฏผลลัพธ์ดังนี้
        A.B(1) A.B(2) A.B(3)....A.B(10)
        A.C(1) A.C(2) A.C(3).... A.C(10)
```

ถ้าหากว่า ใน data list ในคำสั่ง input สำหรับ list–directed หรือ edit–directed trans-mission ซึ่งเมื่อกำหนดค่าให้กับตัวแปรดังตัวอย่าง

```
GET LIST (N,(X(I)DO I = 1 TO N),J,K,SUBSTR (NAME,J,K));
```

เมื่อเครื่องคอมพิวเตอร์ execute คำสั่งข้างต้น ข้อมูลจะถูกเคลื่อนย้ายตามลำดับดังนี้

1. มูลค่าใหม่หนึ่งจำนวนกำหนดให้กับตัวแปร N

2. กำหนดมูลค่าอีลิเม้นต์ของ array X ตามที่กำหนดใน repetitive specification ตามลำดับคือ X(1),X(2),...X(N), ด้วยมูลค่าใหม่ของ N ซึ่งใช้บอกจำนวน item

3. มูลค่าใหม่หนึ่งจำนวนให้กับตัวแปร J

4. มูลค่าใหม่หนึ่งจำนวนกำหนดให้กับตัวแปร K

5. substring ความยาว K ให้แบ่งจาก string variable NAME เริ่มตั้งแต่ character ตัวที่ J

**List–directed Data Specification**

รูปแบบทั่วไปสำหรับ a list–directed data specification ทั้ง input หรือ output มีดังนี้

[LIST] (data–list)

data list เหมือนกับที่ได้กล่าวมาแล้วข้างต้น ส่วน keyword LIST เป็นตัวบอกว่าเป็นการเคลื่อนย้ายข้อมูลชนิด list–directed

**ตัวอย่าง**

```
LIST (CARD, RATE, DYNAMIC_FLOW)
LIST ((THICKNESS (DISTANCE) DO DISTANCE = 1 TO 100))
LIST (A,B,C,D)
LIST (A*B/C,(X + Y)**2)
```

สำหรับ specification ในตัวอย่างสุดท้ายใช้เฉพาะใน output เท่านั้น เพราะเป็นมูลค่าของ expression ซึ่งจะถูกประเมินผลเมื่อเครื่องทำการ execute คำสั่งนี้ และผลลัพธ์จะถูกนำไปใน stream

**List–Directed Input Format**

ถ้าชื่อของข้อมูลเป็น an array ข้อมูลประกอบด้วยมูลค่าคงที่จำนวนหนึ่ง, ข้อมูลแรกจะถูกกำหนดให้เป็นอีลิเม้นต์ตัวแรกของ array, มูลค่าคงที่ตัวที่สอง จะถูกกำหนดให้เป็นอีลิเม้นต์ตัวที่สอง ตามลำดับในลักษณะของ row–major order

A structure name ใน data list หมายถึง list ของ element variable และ arrays ในลักษณะที่กำหนดโดยโครงสร้างนั้น

Input data items ใน stream ต้องแยกกันด้วยเครื่องหมาย blank หนึ่งตัว หรือเครื่องหมาย comma หนึ่งตัว หรือ blanks จำนวนหนึ่งก็ได้ สำหรับ null field ให้แทนด้วยเครื่องหมาย comma 2 ตัว ซึ่งมีความหมายว่าข้อมูลของ item ตัวนี้ใน data list จะไม่มีการเปลี่ยนแปลง

**ตัวอย่าง**

| | |
|---|---|
| 3, 56 | item แรกมีมูลค่าเท่ากับ 3 |
| หรือ | item ถัดไปมีมูลค่าเท่ากับ 56 |
| 13 789 | item แรกมีมูลค่าเท่ากับ 13 |
| หรือ | item ถัดไปมีมูลค่าเท่ากับ 789 |
| −1.0,,8 | item แรกมีมูลค่าเท่ากับ −1.0 |
| | item ตัวที่สองมีมูลค่าเดิม และ item ตัวที่สามมีมูลค่าเท่ากับ 8 |

ถ้าข้อมูลเป็นมูลค่าคงที่ชนิด character−string, เครื่องหมาย quotation marks ซึ่งกำกับข้างหน้าหนึ่งตัว ข้างหลังหนึ่งตัว จะถูกนำออก และ character ภายในเครื่องหมายนี้จะเป็นมูลค่าของ character string

**ตัวอย่าง**

คำสั่ง GET LIST (NAME);

'RAMKHAMHAENG U.'

มูลค่าของ string คือ RAMKHAMHAENG U.

ถ้าข้อมูลเป็นมูลค่าคงที่ชนิด bit−string, เครื่องหมาย quotation marks และอักษร B ขวามือสุดจะถูกนำออกและ character ภายใน จะเป็น bit string

**ตัวอย่าง**

'0011'B

มูลค่าของ bit string คือ 0011

ถ้าข้อมูลเป็นมูลค่าคงที่คณิตศาสตร์ มันจะถูกเปลี่ยนเป็นรหัสของ ข้อมูลคณิตศาสตร์ที่มี base, scale, mode และ precision ตามที่กำหนดไว้

**List Directed Output Format**

มูลค่าของ element variables และ expressions ใน data list จะถูกเปลี่ยนรูปให้เป็น character และเคลื่อนย้ายไปยัง data stream การเปลี่ยนรูปชนิด arithmetic ให้เป็น character ใช้กฎเกณฑ์ปกติ ยกเว้น item ที่เป็น floating−point เท่านั้น

เครื่องหมาย blank ใช้สำหรับแยก data items สำหรับ PRINT file items จะถูกแยกจากกันตาม program tab ที่กำหนดเอาไว้เป็นตำแหน่งที่ 1, 25, 49, 73, 97 และ 121

Binary data item ถูกเปลี่ยนรูปเป็นสัญญลักษณ์ ฐานสิบก่อนนำเข้าไปใน stream

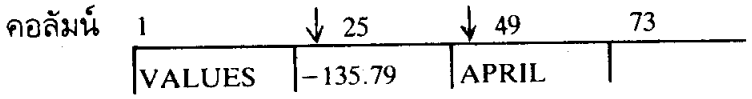
สำหรับมูลค่าของ numeric character string ถูกเปลี่ยนเป็นมูลค่า character string

Bit string เปลี่ยนเป็น character แทน bit–string constants ซึ่งประกอบด้วย character 0 และ 1 ภายในเครื่องหมายคำพูด ตามด้วยอักษร B

## ตัวอย่าง

