

บทที่ 9

ตัวแปรประเภทต่าง ๆ

โดยปกติแล้วคอมพิวเตอร์จะปฏิบัติงาน โดยการนำ ข้อมูลไปประมวลผลตามที่โปรแกรมกำหนด จนกว่าจะบรรลุผลตามที่กำหนดไว้ ดังนั้นการเรียนรู้เรื่องเกี่ยวกับข้อมูลว่ามีที่กำเนิดใด มีวิธีการจัดเก็บแบบใดจึงเป็นพื้นฐานที่สำคัญที่จะนำไปสู่ความสำเร็จของการปฏิบัติงานด้วยคอมพิวเตอร์

แหล่งที่รับข้อมูล

เราสามารถแบ่งที่มาของข้อมูลที่คอมพิวเตอร์รับไปปฏิบัติงานได้ออกมา เป็น 2 แหล่งใหญ่ๆ คือ

1. เกิดขึ้นภายในโปรแกรม
2. เกิดขึ้นจากแหล่งข้อมูลภายนอก ซึ่งอาจจะมาจาก external file เช่น แป้นพิมพ์(keyboard) ดิสก์ เทป เป็นต้น

1. การสร้างข้อมูลที่เกิดขึ้นภายในโปรแกรมนั้นจะมีลักษณะดังนี้คือเราถือว่า ข้อมูลนั้นเป็น คำสั่งประเภทหนึ่งในโปรแกรม ดังนั้นการสร้างข้อมูลในลักษณะเช่นนี้มักจะเหมาะสมกับข้อมูลที่มีจำนวนตัวแปรไม่มากนัก และมีลักษณะ ไม่ค่อย เปลี่ยนแปลงบ่อยนัก เพราะถ้าหากข้อมูลมีจำนวนมาก ก็จะทำให้โปรแกรมที่มีขนาดใหญ่มากตามไปด้วยจนอาจจะเกินขีดจำกัดที่จะสามารถนำเข้าไปปฏิบัติงานได้ในเครื่องคอมพิวเตอร์ได้ โดยเฉพาะกับเครื่องประเภทไมโครคอมพิวเตอร์ ตัวอย่าง เช่นการนำแฟ้มข้อมูลของพนักงานในบริษัทจำนวนสองหมื่นคนใส่เข้าไปเป็นส่วนหนึ่งในโปรแกรม หรือในกรณีที่มีข้อมูลภายในโปรแกรม มีการเปลี่ยนแปลงบ่อยมาก ตัวอย่าง เช่น เราจะแก้สมการทางคณิตศาสตร์ ดังต่อไปนี้

$$Y = AX^2 + BX + C \quad \text{โดยการหาค่า } Y$$

ณ.ที่ X มีค่าตั้งแต่ 0, 1, 2, ..., 100 และ ค่าของสัมประสิทธิ์ A, B, C ในสมการมีค่าเปลี่ยนแปลงบ่อยมากในแต่ละครั้งของการคำนวณในฟังก์ชันดังกล่าว ลักษณะเช่นนี้ก็ไม่ควรกำหนดค่าของ A, B, C ให้ปรากฏในโปรแกรม

ตัวอย่างที่ 1 การกำหนดค่าของข้อมูลในโปรแกรม เช่น จงหาค่าของ Y จากสมการ $Y = AX + B$ เมื่อ X มีค่าเป็น 1 - 100 โดยที่กำหนดให้ A มีค่าเท่ากับ 1.42 และ B มีค่าเท่ากับ 0.4529 ซึ่งเราจะ เขียนโปรแกรมในภาษา ปาสคาลได้ดังนี้

```
PROGRAM PI(OUTPUT);
VAR
    X          : INTEGER   ;
    A,B,Y      : REAL     ;
BEGIN
    A := 1.42;
    B := 0.4592;
    FOR X:= 1 TO 100 DO
        BEGIN
            Y := A*X + B ;
            WRITELN( 'Y=', Y:6:2)
        END
    END.
END.
```

จากตัวอย่างนี้ จะเห็นได้ว่าค่าของ A,B ถูกสร้างขึ้นภายในโปรแกรม โดยการกำหนด(assign) ในภาษาปาสคาล

ตัวอย่างที่ 2 การทดสอบว่าโปรแกรมคำนวณหาค่าน้ำประปาตามเกณฑ์ต่อไปนี้สามารถปฏิบัติงานจริง โดยวิธีการสร้าง ข้อมูลทดสอบภายในโปรแกรม ตัวอย่างต่อไปนี้จะใช้ภาษาเบสิกในการเขียนโปรแกรม

กติกาคิดค่าน้ำประปามีดังนี้คือ

- ผู้ใช้น้ำประปาที่ใช้น้ำน้อยกว่า 10 หน่วยต่อเดือนจะคิดค่าน้ำประปาต่อหน่วยเดือนละ 2 บาท
- ผู้ใช้น้ำรายใดที่ใช้น้ำมากกว่า 10 หน่วยต่อเดือน แต่ไม่เกิน 51 หน่วย จะคิดค่าน้ำประปาหน่วยที่เกินจากที่เกิน 10 ไปในราคาหน่วยละ 3 บาท
- ผู้ใช้น้ำประปารายใดที่ใช้น้ำมากกว่า 50 หน่วย ขึ้นไปจะคิดหน่วยที่เกิน 50 ขึ้นไป ในราคาหน่วยละ 4 บาท

จากข้อตกลงที่กล่าวมาแล้ว สรุปตัวอย่างได้ว่า

ถ้า นาย ก. ใช้น้ำ 8 หน่วย เขาจะต้องเสียเงินค่าน้ำประปา 16 บาท

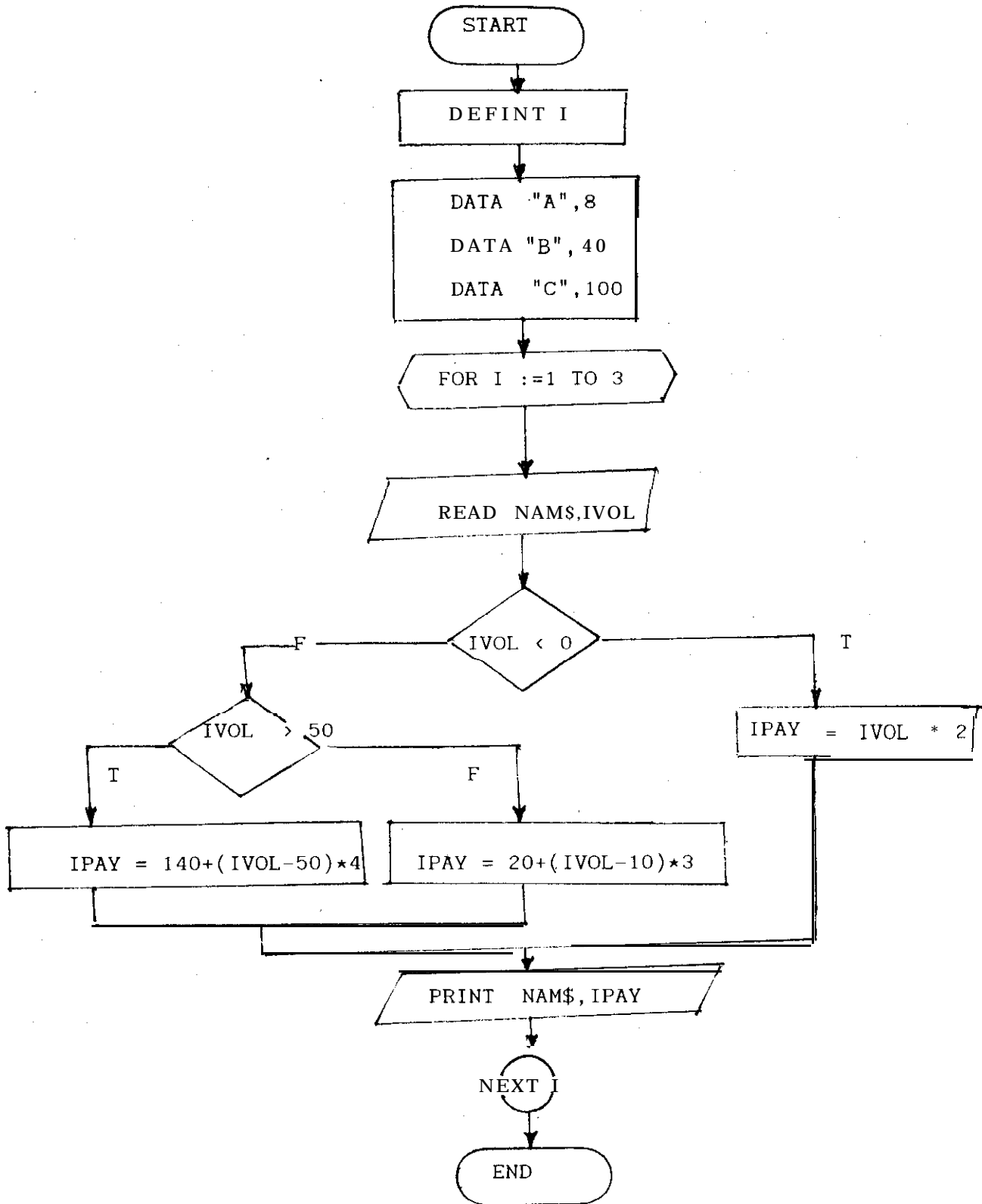
ถ้า นาย ข. ใช้น้ำ 40 หน่วย เขาจะต้องเสียเงินค่าน้ำประปา

$$20 + 90 = 110 \text{ บาท}$$

ถ้า นาย ค. ใช้น้ำ 100 หน่วย เขาจะต้องเสียเงินค่าน้ำประปา

$$20 + 120 + 200 = 340 \text{ บาท}$$

ผังโปรแกรมต่อไปนี้จะแสดงวิธีการทำงาน โดยมีการข้อมูลให้ทดสอบภายในโปรแกรม ทั้งนี้เพื่อที่จะทดสอบว่าโปรแกรมนี้จะสามารถคำนวณได้ถูกต้อง เหมือนที่เราคำนวณด้วยเครื่องคิดเลข ในโปรแกรมนี้อาจจะใช้คำสั่ง DATA เพื่อสร้างข้อมูลทดสอบ



จากผังโปรแกรมดังกล่าว เราสามารถจะเขียนโปรแกรมได้ดังนี้คือ

```

10 CLEAR :CLS: DEFINT I
20 DATA "A",8
30 DATA "B",40
40 DATA "C",100
50 FOR I = 1 TO 3
60     READ NAM$,IVOL
70     IF (IVOL < 10) THEN IPAY = IVOL * 2
80     IF (IVOL < 50 AND IVOL >=10 ) THEN
           IPAY = 20+(IVOL-10)*3
90     IF (IVOL > 50) THEN IPAY = 140+(IVOL-50)*4
100    PRINT "NAME OF CUSTOMER";N$
110    PRINT "VOLUME OF WATER";IVOL
120    PRINT "AMOUNT PAYMENT";IPAY
130 NEXT I
140 END

```

ตัวอย่างโปรแกรมที่ 2 จะประกอบด้วย ข้อมูลทดสอบที่ครอบคลุมทั้ง 3 กรณี ตามเงื่อนไขของข้อตกลง ภายหลังเมื่อทดสอบโปรแกรมว่าให้คำตอบที่ถูกต้องแล้ว เราจะทิ้งข้อมูลที่ทดสอบนี้ออกจากโปรแกรมไป และปรับโปรแกรมให้รับข้อมูลที่เป็น external mass storage file จริงๆแทน

การกำหนดค่าของตัวแปรอีกวิธีที่นิยมใช้กันในภาษาปาสคาลคือการกำหนดเป็น constant แต่จะมีข้อแตกต่างจากการกำหนดวิธีอื่นตรงที่ว่า การกำหนดวิธีนี้จะทำให้ตัวแปรที่กำหนดนั้นถูกจองที่ตั้งแต่ตอนที่ compile program ดังนั้นเราจึงไม่สามารถจะเปลี่ยนแปลงค่านี้อีกในตอนที่ execute program กรรมวิธีที่ว่่านี้นี้เหมาะจะสมกับการกำหนดค่าชนิดคงที่ที่ใช้ตลอดไปในการทำงานเช่น ค่าของ $\pi(x)$ ค่าของตัวเลขที่เล็กที่สุดที่โปรแกรมสามารถเก็บได้ (minimum value) ค่าขนาดของ array ที่ใช้กันในแต่ละครั้ง

ตัวอย่างของการกำหนดค่าของ constant

```
const
    pi      = 3.1415927 ;
    minimum = 0.00014   ;
    bigrow  = 20        ;
```

หลักการในการสร้างตัวแปร constant มีดังนี้คือ

SYMBOLIC CONSTANT

any scalar constant that either :

1. is used frequently within a program
2. could possibly bechange in future

versions of the program
 3. is important to an understanding of
 the logic of a program

ตัวอย่างที่ 3 ตัวอย่างโปรแกรมนี้จะแสดงให้เห็นถึงการใช้วิธีในการสร้าง constant เพื่อช่วยให้การออกแบบโปรแกรมสามารถปรับไปใช้กับ array ที่มีขนาดไม่เท่ากันได้

```

PROGRAM ECPLAIN (INPUT,OUTPUT)
CONST
    BIG = 20 ;
VAR
    MARK : ARRAY[1..BIG] OF INTEGER;
:
BEGIN
:
    FOR I:= 1 to BIG DO
        BEGIN
:
:
:
END.