```
DCL PRICE DEC FIXED (5,2) INIT (–135.79),
    MONTH CHAR(5)         INIT ('APRIL');
    PUT LIST ('VALUES', PRICE, MONTH);
```

เครื่องจะพิมพ์ผลลัพธ์ให้ดังนี้




หรือจะใช้คำสั่งข้างล่างนี้ ซึ่งจะให้ผลลัพธ์อย่างเดียวกัน

```
PUT LIST ('VALUES');
PUT LIST (PRICE);
PUT LIST (MONTH);
```

### Data–Directed Data Specification

รูปแบบทั่วไปของ a data–directed data specification ทั้ง input หรือ output มีดังนี้

```
DATA [(data–list)]
```

## กฎเกณฑ์ทั่วไป

1. data list เหมือนกับที่กล่าวมาแล้วในบทนี้ สำหรับ input data list จะมี subscripted names ไม่ได้

2. ถ้าไม่มี data list เครื่องจะถือว่ามีโดยสมมติ

3. เครื่องหมาย semi colon หนึ่งตัวหรือ an end of file ใน input stream เป็นตัวบอกขอบเขตสิ้นสุด การเคลื่อนย้ายข้อมูลใน output หลังจากเคลื่อนย้าย data item ตัวสุดท้ายด้วยคำสั่ง PUT แล้วเครื่องจะให้เครื่องหมาย semi colon หนึ่งตัว

### Data–Directed Data in the Stream

ข้อมูลใน stream ซึ่งจะถูกเคลื่อนย้ายชนิด data–directed อยู่ในรูปแบบของ a list of element assignments

สำหรับ problem data ในรูปแบบทั่วไปดังนี้

```
element-variable = data value
[{b | ,} element-variable = data value]...;
```

## กฎเกณฑ์ทั่วไป

1. element variable อาจจะเป็น a subcripted name ก็ได้ subscripts ต้องเป็น optionally signed decimal integer constant

2. Input, element assignment อาจจะแยกกันด้วยเครื่องหมาย blank หนึ่งตัว (b ในรูปแบบข้างต้น) หรือเครื่องหมาย comma หนึ่งตัว ถ้ามี blanks มากกว่าหนึ่งตัว เครื่องจะไม่สนใจ สำหรับ PRINT files, item ถูกแยกกันด้วยโปรแกรม tab setting

3. มูลค่าของข้อมูลแต่ละตัวใน stream จะมีรูปแบบอย่างใดอย่างหนึ่งเหมือนกับที่บรรยายใน list-directed transmission

4. Input, เครื่องหมาย semi colon หนึ่งตัว ตามหลัง an element, assignment เป็นตัวบอกว่าจบ list of element assignment

สำหรับการ execute หนึ่งคำสั่ง GET DATA และเป็นตัวกำหนดจำนวน element assignment ซึ่งจะถูกเคลื่อนย้ายคำสั่ง 1 คำสั่ง

ใน Output เครื่องหมาย semi colon หนึ่งตัวจะถูกส่งออกมาหลังจากจบคำสั่ง PUT DATA

**Data-Directed Data Specification for input**

## ตัวอย่าง

DATA (B,A,C,D)

เมื่อ A,B,C, และ D เป็น element variables

และมี input data stream ข้างล่างนี้

```
A = 2.5, B = .0047, D = 125, Z = 'ABC';
```

## หมายเหตุ

C ปรากฏใน data list แต่ไม่มีใน stream หมายถึงว่า มูลค่าของ C ไม่เปลี่ยนแปลงแต่ Z ไม่มีใน data list ฉะนั้นจึงเกิดเงื่อนไข NAME ขึ้น

**ตัวอย่าง**

```
DECLARE X(2,3);
```

เมื่อ

```
Data specification คือ DATA (X)
Input data stream คือ X(1,1) = 7.95
                      X(1,2) = 8085,
                      X(1,3) = 73;
```

ในกรณีนี้ กำหนดมูลค่าให้อีลิเม้นต์เพียง 3 ตัว ฉะนั้นอีลิเม้นต์ส่วนที่เหลือจะเหมือนเดิม

**Data–Directed Data specification for output**

กฎเกณฑ์ทั่วไปสำหรับ data–directed output

1. An element ของ data list อาจจะเป็น an element, array, หรือ structure variable หรือ a repetitive specification สำหรับ problem data, names ซึ่งปรากฏใน data list รวมทั้งมูลค่าของมัน จะถูกเคลื่อนย้ายในรูปของ a list of element assignments แยกกันด้วยเครื่องหมาย blanks และจบด้วยเครื่องหมาย semi colon 1 ตัว (สำหรับ PRINT file, items แยกกันด้วยโปรแกรม tab setting)

2. Array variables ใน data list จะถูกปฏิบัติเป็น a list of element ที่มี subscripted ในลักษณะของ row–major order

**ตัวอย่าง**

```
DECLARE X(2,4) FIXED;
```

ถ้า X ปรากฏใน data list ดังนี้

```
DATA (X)
```

ใน output, output data stream จะมีรูปแบบดังนี้

```
X(1,1) = 1    X(1,2) = 2    X(1,3) = 3    X(1,4) = 4
X(2,1) = 5    X(2,2) = 6    X(2,3) = 7    X(2,4) = 8;
```

**ตัวอย่าง**

```
AB : PROCEDURE;
     DECLARE (A(6), B(7)) FIXED;
     GET FILE (X) DATA (B);
     DO I=1 TO 6;
     A(I) = B(I + 1) + B(I);
     END;
     PUT FILE (Y) DATA (A);