```

จากตัวอย่างนี้จะเห็นได้ว่าการขยายหรือลดขนาดของ array จะปรับแก้ได้ง่าย เพราะเพียงแต่เราแก้ไขเฉพาะส่วนที่เป็นการสร้าง constant แล้วนำโปรแกรมนั้นไป compile ใหม่ ก็จะสามารถทำให้โปรแกรมนั้นสามารถใช้ได้กับ array ขนาดใดก็ได้ที่คอมพิวเตอร์สามารถรับเข้าไปเก็บได้

นอกจากกรรมวิธีนี้แล้วในภาษาปาสคาลยังสามารถกำหนดข้อมูลในสภาพคล้ายแบบนี้ได้อีกหลายรูปแบบ แต่ในที่นี้จะไม่กล่าวถึง

2. ข้อมูลประเภทที่จะต้องกำหนดให้รับจาก external file เช่นรับจากแป้นพิมพ์ซึ่งเรามักจะเรียกว่า stream file หรือจากสื่อบันทึกข้อมูลชนิดอื่น ๆ นั้น เราจำเป็นต้องมีการวิธีการสร้างตัวแปร ไว้เก็บข้อมูลที่อ่านจากสื่อเหล่านั้นมาไว้ในสมองเครื่องเพื่อที่จะได้นำไปดำเนินการประมวลผลข้อมูลต่อไป ดังนั้นเราจึงจำเป็นต้องทราบและทำความเข้าใจกับองค์ประกอบพื้นฐานเพื่อนำไปใช้ร่วมในการออกแบบโปรแกรมต่อไป พื้นที่ที่ข้อมูลเหล่านี้จะไปพักพิงได้นั้นคือพื้นที่ส่วนที่เราเรียกว่า main storage area นั่นเอง ที่จริงแล้วการเรียกใช้ หรือ จัดเก็บข้อมูล ในพื้นที่ส่วนนี้จริงๆแล้วเครื่องคอมพิวเตอร์จะต้องอ้างอิงโดยใช้ actual address และในลักษณะของการเขียนโปรแกรมในภาษาระดับสูงนั้น เราจะอ้างอิงถึงพื้นที่ดังกล่าว โดยการใช้ตัวแปร (variable) หรือ data name ซึ่งเปรียบเสมือน virtual address นั้นเพื่อความสะดวกและง่ายกับการดำเนินงาน

สิ่งที่ต้องทำความเข้าใจในสาระพื้นฐานเหล่านี้จะประกอบด้วย

1. จะต้องมีการประกาศจองพื้นที่เพื่อให้ข้อมูลพักพิงเสียก่อน แล้วจึงดำเนินการนำข้อมูลเข้าไปพักพิง ในข้อนี้ในบางภาษา บอม ให้มีการประกาศพร้อมกับการเก็บข้อมูลไว้ ณ ที่นั้นโดยอาศัยคำสั่งเดียวกันได้ เช่น ภาษา BASIC ,FORTRAN เป็นต้น ซึ่งถ้าจะกล่าวว่าเป็นข้อดีก็ได้ คือทำให้ผู้ที่ออกแบบโปรแกรมไม่ต้องวางแผนล่วงหน้าว่าจะใช้ตัวแปรใดบ้าง มีขนาดเท่าไร จะใช้พื้นที่สำหรับส่วนของข้อมูลเท่าไร แต่ในทางตรงข้ามข้อเสียที่เกิดขึ้นก็คือไม่มีมีการวางแผนเรื่องการใช้พื้นที่ไว้ทำให้อาจเกิดปัญหาการ overflow ได้ สำหรับในภาษาที่ไม่ยอมให้มีการใช้งานในลักษณะนี้เช่น ภาษา PASCAL,C จะต้องมีการประกาศจองพื้นที่เสียก่อน ภายหลังเมื่อได้รับการจัดสรรแล้วจึงใช้งานได้ต่อไปในการจัดเก็บข้อมูล วิธีนี้ดีในแง่ของการควบคุม และความชัดเจนเรื่องของตัวแปร ในกรณีที่มีผู้อื่นมาใช้โปรแกรมนั้น และในแง่ของการพัฒนาโปรแกรมต่อไปในอนาคต

โดยสรุป ก็คือถึงแม้ว่าในบางภาษาจะยอมให้เราสามารถจะใช้ตัวแปรปริยาย (default variable) เพื่อประกาศจองและเก็บค่าของข้อมูลได้ (ดังตัวอย่าง ในภาษา FORTRAN คำสั่งต่อไปนี้จะ เป็นทั้งประกาศจองและเก็บค่า I ที่เป็นพื้นที่ที่สามารถเก็บเลขจำนวนเต็ม และกำหนดให้เก็บค่า 5 ไว้) เราก็ไม่ควรจะทำ สิ่งที่เราควรปฏิบัตินั้นควรจะใช้วิธีการประกาศแจ้งก่อนที่จะมีการนำไปใช้งานจะดีกว่า

ตัวอย่างของคำสั่งดังกล่าว

I = 5

(ข้อตกลงของภาษา FORTRAN มีว่าตัวแปร หรือ data name ที่มีชื่อขึ้นต้นด้วยตัวอักษรต่อไปนี้ I, J, K, L, M, N จะเป็นตัวแปรปริยายที่สามารถเก็บได้เฉพาะ เลขจำนวนเต็ม)

หมายเหตุ การดำเนินการตามตัวอย่างที่กล่าวมานี้ สามารถใช้ได้เฉพาะตัวแปรประเภทตัวแปรเดี่ยวเท่านั้น จะดำเนินการกับตัวแปรกลุ่ม เช่น array ไม่ได้

2.. ตัวแปรที่มีการประกาศจองนั้นจะมีอยู่ 2 ประเภทคือตัวแปรชนิด static และตัวแปรประเภท dynamic ความแตกต่างของตัวแปรสองประเภทนี้ก็คือพื้นที่ที่ถูกจองแบบ static จะถูกครอบครองตลอดไป ไม่มีการปลดปล่อยไปใช้ในงานอื่น แต่ถ้าเป็นตัวแปรแบบ dynamic เรามีสิทธิ์ที่จะปลดปล่อยจากการครอบครองเพื่อใช้ในภาระกิจอื่นได้ ตัวอย่างของการจองพื้นที่ ที่เป็น static ก็คือการประกาศจอง array ส่วนพื้นที่ที่จัดว่าเป็นการประกาศจองในลักษณะของ dynamic ก็คือการใช้ตัวแปรประเภท pointer ในภาษาปาสคาล