END AB;
```

ตามตัวอย่างโปรแกรมข้างต้น input stream จะมีลักษณะดังนี้

B(1) = 1, B(2) = 2 B(3) = 3,

B(4) = 1, B(5) = 2, B(6) = 3, B(7) = 4;

และ output stream มีลักษณะดังนี้

A(1) = 3 A(2) = 5 A(3) = 4 A(4) = 3

A(5) = 5 A(6) = 7;

**ตัวอย่างโปรแกรม**

1. จงเขียน function procedure ตรวจสอบว่า character–string argument 1 ชุด ซึ่งได้มาจาก main procedure นั้นเป็นตัวอักษรทั้งหมดในภาษาอังกฤษ (A–Z) หรือไม่ โดยกำหนดว่า string มีความยาวไม่เกิน 25 ตัว ถ้า character string ทุกตัวใน string นั้นเป็นตัวอักษรในภาษาอังกฤษทั้งหมด ให้มูลค่าของฟังก์ชันเป็น bit string 1 ตัว มีมูลค่าเป็น '1'B ถ้าเป็นอย่างอื่นให้มูลค่าของฟังก์ชันเป็น bit string หนึ่งตัวเช่นกัน แต่มูลค่าเป็น '0'B

PL/1 OPTIMIZING COMPILER PROG 6: PROCEDURE OPTIONS (MAIN);

SOURCE LISTING

```
STMT LEV NT
  1       0   PROG6:PROCEDURE OPTIONS (MAIN);
  2   1   0         DCL STRING CHAR (25) VAR;
  3   1   0         DCL FUNCTION BIT (1);
  4   1   0         DCL SYSIN INPUT, SYSPRINT OUTPUT;
  5   1   0         ON ENDFILE (SYSIN) STOP;
  6   1   0         PUT PAGE;
  7   1   0         DO WHILE (1 = 1);
  8   1   1             GET LIST (STRING);
  9   1   1             FUNCTION = FN (STRING);
 10   1   1             PUT SKIP (2) DATA (STRING, FUNCTION);
 11   1   1         END;
 12   1   0   FN:  PROCEDURE (ST) RETURNS (BIT);
 13   2   0       DCL ST CHAR (*);
 14   2   0       DCL PAT CHAR (26) INIT ('ABCDEFGHIJKLMNOPQRSTUVW
                    XYZ');
 15   2   0       DCL VERIFY BUILTIN;
 16   2   0       IF VERIFY (ST,PAT) = 0
                      THEN RETURN ('1'B);
 17   2   0           ELSE RETURN ('0'B);
 18   2   0   END FN;
 19   1   0   END PROG6;
STRING = 'HI STRANGER X I LOVEYOU'                    FUNCTION = '0'B;
STRING = 'SMILE'                    FUNCTION = '1'B;
STRING = 'I LOVE MY COUNTRY'                          FUNCTION = '0'B;
STRING = ZZZZZZZYYYYYYYXXXWWU'                        FUNCTION = '1'B;
STRING = 'MNBVCXZASDFGHJKLOPIUYTREW'                  FUNCTION = '1'B;
STRING = '+ -) - |123$$$.....,###%@,'                 FUNCTION = '0'B;
STRING = 'DERTYUHGBVNMC'      FUNCTION = '1'B;
```

2. จงเขียนโปรแกรมทำตามขั้นตอนต่อไปนี้

a) อ่าน character string 80 ตัวจากบัตรข้อมูล หนึ่งใบ

b) เอา character 8 ตัวท้ายสุดของ string ชุดนี้ออกทำให้เหลือ 72 ตัว

c) ตัด character 6 ตัวแรกของ string ชุดนี้ทิ้ง

d) เอา character blanks ทั้งหมดที่อยู่ใน string 66 character ตัวถัดไปของ string ชุดนี้ออก

e) พิมพ์ข้อมูลเริ่มต้นจากข้อ a) และพิมพ์ข้อมูลที่เหลืออยู่ท้ายสุดจากข้อ d)

```
PL/I OPTIMIZING COMPILER
                                SOURCE LISTING
STMT LEV NT
  1        0   PROG1:PROCEDURE OPTIONS (MAIN);
  2    1   0       DCL (A,B,C,D) CHAR (80) VARYING INIT ('');
  3    1   0       DCL SUBSTR BUILTIN, LENGTH BUILTIN;
  4    1   0       DCL SYSIN INPUT, SYSPRINT OUTPUT;
  5    1   0       GET EDIT (A) (A(80));
  6    1   0       B = SUBSTR (A,1,72);
  7    1   0       C = SUBSTR (B,7);
  8    1   0       I = 1;
  9    1   0       L = LENGTH (C);
 10    1   0       DO WHILE (I< = L);
 11    1   1       IF SUBSTR (C,I,1) ¬= ' ' THEN DO;
 12    1   2                                 D = D || SUBSTR (C,I,1);
 13    1   2                                 I = I + 1;
 14    1   2                                 END;
 15    1   1                             ELSE I = I + 1;
 16    1   1       END;
 17    1   0       PUT PAGE;
 18    1   0       PUT SKIP(3) DATA (A);
 19    1   0       PUT SKIP(3) DATA (B);
 20    1   0       PUT SKIP(3) DATA (C);
 21    1   0       PUT SKIP(3) DATA (D);
 22    1   0   END PROG1;
```

A ='A A S D F G H G F D S A F T R E W Q T T R E W Q Y H U J M N B V C X Z D Z A D S';

B = 'A A S D F G H G F D S A F T R E W Q T T R E W Q Y H U J M N B V C X Z D';

C = 'D F G H G F D S A F T R E W Q T T R E W Q Y H U J M N B V C X Z D';

D = 'DFGHGFDSAFTREWQTTREWQYHUJMNBVCXZD';

**Edit–Directed Data Specification**

ไม่ว่าจะเป็น input หรือ output มีรูปแบบทั่วไปดังนี้