ข้อดีของการใช้ตัวแปรประเภท dynamic ที่เหนือกว่าตัวแปรประเภท static ก็คือ เราใช้พื้นที่ในเครื่องคอมพิวเตอร์ได้ประโยชน์สูงกว่าเพราะมีลักษณะของการ reuse พื้นที่

เพื่อให้นักศึกษาเข้าใจในเรื่องตัวแปรได้ดียิ่งขึ้น ขอให้อ่านจากหัวข้อในเรื่อง วิธีการจัดการข้อมูล ภายในสมองของเครื่องคอมพิวเตอร์

ความหมายและการจัดเก็บข้อมูลภายในเครื่องคอมพิวเตอร์

ดังที่ได้กล่าวมาแล้วว่าการจัดเก็บข้อมูลภายในเครื่องคอมพิวเตอร์ ในส่วนของหน่วยความจำหลัก ซึ่งไม่ใช่หน่วยความจำถาวรไว้ใช้ได้ตลอดเวลา แต่เก็บไว้ใช้ได้เฉพาะในช่วงเวลาของการประมวลผลเท่านั้น และด้วยเหตุผลที่เราจะต้องมีการจัดสรรพื้นที่ส่วนนี้ให้ใช้ประโยชน์ได้อย่างมีประสิทธิภาพ เราจึงต้องมีกาวางแผนและระวอกแบบเพื่อที่จะสร้างที่เก็บข้อมูลที่เกี่ยวข้องกับการใช้งาน พื้นที่ที่เป็นที่อยู่นี้เราเรียกกันทั่วไปว่า ตัวแปร (variable) หรือ data name

เราอาจแบ่งตัวแปรที่เก็บข้อมูลออกมาได้เป็น 2 ประเภทดังนี้

1. ตัวแปรเดี่ยว

2. ตัวแปรกลุ่ม

1. ตัวแปรเดี่ยว หมายถึงพื้นที่ที่เก็บข้อมูลซึ่งแต่ละตัวจะ ไม่มีความสัมพันธ์กันเลย

ตัวอย่างเช่น การประกาศของตัวแปรในภาษาปาสคาล และภาษาซีดังนี้

ภาษาปาสคาล

```
VAR A,B : INTEGER ;
```

```
X : REAL ;
```

ภาษาซี

```
int a,b ;
```

```
float x ;
```

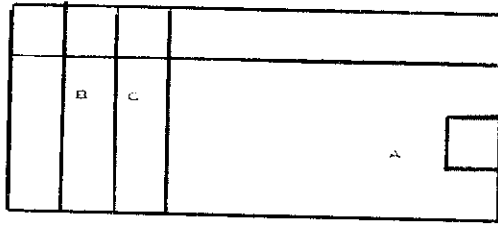
คำประกาศของทั้ง 2 ภาษาจะก่อให้เกิดการจับจองพื้นที่ในรูปแบบใดก็ได้ เช่น

ดังตัวอย่างที่ปรากฏต่อไปนี้

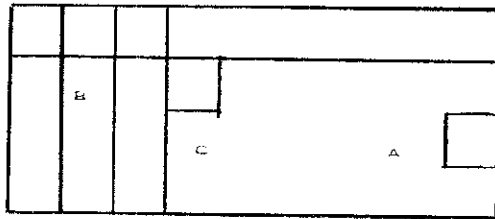
แบบที่ 1

A	B	C	

แบบที่ 2



แบบที่ 3



สมมติว่า บนข้อตกลงดังกล่าวได้รับการจัดสรรแล้วให้พื้นที่ A,B มีขนาดตัวแปรละ 2 byte ส่วนตัวแปร C มีขนาด 4 byte ดังนั้นเครื่องคอมพิวเตอร์จะจัดสรรให้พื้นที่ใดๆที่ว่างๆให้ตามที่โปรแกรมร้องขอ

ตัวแปรเดี่ยวที่กำลังกล่าวถึงนี้จะสามารถนำไปจำแนกย่อยลงไปอีกได้ดังนี้คือ

1.1 ตัวแปรแบบ Numeric หมายถึงตัวแปรที่สามารถเก็บข้อมูลที่นำไปประเมินผลทางคณิตศาสตร์ (นำไปประมวลผลกับเครื่องหมายทางคณิตศาสตร์ *,/,+,-) ได้ ซึ่งเราจะจำแนกเป็นประเภทย่อยๆได้ดังนี้คือ

1.1.1 ตัวแปรชนิดจำนวนเต็ม (Integer)

จะหมายถึงตัวแปรที่เก็บได้เฉพาะจำนวนเต็มซึ่งอาจจะมีค่าเป็นบวก หรือ ลบ ก็ได้ วิธีการเก็บค่าของข้อมูลที่มีค่าเป็น บวก หรือ ลบ ดำเนินการได้ดังตัวอย่างนี้คือ ประกาศให้ X เป็นตัวแปรจำนวนเต็ม สมมติว่าเครื่องคอมพิวเตอร์จัดสรรให้ 2 byte

และ 1 byte มีขนาดเท่ากับ 8 bits ดังนั้น พื้นที่ที่ตัวแปร X ได้รับการจัดสรรก็คือ 16 bits ซึ่งจะมีความสามารถในการเก็บข้อมูลขนาดดังนี้คือ -2^{15} ถึง $+2^{15}-1$ ทั้งนี้ เนื่องจาก 1 byte เท่ากับ 8 bits ดังนั้น 2 bytes จะมีความเท่ากับ 16 bits ซึ่งในพื้นที่ดังกล่าวจะสามารถแสดงผลข้อมูลได้ถึง 16^2 แต่เนื่องจากพื้นที่นี้จะต้องแบ่งที่เพื่อเก็บข้อมูลทั้งค่าของ บวก และ ลบ ดังนั้นค่าด้านลบและบวกที่สามารถแสดงได้ก็คือ $2^{16}/2 = 2^{15}$ ดังนั้นค่าของเลขลบที่สามารถแสดงได้ก็คือ $-1, -2, \dots, -2^{15}$ และค่าของเลขบวกที่สามารถแสดงได้ก็คือ $0, 1, 2, \dots, 2^{15}-1$

1.1.2 ตัวแปรแบบ long integer

จะมีความหมายเหมือนกับตัวแปรแบบข้อ 1.1 จะแตกต่างกันก็ตรงที่พื้นที่ที่ได้รับการจัดสรรจะมีขนาดถึง 4 bytes ซึ่งส่งผลให้สามารถเก็บบันทึกค่าของข้อมูล ได้น้อย และมากกว่าแบบที่ 1.1

1.1.3 ตัวแปรแบบ เลขจำนวนจริง (real)

จะเป็นพื้นที่ที่มีความสามารถในการจัดเก็บข้อมูลได้ทั้งส่วนของ เลขจำนวนเต็มและ เศษและ ได้ทั้งค่าบวกและค่าลบ

1.1.4 ตัวแปรแบบ double

ในกรณีที่บางภาษายอมให้มีการขยายขนาดของตัวแปรแบบ real ให้มีขนาดใหญ่ขึ้นเพื่อใหขนาดของทศนิยมสามารถเก็บได้ถูกต้องมากยิ่งขึ้น ก็จะทำให้พื้นที่ที่ได้รับการจัดสรรโดยตัวแปรชนิดนี้มีขนาดเท่ากับ 8 bytes

ต่อไปนี้จะ เป็นตัวดำเนินการทางคณิตศาสตร์ (numeric operator) ที่อนุญาตให้ใช้กับตัวแปรประเภท numeric ทั้งหมดที่กล่าวมาแล้ว (ใช้ภาษาปาสคาลเป็นตัวอย่าง)

+ addition

- substitution (or unary negation)

* multiplication

div integer division ตัวอย่างเช่น $19 \text{ div } 7 = 2$

mod modulus เช่น $a \text{ mod } b$ เช่น ตัวอย่างเช่น $8 \text{ mod } 3 = 2$

ตัวดำเนินการ relational operator

> grater than
 >= greater than or equal to
 < greater than
 <= less than or equal
 = equaal
 <> not equal

นอกเหนือจากนี้ในภาษาปาสคาล ยังกำหนดให้มีมาตรฐาน (standard function) ที่จะใช้กับ integer ดังตัวอย่างเช่น

abs(i) : the absolute value of integer i

sqr(i) : i square

trunc(r) : r is a decimal number the result is the integer portion of r ตัวอย่าง เช่น trunc(2.6) จะได้มีค่าเท่ากับ 2

round(r) : same as trunc but r is rounded to the nearest integer ตัวอย่างเช่น round(2.6) = 3

การกำหนดค่าให้กับตัวแปรแบบ real สามารถกำหนดได้ดังนี้

3.1415927

-26.0

+0.198

5e6 (มีค่าเท่ากับ $5 \times 10^6 = 5,000,000$)

-6e-8 (มีค่าเท่ากับ $-6 \times 10^{-8} = -6 \times (1/10^8 = -0.00000006)$)

+6.08 e +27 มีค่าเท่ากับ 6.08 e +27 มีค่าเท่ากับ 6.08×10^{27}

นอกเหนือจาก numeric operator ที่กล่าวมาแล้ว ยังมี function ที่สามารถนำมาใช้กับค่าของ real ในภาษาปาสคาลอีกอาทิเช่น

abs(r)	จะให้ค่าของเลขสัมบูรณ์ของ r (r อาจเป็น real หรือ integer ก็ได้)
sqr(r)	จะให้ค่า r ยกกำลังสอง (r อาจเป็น real หรือ integer ก็ได้)
sin(r)	จะให้ค่า sin ของ r (r อาจเป็น real หรือ integer ก็ได้)
cos(r)	จะให้ค่า cos ของ r (r อาจเป็น real หรือ integer ก็ได้)
arctan(r)	จะให้ค่า arctan ของ r (r อาจเป็น real หรือ integer ก็ได้)
ln(r)	จะให้ค่า log ฐาน e ของ r (r อาจเป็น real หรือ integer ก็ได้)
exp(r)	จะให้ค่า exponential ของ r (r อาจเป็น real หรือ integer ก็ได้)
sqrt(r)	จะให้ค่า รากที่สองของ r (r อาจเป็น real หรือ integer ก็ได้)

ความถูกต้องของ เลขจำนวนจริงที่เก็บในคอมพิวเตอร์

การที่จะเก็บขนาดของตัวเลขจำนวนจริงไว้ภายในสมองของเครื่องคอมพิวเตอร์ จะขึ้นอยู่กับขนาดของหน่วยความจำของตัวแปรที่กำหนด ซึ่งจะมีขนาดแตกต่างกันไปตามระบบคอมพิวเตอร์ ให้พิจารณาจากตัวอย่างต่อไปนี้

$$\text{ค่าของ } 1.0/3.0 = 0.333333\dots$$

ซึ่งเครื่องคอมพิวเตอร์จะตัดตอนข้อมูล (truncate) ขนาดที่จะเก็บในตัวแปรได้ ผลกระทบที่เกิดขึ้นในกรณีของการตัดตอนนี้ก็คือ

- เกิดมีความผิดพลาดในตัวเลขหลักทศนิยมท้ายๆ ในกรณีของการนำค่าดังกล่าวไปคำนวณมากครั้งก็ยิ่งจะเกิดการปัดเศษมากขึ้น ซึ่งทำให้เกิดความผิดพลาดยิ่งสูงขึ้น
- จากผลที่เกิดขึ้นดังที่ได้กล่าวมาแล้ว เราจึงจำเป็นที่จะต้องทำความเข้าใจว่าการตรวจสอบนิพจน์คณิตศาสตร์บางประเภท จะต้องทราบว่าหลักการปฏิบัติงานกับข้อมูลที่เกิดขึ้นในระบบคอมพิวเตอร์เป็นเช่นใด ดังตัวอย่างต่อไปนี้

$1.0/3.0 + 1.0/3.0 + 1.0/3.0$ จะมีค่าไม่เท่ากับ 1 ถ้าเราให้คอมพิวเตอร์ดำเนินการ ดังนั้นการเขียนโปรแกรมเพื่อตอบคำถามนี้จึงต้องยอมให้มีการผิดพลาดได้ในระดับหนึ่งที่กำหนดไว้ ดังวิธีการใช้คำสั่งดังนี้

```
if abs(x-y) <= ε then write 'x is equal y'
```

โดยที่ ϵ คือค่าที่ยอมให้ผิดพลาดได้

1.2 ตัวแปรแบบอักขระ (Character) หมายถึงตัวแปรที่สามารถเก็บได้เพียงตัวอักขระเดียว อักขระที่ใช้กับการทำงานในระบบคอมพิวเตอร์ อักขระที่ใช้เราแบ่งออกเป็น 2 กลุ่มคือ

กลุ่มที่ 1 เป็นรหัสอักขระที่ใช้ควบคุมระบบ จะเป็นประเภท unseen character ตัวอย่างเช่น FF ,BS,LF,CR เป็นต้น

กลุ่มที่ 2 เป็นอักขระที่ใช้กันในการสร้างข้อมูลหรือโปรแกรม เราแบ่งออกเป็น 3 กลุ่มย่อยดังนี้คือ