```
EDIT {(data–list) (format–list)} [(data–list) (format–list)]...
```

1. data list, จะต้องอยู่ภายในเครื่องหมายวงเล็บ, และ format list ก็ต้องอยู่ภายในเครื่องหมายวงเล็บเช่นเดียวกัน ซึ่งประกอบด้วย format items ตั้งแต่หนึ่งตัวขึ้นไป

format items แบ่งออกเป็น 3 ชนิดด้วยกันคือ

| | |
|---|---|
| data format items | ใช้บรรยายลักษณะข้อมูลใน stream |
| control format items | ใช้บรรยายเกี่ยวกับ หน้ากระดาษ, บรรทัด และการเว้นที่ |
| remote format items | ใช้กำหนด label ของแต่ละคำสั่งซึ่งประกอบด้วย format list ที่ใช้ |

**หมายเหตุ**

program–control variable จะกำหนดใน data list สำหรับการเคลื่อนย้ายข้อมูล ชนิด edit–directed transmission ไม่ได้

2. สำหรับ input, ข้อมูลใน stream เป็น string ของ characters ซึ่งต่อเนื่องกัน ไม่ได้แยกเป็น data item แต่ละชุด จำนวนของ character สำหรับ data item แต่ละชุดกำหนดโดย format item ใน format list

3. สำหรับ output, มูลค่าของ item แต่ละตัวใน data list จะถูกเปลี่ยนให้อยู่ในรูปแบบซึ่งกำหนดโดย format item ซึ่งใช้คู่กันและนำไปใส่ใน stream ใน field ซึ่งกำหนดความกว้างโดย format item

4. ไม่ว่าจะเป็น input หรือ output ก็ตาม data format item แรกจะใช้คู่กับ item แรกใน data list ส่วน data format item ที่สองจะใช้คู่กับ item ที่สองใน data list เช่นนี้ตามลำดับ **ถ้าใน format–list มีจำนวน format items น้อยกว่าจำนวน item ที่คู่กันใน data list เครื่องจะจัด format items ให้ใช้ซ้ำอีกครั้ง**, ถ้าจำนวน format items มีมากเกินไป เครื่องจะไม่รับ (ignores) ส่วนที่เกิน

**ตัวอย่าง**

```
PUT LIST (A,B,C,D,E,F,G,H,I,J) (5 F(3));
```

หรือ `PUT LIST (A,B,C,D,E,F,G,H,I,J) ((5)F(3));`

ในที่นี้เลข 5 เป็น iteration facter ใน format list มี data format items 5 ชุด แต่มี item ใน data list ซึ่งใช้คู่กัน 10 ตัว ดังนั้น item ที่ 6 ใน data list จึงต้องใช้ data format item แรกซ้ำอีกครั้ง เช่นนี้เรื่อย ๆ ไป

แต่ถ้า format list มี data format items 10 ชุด แต่มี items ใน data list ซึ่งจะใช้คู่กัน 5 ชุด เท่านั้น ฉะนั้น data format item ตั้งแต่ตัวที่ 6 จนถึงตัวที่ 10 เครื่องจะไม่รับ

5. สำหรับ array หรือ structure variable ใน data list ซึ่งเท่ากับจำนวน n items ใน data-list เมื่อ n เป็นจำนวน element items ใน array หรือ structures แต่ละตัวจะต้องใช้คู่กับ data format item หนึ่งตัวแยกต่างหากจากกัน

6. ถ้าใน format list มี control format item เมื่อเครื่อง execute ก็จะทำ action control นั้นและ data list item ก็จะคู่กับ format item ตัวถัดไป

7. การเคลื่อนย้ายข้อมูลซึ่งกำหนดนั้น จะเสร็จเรียบร้อยเมื่อ item ตัวสุดท้ายใน data-list ได้รับการประเมินผลโดยใช้ format item ซึ่งใช้คู่กันแล้ว, และถ้ายังมี format items หรือ control format item อีกเครื่องก็จะไม่สนใจ

8. ใน output, data items ไม่ได้แยกจากกันโดยอัตโนมัติ แต่ arithmetic data item โดยปกติจะมี blanks นำหน้าเพราะว่าเป็นไปตามกฎเกณฑ์การเปลี่ยนรูปข้อมูล และการตัดเลขศูนย์ข้างหน้าตัวเลขทิ้ง

**ตัวอย่าง**