2.1 อักขระที่เป็นตัวเลข 0-9

2.2 อักขระที่เป็นตัวอักษรภาษาอังกฤษ A-Z, a-z

2.3 อักขระพิเศษ เช่น *,%,/,# เป็นต้น รวมทั้งอักขระภาษา

ไทยด้วยในกรณีที่ใช้ภาษาไทยร่วมด้วย

ระบบอักขระที่ใช้กันบนเครื่องคอมพิวเตอร์ มีทั้งสิ้น 256 ตัว แต่ละภาษาโปรแกรมที่เขียนมักจะมีฟังก์ชันที่ใช้สำหรับพวกอักขระอยู่มากมาย จะขอยกตัวอย่าง character function ที่ใช้ในภาษาปาสคาลที่ใช้บ่อยๆดังนี้คือ

`ord(c)` : ordinal function ความหมายก็คือฟังก์ชันดังกล่าวจะให้คำตอบว่า
ascii code ที่เก็บไว้ในเครื่องใช้รหัสอะไร เช่น `ord('L')`
จะให้คำตอบคือ 76

`chr(i)` : character function โดยที่ `i` จะหมายถึงค่า
ของเลขจำนวนเต็มบวก ฟังก์ชันนี้จะให้คำตอบว่าเลขที่เราระบุนั้น
ตรงกับ ascii code ใด

`pred(c)` : predecessor function โดยที่ `c` คือรหัสอักขระใด
ฟังก์ชันนี้จะให้คำตอบว่าตัวอักขระก่อนหน้าตัว '`c`' ที่ระบุคืออะไร
ดังนั้น `pred(c)` จะให้ผลเท่ากับ `chr(ord(c)-1)`

`succ(c)`: succesor function โดยที่ `c` คือ character คำตอบที่ได้
จากฟังก์ชันนี้ก็คือ next character ของ `c` ดังนั้น `succ(c)`
จะให้ผลเท่ากับ `chr((ord(c)+1))` ตัวอย่าง เช่น
`pred('B') = A` และ `succ('B') = C`

1.3 ตัวแปรแบบ Boolean ค่าของตัวแปรแบบนี้จะปรากฏได้เพียง 2 ลักษณะคือ
`true` หรือ `false` เท่านั้น โดยที่ `false < true`

สัญลักษณ์ของ relational operator ที่ใช้กับตัวแปรประเภทนี้
จะปรากฏดังนี้

`and` คือ logical conjunction (binary operator)

`or` คือ logical disjunction (binary operator)

`not` คือ logical negation (unary operator)

ตัวอย่างของการใช้ตัวแปรแบบ boolean ในภาษาปาสคาล

```
var
    T :boolean ;
    :
begin
    T := (A =10) OR (B>5)
```

ดังนั้น ถ้า A = 9 และ B = 6 ดังนั้นคำตอบของ T ก็คือ true

เราจะเรียก (A =10) OR (B>5) ว่า logical expression ตารางต่อไปนี้จะแสดง
วิธีการใช้ boolean operator

p	q	p and q	p or q	not p
T	T	T	T	F
T	F	F	T	F
F	T	F	T	T
F	F	F	F	T

2. ตัวแปรแบบกลุ่ม หมายถึงพื้นที่ที่จัดจงบเป็นกลุ่มไว้กับข้อมูลพักพิงอยู่ ซึ่งในลักษณะนี้จะแตกต่างจาก ตัวแปรเดี่ยวที่กล่าวมาแล้วในแง่ที่ว่า จะต้องมีการจัดสรรพื้นที่ที่ต่อเนื่องกันเพื่อให้ข้อมูลกลุ่มอยู่ และภายในกลุ่มก็จะแบ่งเป็นพื้นที่ย่อยๆให้ ตัวแปรสมาชิกอยู่ตัวแปรกลุ่มที่กล่าวถึงนี้จะจำแนกคร่าว ๆตามลักษณะของภาษาโปรแกรมที่ใช้กันอยู่ทั่วไป อยู่ 2 ประเภท คือ

2.1 Array

2.2 Record

2.1 Array แบ่งออกเป็น 2 ประเภทย่อยคือ

2.1.1 Array ประเภทมิติเดียว (One Dimension)

2.1.2 Array ประเภทหลายมิติ (Multi Dimesion)

2.1.1 Array มิติเดียว (One Dimension)

เพื่อความสะดวกและง่ายแก่การทำความเข้าใจ จะขอปูพื้นฐานเริ่มตั้งแต่ Array มิติเดียวก่อน โดยการเปรียบเทียบแนวคิดความรู้จากเรื่องของตัวแปรเดี่ยวก่อนดังนี้คือ สมมุติว่าเราต้องการที่จะนำคะแนนสอบวิชาคณิตศาสตร์ของนักเรียน 50 คน มาหาคะแนนเฉลี่ยแล้ว นำมาตัดสินว่าถ้าใครได้คะแนนสอบสูงกว่าคะแนนเฉลี่ยก็จะสอบผ่าน แต่ถ้าได้คะแนนสอบน้อยกว่าคะแนนสอบเฉลี่ยก็จะสอบไม่ผ่าน ในลักษณะของข้อตกลงที่กล่าวมาแล้วถ้าเราออกแบบการจัดการเรื่องข้อมูล โดยใช้ตัวแปรเดี่ยวในการเก็บข้อมูล แล้วเราจะต้องใช้ประกาศสร้างตัวเดี่ยวเพื่อเก็บคะแนนสอบถึง 50 ที่ และ อีก 50 ที่เพื่อเป็นเก็บรหัสประจำตัวของนักเรียน เราลองวาดภาพถึงสิ่งที่เกิดขึ้นในการจัดสรรและบริหารตัวแปรที่เก็บข้อมูลว่าจะยุ่งยากเพียงใด ไม่เพียงแต่เท่านั้นการนำตัวแปรดังกล่าวไปทำการประมวลผลหาคำตอบที่เราต้องการก็จะทำด้วยความลำบาก หนทางหนึ่งสำหรับช่วยการแก้ปัญหาในงานนี้ก็คือการจัดโครงสร้างของข้อมูลโดยใช้รูปแบบของ Array แล้วใช้ประโยชน์จากคำสั่ง FOR หรือ WHILE ที่มีอยู่ในเกือบทุกภาษาโปรแกรมแก้ปัญหาที่เกิดขึ้น ดังนั้นในปัญหานี้เราจะดำเนินการดังนี้คือ สร้างตัวแปร Array ซึ่งจะประกอบด้วย 50 สมาชิกย่อยโดยที่แต่ละสมาชิกย่อยจะเป็นตัวแปรประเภท Integer และ สร้างตัวแปรอีก 1 Array ซึ่ง Array นี้จะใช้เก็บรหัสของนักเรียนดังนั้นจึงมี 50 สมาชิกย่อยโดยที่แต่ละสมาชิกย่อยจะเป็นตัวแปรประเภท Integer (สมมุติว่ารหัสนักเรียนไม่มีค่าใหญ่มากเกินไป) คำสั่งของการประกาศจงบและดำเนินการประมวลผลหาคำตอบที่ต้องการ ในงานนี้จะเขียนออกมาได้ดังนี้คือ

```

PROGRAM REPORT(INPUT,OUTPUT) ;
TYPE
    X      = ARRAY [1..50] OF INTEGER;
VAR
    ID     : X;
    MARK   : X;
    I,SUM   : INTEGER ;
    AVER    : INTEGER ;
    GRADE  : CHAR ;
BEGIN
    SUM :=0;
    FOR I:= 1 TO 50 DO
        BEGIN
            WRITE ('ID AND MARK OF STUDENT NO.',I);
            READLN(ID[I],MARK[I]);
            SUM := SUM + MARK[I]
        END;
    AVER := SUM DIV 50 ;
    FOR I:= 1 TO 50 DO
        GRADE := 'P'
        BEGIN
            IF (MARK[I] < AVER ) THEN GRADE := 'F';
            WRITE (ID[I],MARK[I],GRADE)
        END
    END
END.