```
GET EDIT (NAME, DATA, SALARY)
         (A(N),X(2),A(6),F(6,2));
PUT EDIT ('INVENTORY = ' || INUM, INVCODE)
         (A,F(5));
```

ตัวอย่างแรก กำหนดว่า character จำนวน N ตัวแรกใน stream มีลักษณะเป็น character string และเป็นมูลค่าของตัวแปร NAME character อีก 2 ตัวถัดไปให้ข้ามไป, อีก 6 ตัวถัดไปเป็นมูลค่าของตัวแปร DATA มีรูปเป็น character อีก 6 ตัวถัดไป เป็นมูลค่าคงที่ชนิด decimal fixed-point อาจจะมีเครื่องหมายกำกับหรือไม่ก็ได้ ให้เป็นมูลค่าของตัวแปรชื่อ SALARY

ตัวอย่างที่สอง กำหนดว่า character string 'INVENTORY = ' ให้ต่อกับมูลค่าของ INUM และใส่ลงไปใน stream ภายใน field ซึ่งมีความกว้างเท่ากับความยาวของ string และมูลค่าของตัวแปร INVCODE จะถูกเปลี่ยนรูปเป็นมูลค่าคงที่ชนิด decimal fixed-point อาจจะมีเครื่องหมายกำกับหรือไม่มีก็ได้ นำไปไว้ใน stream ในลักษณะชิดขวา ใน field ซึ่งมีความกว้างเท่ากับ 5 ตำแหน่ง

**หมายเหตุ**

สำหรับมูลค่าของ expressions และมูลค่าคงที่จะปรากฏใน output data list เท่านั้น

**Format lists**

ใน edit-directed data specification แต่ละชุดต้องใช้คู่กับ format list ซึ่งมีรูปแบบทั่วไปดังนี้

(format-list)

เมื่อ 'format-list' หมายถึง

$$\left\{\begin{array}{l}\text{item}\\ \text{n item}\\ \text{n (format-list)}\end{array}\right\}\left[\begin{array}{l},\ \text{item}\\ ,\ \text{n item}\\ ,\ \text{n (format-list)}\end{array}\right]\ldots$$

**กฎเกณฑ์ของ syntax**

1. 'item' แต่ละตัวในที่นี้ หมายถึง format item

2. ตัวอักษร n หมายถึง an interation factor ซึ่งอาจจะเป็น an expression ภายในเครื่องหมายวงเล็บ หรือ มูลค่าคงที่จำนวนเต็มฐานสิบ ไม่มีเครื่องหมายกำกับ, ถ้าเป็นตัวหลัง ต้องมี blank หนึ่งตัว แยกระหว่างมูลค่าคงที่นั้น กับ format item ที่ตามหลัง

iteration factor เป็นตัวบอกว่า format item ตัวนั้นหรือ format-list นั้นจะใช้ทั้งหมด n ครั้ง, ถ้า iteration factor มีมูลค่าเป็นศูนย์ หมายถึงเครื่องจะไม่ใช้ format item หรือ format list ชุดนั้น (ให้ข้ามไปเลย) ฉะนั้น data list item ก็จะใช้คู่กับ data format item ตัวถัดไป

ถ้า iteration factor นั้นเป็น an expression เมื่อประเมินผลแล้วจะถูกเปลี่ยนรูปเป็นมูลค่าคงที่จำนวนเต็ม ต้องไม่มีมูลค่าเป็นลบ หมายถึง แต่ละกลุ่มของ iteration format item หรือ format list หมายถึง item หรือ list ของ items ซึ่งตามหลังทางขวามือทันทีของ iteration factor นั้น

## กฎเกณฑ์ทั่วไป

format item มี 3 ชนิด คือ data format items, control format item และ remote format item, ชนิดแรกกำหนดรูปแบบภายนอกของข้อมูล field ซึ่งกำลังจะใช้, ชนิดที่สองกำหนดหน้ากระดาษ, บรรทัดที่, คอลัมน์ที่, และเว้นบรรทัด ส่วนชนิดที่สาม อนุญาตให้ format items ซึ่งกำหนดในคำสั่ง FORMAT แยกต่างหากที่ไหนก็ได้ภายในบล๊อก

รายละเอียดเกี่ยวกับชนิดของ format items อยู่ใน section E ของหนังสือเล่มนี้และที่จะกล่าวต่อไปนี้ เป็นเพียงตัวอย่างการใช้ format items เท่านั้น

### Data Format Items

ใน input, data format item แต่ละตัวจะบอกจำนวน characters ซึ่งคู่กับ data item และมีลักษณะเป็นข้อมูลภายนอกอย่างไร, data item ถูกกำหนดให้คู่กับ ตัวแปรใน data list และถ้าจำเป็นจะถูกเปลี่ยนรูปให้มีลักษณะเหมือนกับ attribute ของตัวแปรนั้น

ใน output, มูลค่าของ อีลิเม้นต์ ใน data list จะถูกเปลี่ยนรูปให้เป็น character เหมือนที่กำหนดใน format item และนำไปใส่ใน data stream

data format items มี 6 ชนิดคือ fixed–point(F), floating-point(E), complex(C), picture(P), character-string(A), และ bit-string(B) โดยที่มีรูปแบบทั่วไปดังนี้

```
F (w[,d[,p]])
E (w,d[,s])
C (real-format-item [,real-format-item])
P 'picture-specification'
A [(w)]
B [(w)]
```

เมื่อ w เป็น an expression หมายถึงจำนวน character ใน field นั้น

d หมายถึง จำนวนเลขหลังจุดทศนิยมว่ามีกี่หลัก, ถ้าไม่กำหนดไว้ หมายถึง fixed point สำหรับ real-format-item ใน complex format item อาจจะเป็น F, E หรือ P format item ก็ได้

picture-specification ของ P format item อาจจะเป็น

a numeric character specification หรือ a character string specification ก็ได้ใน output

## ตัวอย่างโปรแกรม

คำนวณค่า factorial 1 ถึง factorial 20

```
PL/1 OPTIMIZING COMPILER                    FACT: PROC OPTIONS (MAIN);