```

```

1
2
3

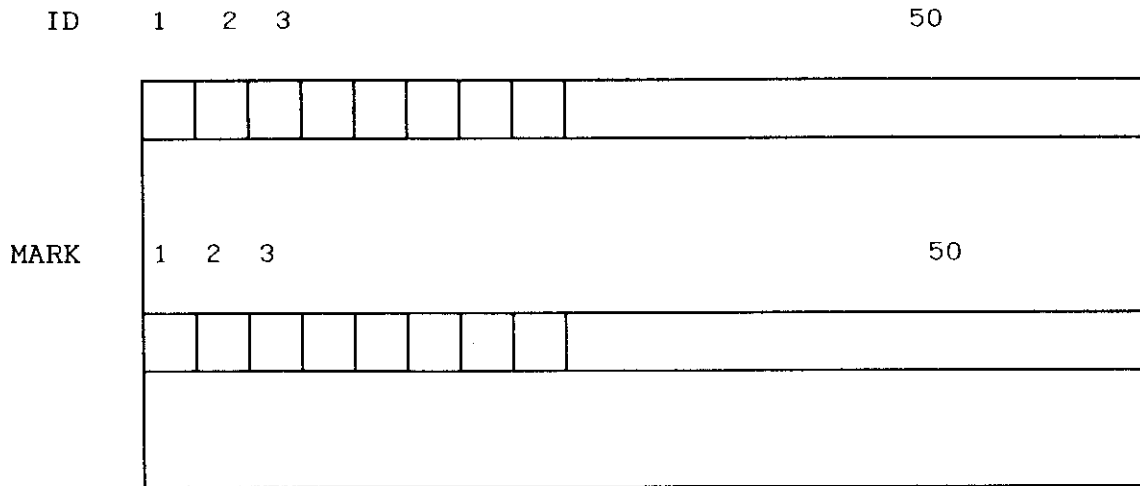
```

```

4

```

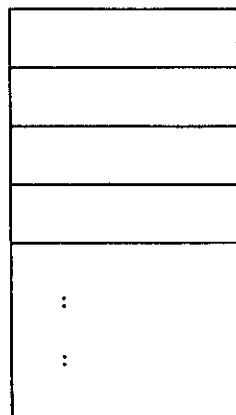
โครงสร้างของข้อมูลที่ได้รับการจัดสรรภายในคอมพิวเตอร์จะมีรูปแบบดังนี้



ในกรณีที่ต้องการอธิบายถึงโครงสร้างทั่วไปในการทำความเข้าใจถึงวิธีการเก็บข้อมูล เรามักจะใช้รูปนี้แทน เพราะเข้าใจได้ง่ายกว่า

Organization of Array

Individual
element



entire Array

ตัวอย่างคำสั่งในภาษาปาสคาลเพื่อประกาศจับจองพื้นที่สำหรับ Array

ตัวอย่างที่ 1

```
type sample = array [1..100] of integer;
var
    x :sample ;
```

ผลที่เกิดขึ้นก็คือมีการประกาศจับจองพื้นที่ให้กับตัวแปร x ซึ่งเป็นชนิดตัวแปรที่เก็บได้เฉพาะตัวเลขจำนวนเต็มจำนวน 100 ตัว โดยที่สมาชิกในกลุ่มนี้จะมีการอ้างอิงถึงที่ในนามของ $x[1], x[2], \dots, x[100]$

ตัวอย่างที่ 2

```
type sand = array [-5..5] of char ;
var y : sand ;
```

ดังนั้น ตัวแปร y จะถูกจัดสรรให้มีพื้นที่เท่ากับ 11 byte แต่ละ byte จะเก็บได้ 1 อักขระ โดยที่พื้นที่ย่อยหรือที่เรียกว่าสมาชิกย่อยนั้นจะถูกเรียกว่า $y[-5], y[-4], y[-3], \dots, y[0], \dots, y[2], y[1]$ มีทั้งสิ้น 11 ตัว หมายเหตุการประกาศวิธีที่ ตัวชี้ (index) มีสิทธิ์ที่จะเป็นค่าน้อยกว่า 0 ได้ นั้น ในบางภาษาไม่อนุญาตให้ใช้ แต่กฎที่เหมือนกันทุกภาษาก็คือ ห้ามใช้ตัวชี้เป็นเลขทศนิยม

ลักษณะการจัดเก็บข้อมูลในตัวอย่างนี้จะปรากฏเช่นนี้

h	e	l	l	o		c	a	t	h	y
---	---	---	---	---	--	---	---	---	---	---

ตัวอย่างที่ 3

```

type
    word = array [1..7] of char;
var
    name : array [1..5] of word

```

ตัวอย่างนี้จะเห็นความซับซ้อนของการจับจอง Array ในแง่ที่ว่า name นั้นเป็น Array of Array ซึ่งเราอาจถือว่าเป็น กรณีของ Array 2 มิติ ซึ่งอาจจะเขียนคำสั่งเป็นรูปแบบของการจับจอง Array 2 มิติได้ดังนี้

```

type
    word = array [1..5,1..7] of char;
var
    name : word ;