                        SOURCE LISTING

STMT LEV NT

 1     0  FACT:PROC OPTIONS (MAIN);
 2   1 0  DCL STAR CHAR (33) INIT ((33)'*');
 3   1 0  DCL MULT FLOAT (5);
 4   1 0  MULT = 01.00000E+00;
 5   1 0  PUT EDIT ('THE RESULTS OF FACTORIAL (1-20)')(PAGE,SKIP,X(49),A(33));
 6   1 0  PUT EDIT (STAR)(SKIP,X(49),A(33));
 7   1 0  DO I = 1 TO 20;
 8   1 1      MULT = MULT* I;
 9   1 1      PUT EDIT ('FACTORIAL',I,' = ',MULT)(SKIP(2),X(49),A(10),F(2),A(3),
                E(12,5));
10   1 1  END;
11   1 0  END FACT;
```

THE RESULTS OF FACTORIAL (1–20)

| | |
|---|---|
| FACTORIAL 1 = | 1.00000E+00 |
| FACTORIAL 2 = | 2.00000E+00 |
| FACTORIAL 3 = | 6.00000E+00 |
| FACTORIAL 4 = | 2.40000E+01 |
| FACTORIAL 5 = | 1.20000E+02 |
| FACTORIAL 6 = | 7.20000E+02 |
| FACTORIAL 7 = | 5.04000E+03 |
| FACTORIAL 8 = | 4.03200E+04 |
| FACTORIAL 9 = | 3.62880E+05 |
| FACTORIAL 10 = | 3.62880E+06 |
| FACTORIAL 11 = | 3.99168E+07 |
| FACTORIAL 12 = | 4.79002E+08 |
| FACTORIAL 13 = | 6.22702E+09 |
| FACTORIAL 14 = | 8.71782E+10 |
| FACTORIAL 15 = | 1.30767E+12 |
| FACTORIAL 16 = | 2.09228E+13 |
| FACTORIAL 17 = | 3.55687E+14 |
| FACTORIAL 18 = | 6.40236E+15 |
| FACTORIAL 19 = | 1.21645E+17 |
| FACTORIAL 20 = | 2.43290E+18 |

## แบบฝึกหัด

1. จงแก้ไขคำสั่งข้างล่างนี้ให้ถูกต้อง

   a) GET LIST UP, DOWN;

   b) PUT SKIP LIST (A B C)

   c) MEAN: PROCEDURE

   d) ENDVOLUME;

   e) A + B = C;

   f) PUT SKIP LIST (HEIGHT, WIDTH,);

   g) END, AREASO;

   h) GET LIST (AT OLCH, BOG);

   i) GO TO LAND;

   j) DO = 1 BY 1;

2. ข้อมูลมี 3 บรรทัดดังนี้

   ```
   ABCDEF
   GHIJKL
   MNOPQR
   ```

   จงบอกค่าของ A, B และ C หลังจากเครื่อง execute คำสั่งต่อไปนี้

   a) GET EDIT (A, B, C) (A (2), A(2), A(2));

   b) GET EDIT (A, B, C) (A (2));

   c) GET EDIT (A, B, C) (COLUMN (1). A(2));

   d) DO I = 1 TO 3;
      GET EDIT (A (I)) (A(2));
      END;

3. จงเขียนโปรแกรมอ่านข้อมูล ซึ่งเป็นเลขจำนวนเต็มจำนวนหนึ่งดังนี้

   **ตัวอย่าง** 6, 1, 5, 10, 10, 5, 1,

   เมื่อเลขตัวแรก (6) หมายถึงจะมีเลขตามมาอีก 6 ตัว, เลขตัวที่สองจนถึงตัวสุดท้าย ให้พิมพ์ออกมาเป็นรูปกราฟดังนี้

   Output :

   ```
   *
   *****
   **********
   **********
   *****
   *
   ```

4. จงเขียนโปรแกรมพิมพ์ตารางเปลี่ยนความยาว 1-12 นิ้ว ให้มีหน่วยเป็นเซ็นติเมตร (1 นิ้ว = 2.54 เซ็นติเมตร)

5. a) ฟังก์ชันทางคณิตศาสตร์เกี่ยวกับ square-wave ได้กำหนดดังนี้
ถ้า X อยู่ในช่วง $0 \leq x < 1, 2 \leq x < 3, 4 \leq x < 5$ และต่อ ๆ ไป, มูลค่าของ y จะเป็น 1
ถ้า x เป็นค่าอื่นที่ไม่ใช่มูลค่าข้างต้น, มูลค่าของ Y จะเป็น 0 กำหนดมูลค่าของ x มาให้ จงเขียน flowchart และโปรแกรมคำนวณหามูลค่าของ Y

b) กำหนดให้ B เป็น array 1 มิติ มีอีลิเม้นต์ทั้งหมด 30 ตัว แต่อีลิเม้นต์ทั้งหมดนี้มีอยู่ 1 ตัวที่มีมูลค่าเป็นศูนย์ จงเขียน flowchart และโปรแกรมค้นหาอีลิเม้นต์ตัวนี้แล้วแทนด้วยเลข 1

6. กำหนดข้อมูลในบัตรมีลักษณะดังนี้

```
8E8 2E8 2,, - 1E8,, 5.6, 3.4, -5
```

```
SAMPLE: PROCEDURE OPTIONS (MAIN);
        ON ENDFILE (SYSIN) STOP;
        PUT PAGE;
        DO WHILE (1 = 1);
        GET LIST (X, Y, L);
        Z = (X - Y) * L;
        PUT DATA (Z, L);
        END;
    END;
```

ให้แสดงผลลัพธ์ที่เครื่องจะพิมพ์ออกมาอย่างถูกต้อง

7. Magic square หมายถึง square matrix ใด ๆ ก็ตามที่ผลรวมของอีลิเม้นต์แต่ละโรว์ และผลรวมของอีลิเม้นต์แต่ละคอลัมน์ และผลรวมของอีลิเม้นต์ที่อยู่ในแนวเส้นทะแยงมุมทั้ง 2 ด้าน มีมูลค่าเท่ากันหมด

**ตัวอย่าง**

magic square ขนาด 3 × 3

$$\begin{bmatrix} 3 & 4 & 5 \\ 6 & 4 & 2 \\ 3 & 4 & 5 \end{bmatrix}$$

จงเขียนโปรแกรมตรวจสอบว่า matrix Aขนาด N × N เป็น magic square หรือไม่

8. จงเขียน flowchart และโปรแกรม ทำตามขั้นตอนต่อไปนี้

a) อ่าน integer n ซึ่งเป็นขนาดของ square matrix A

b) อ่าน matrix A ขนาด N×N

c) คำนวณและพิมพ์ผลรวมของอีลิเม้นต์ทุกตัวในแต่ละโรว์

d) คำนวนและพิมพ์ผลรวมของอีลิเม้นต์ทุกตัวในแต่ละคอลัมน์

e) คำนวนและพิมพ์ผลรวมของอีลิเม้นต์ทุกตัวในแนวเส้นทะแยงมุม (main diagonal)

9. กำหนดให้ค่าเบี่ยงเบนมาตรฐาน (sd) คำนวณได้จากสูตรข้างล่างนี้

$$sd = \sqrt{\frac{1}{n} \Sigma x^2_i - \bar{x}^2}$$

เมื่อ $\bar{x}$ เป็นมูลค่าเฉลี่ยของ $x_1$ ถึง $x_n$ เรียกว่า mean

$\frac{1}{n} \Sigma x^2_i$ เป็นมูลค่าเฉลี่ยของผลรวมของ $x^2_1$ ถึง $x^2_n$

จงเขียน flowchart และโปรแกรมอ่าน integer n แล้วพิมพ์ค่า n, จากนั้นอ่านข้อมูลของ $x_1$ ถึง $x_n$ คำนวณหาผลรวมของ $x_i$ และ $x_n^2$ แล้วคำนวณและพิมพ์ค่าเฉลี่ย $\bar{x}$ และค่าเบี่ยงเบนมาตรฐาน sd ตามลำดับ

10. ห้างสรรพสินค้าหัวหมากสาขาหนึ่งได้จดรายงานการขายเครื่องดื่มประเภทน้ำอัดลมจำนวน 4 ชนิด ในหนึ่งอาทิตย์ที่ผ่านมาได้ตัวเลขในตารางข้างล่างนี้

| | ชนิดที่ 1 | ชนิดที่ 2 | ชนิดที่ 3 | ชนิดที่ 4 |
|---|---|---|---|---|
| อาทิตย์ | 67 | 37 | 20 | 58 |
| วันจันทร์ | 50 | 35 | 22 | 62 |
| วันอังคาร | 58 | 40 | 20 | 60 |
| วันพุธ | 70 | 36 | 24 | 64 |
| วันพฤหัสบดี | 66 | 32 | 21 | 57 |
| วันศุกร์ | 70 | 30 | 25 | 63 |
| วันเสาร์ | 40 | 10 | 15 | 30 |

จงเขียน flowchart และโปรแกรม

1) พิมพ์ยอดขายว่า เมื่ออาทิตย์ที่ผ่านมานั้นน้ำอัดลมแต่ละชนิดขายได้จำนวนกี่ขวด
2) พิมพ์ยอดขายว่า ในแต่ละวันขายน้ำอัดลมได้กี่ขวด
3) พิมพ์จำนวนน้ำอัดลมทั้งหมดที่ขายได้ในอาทิตย์นั้น

**หมายเหตุ** ให้พิมพ์ heading ของตารางออกมาด้วย

11. บริษัทพรมไทยจำกัด ประกาศลดราคาพรม 15% ให้กับลูกค้าที่ซื้อพรมมากกว่า 75 หลา

จงเขียน flowchart และโปรแกรมอ่านบัตรข้อมูลซึ่งมีหมายเลขลูกค้าเป็นเลข 5 หลัก ราคาพรมต่อหลา (ราคาไม่เกินหลาละ 1,000 บาท) และจำนวนหลาที่สั่งซื้อ (ไม่เกิน 1,000 หลา) ถ้าหมายเลขลูกค้าเท่ากับ 99999 แสดงว่าบัตรข้อมูลหมดแล้ว จากบัตรข้อมูลเหล่านี้ ให้คำนวณจำนวนเงินที่ลูกค้าจะต้องจ่ายเป็นค่าพรมที่ซื้อ, แล้วพิมพ์หมายเลขลูกค้า, ความยาวของพรมที่ซื้อ, ราคาพรม, และราคาที่ลูกค้าต้องจ่าย เมื่อหักส่วนลดให้แล้ว ในกรณีจำนวนเงินที่ลูกค้าจะต้องจ่ายน้อยกว่า 10,000.00 บาท ลูกค้าจะต้องจ่ายเพิ่มอีก 100 บาท เป็นค่าขนส่งต่างหากด้วย

12. จงศึกษาโปรแกรมต่อไปนี้ แล้วแสดงผลลัพธ์ที่เครื่องจะพิมพ์ออกมาให้

```
DO: PROCEDURE OPTIONS (MAIN),
    DECLARE TEXT CHARACTER (5000) VARYING;
    DECLARE (I,N, COUNT) DECIMAL (5) FIXED;
    PUT PAGE;
    GET DATA;
    I = 1; COUNT =  0;N = LENGTH (TEXT);

STEP_ONE:
    IF SUBSTR (TEXT,I, 1) = SUBSTR (TEXT,I + 1, 1)
       THEN COUNT = COUNT + 1;
    IF I = N - 1 THEN GO TO STEP_TWO;
    I = I + 1; GO TO STEP_ONE;

STEP_TWO:
    PUT SKIP (2) LIST ('NUMBER OF LETTERS =',COUNT);
END DO;
```

ลักษณะของข้อมูล

```
TEXT = 'LITTLE DID OONA REALIZE BETTER BUTTER WAS NEEDED
        FOR THE BATTER';
```

13. ให้นักศึกษาพิจารณาโปรแกรมข้างล่างนี้ แล้วตอบคำถามตอนท้าย

```
FIRST       :PROCEDURE OPTIONS (MAIN);
             DCL (LINE, PRINT) CHAR (80);
             ON ENDFILE (SYSIN) GO TO LABEL_4;


LABEL_1     :DO M = 1 BY 1;
             GET EDIT (LINE) (A(80));
             PUT EDIT (LINE) (COL(1),A);
             PRNT = ' ';

LABEL_2     :DO N = 1 BY 1;
             K = INDEX (LINE, '0');
             IF K = 0 THEN GO TO LABEL_3;
             SUBSTR (LINE, K, 1) = ' ';
             SUBSTR (PINT, K, 1) = '/';
             END LABEL_2;

LABEL_3     :PUT SKIP(0) EDIT (PRINT)(A);
             END LABEL_1;

LABEL_4     :STOP;
             END FIRST;
```

**บัตรข้อมูล** มี 2 ใบดังนี้

```
AAABBBCCC000111222333444DDDEEE000111FFF
1111122222333334444455555666660000777788888
```

จงแสดงลักษณะของผลลัพธ์ที่เครื่องจะพิมพ์ออกมาให้

เฉลยข้อ 7

OPTIMIZING COMPILER EXP 4: PROG OPTIONS ( MAIN);

SOURCE LISTING

```
STMT LEV NT
  1       0     EXP4: PROG OPTIONS (MAIN);
  2    1  0           DCL SYSIN INPUT, SYSPRINT OUTPUT,
                      (M,B,C,X,Y) FIXED DEC (3),
                      (N,I,J,K) FIXED DEC (1),
                      A(N,N) FIXED DEC (2) CONTROLLED;
  3    1  0           ON ENDFILE (SYSIN) STOP;
  4    1  0        IN: GET LIST (N);
  5    1  0           ALLOCATE A;
  6    1  0           GET LIST (((A(I,J) DO J = 1 TO N) DO I = 1 TO N));
                      M,B,C = 0;
  8    1  0           M = N*(N**2 + 1)/2; DO I = 1 TO N; X,Y = 0;
                      DO K = 1 TO N;
 12    1  2           X = X + A(I,K); Y = Y + A(K,I); END;
                      IF X¬ = M THEN GO TO I2;
 16    1  1           IF Y¬ = M THEN GO TO I2; B = B + A(I,I);
                      C = C + A(I,N - I + 1);
 19    1  1           END; IF B = M & C = M
 20    1  0           THEN DO;
 21    1  1                PUT EDIT (((A(I,J) DO J = I TO N) DO I = 1 TO N))
                             (SKIP,COL(35),(N) (F(2),X(2)));
 22    1  1                PUT SKIP EDIT ('THIS MATRIX IS MAGIC
                             SQUARE MATRIX') (COL(35),A);
 23    1  1                END;
 24    1  0           FREE A;
 25    1  0                GO TO IN;
 26    1  0        I2:     PUT EDIT (((A(I,J) DO J = 1 TO N) DO I = 1 TO N))
                             (SKIP, COL(35),(N) (F(2),X(2)));
```

```
27   1   0                    PUT SKIP EDIT ('THIS MATRIX IS NOT MAGIC
                                  SQUARE MATRIX') (COL(35),A);
28   1   0                FREE A;
29   1   0                    GO TO IN;
30   1   0  END EXP4;
```

```
   1   2   3
   4   5   6
   7   8   9
THIS MATRIX IS NOT MAGIC SQUARE MATRIX
   1   2   3
   5   4   6
   7   9   8
THIS MATRIX IS NOT MAGIC SQUARE MATRIX
   8   1   6
   3   5   7
   4   9   2
THIS MATRIX IS MAGIC SQUARE MATRIX
   9   3  22  16  15
   2  21  20  14   8
  25  19  13   7   1
  18  12   6   5  24
  11  10   4  23  17
THIS MATRIX IS MAGIC SQUARE MATRIX
   1   2   3   4   5
   6   7   8   9  10
  11  12  13  14  15
  16  17  18  19  20
  21  22  23  24  25
THIS MATRIX IS NOT MAGIC SQUARE MATRIX
```

```
2   7   6
9   5   1
4   3   8
THIS MATRIX IS MAGIC SQUARE MATRIX
8   3   4
1   5   9
6   7   2
THIS MATRIX IS MAGIC SQUARE MATRIX
2   9   4
7   5   3
6   1   8
THIS MATRIX IS MAGIC SQUARE MATRIX
```

**เฉลยข้อ 9**

OPTIMIZING COMPILER ASSIGN 5: PROCEDURE OPTIONS ( MAIN);

SOURCE LISTING

```
STMT LEV NT
   1      0  ASSIGN5:PROCEDURE OPTIONS (MAIN);
   2   1  0  DCL (X,N) FIXED DEC (2),Y FIXED DEC(3),SX FIXED DEC(5),
              SY FIXED DEC(6), M FIXED DEC(6,2), SD FIXED DEC(5,2),
              Z FIXED DEC(5,2), SQRT BUILTIN;
   3   1  0  DCL SYSIN INPUT, SYSPRINT OUTPUT;
   4   1  0  PUT SKIP EDIT('PROGRAM COMPUTE') (LINE(11),X(58),A);
   5   1  0  PUT SKIP(2) EDIT('MEAN AND STANDARD DEVIATION')(X(52),A);
   6   1  0  PUT SKIP EDIT('2') (X(82),A);
   7   1  0  PUT SKIP EDIT('RANK','X','X') (X(47),A,X(14),A,X,(15),A);PUT SKIP,
   9   1  0  N,SX,SY = 0;READ:GET SKIP EDIT(X) (F(2));
              ON ENDFILE (SYSIN) GOTO FINISH;
  12   1  0  N=N+1;SX=SX+X;Y=X**2;SY=SY+Y;
              PUT SKIP(2) EDIT(N,X,Y)(X(48),F(2),X(14),
  17   1  0  F(2),X(14),F(3));GOTO READ;FINISH:M=SX/N;Z=SY/N-M**2;
              SD=SQRT(Z);
  21   1  0  PUT SKIP (2) EDIT ((33) '*')  (X(49),A);
  22   1  0  PUT SKIP(3) EDIT('N = ',N) (X(66),A,F(2));
  23   1  0  PUT SKIP(2) EDIT('SUM X =',SX) (X(62),A,F(5));
              PUT SKIP EDIT('2') (X(67),A);
  25   1  0  PUT SKIP EDIT('SUM X =',SY) (X(62),A,F(6));
              PUT SKIP(2) EDIT('MEAN = ',M) (X(63),A,F(6,2));
  27   1  0  PUT SKIP(2) EDIT('SD. =',SD)(X(64),A,F(5,2));
  28   1  0  PUT SKIP (2) EDIT ((33) '*')  (X(49),A);
  29   1  0  END ASSIGN5;
```

PROGRAM COMPUTE

MEAN AND STANDARD DEVIATION

| RANK | X | $X^2$ |
|---|---|---|
| 1 | 8 | 64 |
| 2 | 14 | 196 |
| 3 | 20 | 400 |
| 4 | 15 | 225 |
| 5 | 25 | 625 |
| 6 | 30 | 900 |
| 7 | 6 | 36 |
| 8 | 12 | 144 |
| 9 | 13 | 169 |
| 10 | 24 | 576 |

**********

N = 10

SUM X = 167

SUM $X^2$ = 3335

MEAN = 16.70

SD. = 7.38

**********