```

ลักษณะการจัดสรรพื้นที่ในตัวอย่างที่ 3 นี้ จะปรากฏดังนี้

	1	2	3	4	5	6	7
name 1							
name 2							
name 3							
name 4							
name 5							

2.1.2 Array หลายมิติ (Multi Dimension)

จากตัวอย่างที่ 3 จะเห็นได้ว่าการใช้งานในลักษณะของ Array ประเภทมิติเดียว ถึงแม้จะมีประโยชน์มากก็ตาม แต่ในบางครั้งก็ยังมีข้อจำกัดในการใช้งานบางอย่างอยู่ เช่นกรณีของการนำไปประยุกต์ในการจัดเก็บข้อมูลที่มีการให้ความหมาย ในการจำแนกตามลักษณะของข้อมูล ให้พิจารณาจากตัวอย่างต่อไปนี้

ตัวอย่างที่ 1 สินค้าของบริษัทแห่งหนึ่งจัดเก็บตาม ที่ตั้งของคลังสินค้า และ ยังจำแนกตามประเภทของสินค้าอีกด้วย โดยมีรูปแบบการจัดเก็บดังตารางนี้คือ

ที่ตั้งคลังสินค้า	ประเภทของสินค้า			
	ข้าวโพด	น้ำตาล	ข้าว	ถั่วเหลือง
ภาคกลาง	231	459	541	654
ภาคตะวันออก	432	100	387	69
ภาคเหนือ	176	0	250	765
ภาคใต้	621	376	765	437

หน่วย : ตัน

การออกแบบโครงสร้างในการจัดการข้อมูลตามตัวอย่างนี้ วิธีที่ดีก็คือการใช้รูปแบบของ Array 2 มิติ โดยให้ มิติแรก(row) หมายถึงที่ตั้งของคลังสินค้า และ มิติที่ 2 (coloumn) แทนความหมายของ ประเภทของสินค้า ซึ่งโดยรูปแบบที่ออกแบบนี้ จะช่วยให้เราสามารถจะประยุกต์ไปสู่การสร้าง "ข้อเสนอเพื่อสนับสนุนการตัดสินใจของระบบ" ได้หลากหลายด้วยกัน เช่น

- มีสินค้าประเภทใด ที่คลังสินค้าใดที่ไม่มีสินค้าจำหน่ายแล้ว
- การทำงานคลังหลายปี มีมูลค่าของสินค้าเหลืออยู่เท่าไร
- ถ้าราคาน้ำตาลเพิ่มขึ้นตันละ 1500 บาท บริษัทจะได้กำไรเพิ่มขึ้นเท่าไร
- ถ้าเราเอาสินค้าของเราไปประกันภัยไว้ โดยที่บริษัทจะคิดราคาประกันแตกต่างกันไปตาม ที่ตั้งคลังสินค้า และชนิดของราคาสินค้าแล้ว บริษัทจะต้องเสีย ค่าเบี้ยประกันรวมทั้งสิ้น เท่าไร

คำถามที่ ยกมาเป็นตัวอย่างนี้คงจะช่วยให้ผู้อ่านนึกถึง การจัดโครงสร้างข้อมูลเพื่อสะดวกในการหาข้อเสนอที่ต้องการดังกล่าวได้

ตัวอย่างที่ 2 ต้องการที่จะเก็บรวบรวมข้อมูลของนักเรียนจำนวน 100 คนโดยที่แต่ละคนที่เรียนวิชาคณิตศาสตร์ โดยที่ การวัดผลวิชานี้จะแบ่ง คะแนนออกเป็น สามส่วน แต่ละส่วนอาจจะมีน้ำหนักเท่าๆกัน หรือไม่เท่ากันก็ได้ โดยที่คะแนนแต่ละส่วนคือ

- คะแนนการบ้าน
- คะแนนสอบกลางภาค
- คะแนนสอบปลายภาค

นักเรียนคนที่	คะแนนแต่ละประเภท		
	คะแนนการบ้าน	คะแนนสอบกลางภาค	คะแนนสอบปลายภาค
1	67	82	69
2	43	65	72
100	56	76	74

จาก ตารางในตัวอย่างที่ 2 นี้จะเห็นได้ว่า ข้อสนเทศที่ต้องการนำไปตัดสินใจในเรื่องของการศึกษาก็คือ

- อยากทราบว่า คะแนนรวมที่สูงสุด ต่ำสุด และ คะแนนรวมเฉลี่ย มีค่าเท่าไร เพื่อที่เราจะได้นำไป กำหนดเกณฑ์ในการวัดผล ของนักเรียนกลุ่มนี้
- ถ้าเรากำหนดเกณฑ์ในการวัดผลได้แล้ว นักเรียนแต่ละคนจะมีผู้ใดที่จะผ่านเกณฑ์ดังกล่าวได้ และจำนวนผู้ที่ผ่านการประเมิน จะคิดเป็นร้อยละเท่าไร

จากตัวอย่างนี้ ลักษณะของโครงสร้างข้อมูลที่เกิดขึ้นจะอยู่ในรูปแบบที่ กำหนดให้ มิติที่ 1 (row) ก็คือลำดับของนักเรียน ในขณะที่มิติที่ 2 (column) นั้นหมายถึง คะแนนสอบในแต่ละส่วน ดังนั้น ในการประกาศสร้าง Array ในตัวอย่างนี้ที่เหมาะสมก็คือ ประกาศดังนี้

```
var mark : array [1..100,1..3] of integer;
```

ตัวอย่างที่ 3 ต้องการที่จะวิจัยการตลาดสินค้าใน Production "A" จากลูกค้า เพื่อนำข้อมูลมาประกอบการตัดสินใจเพื่อเลือก กลยุทธ์ ในการส่งเสริมการขาย ดังนั้น จึงมีการสุ่มตัวอย่างลูกค้าที่บริโภคสินค้ามาจำนวนหนึ่ง แล้วตามความคิดเห็นเรื่องเกี่ยวกับ "Production "A" ข้อมูลส่วนหนึ่งที่ตาม จะประกอบด้วย

เพศ	<input type="checkbox"/>	ชาย	<input type="checkbox"/>	หญิง
อายุ	ปี		
ความรู้สึกต่อการใช้สินค้า "A" แล้ว			<input type="checkbox"/>	ชอบ
			<input type="checkbox"/>	ไม่ชอบ
			<input type="checkbox"/>	เฉยๆ

จากข้อมูลดังกล่าว สมมติว่าผู้จัดการฝ่ายขายต้องการที่จะได้ข้อสนเทศเพื่อสนับสนุนการตัดสินใจดังนี้คือ

- ตารางจำแนกตามเพศ และความรู้สึกต่อการใช้สินค้า
- ถ้าเราแบ่งกลุ่มลูกค้าตามอายุ คือ
 - กลุ่มที่ 1 อายุต่ำกว่า 16 ปี
 - กลุ่มที่ 2 อายุตั้งแต่ 16 - 30 ปี
 - กลุ่มที่ 3 อายุตั้งแต่ 30 ปี ขึ้นไป

จงสร้างตารางเพื่อจำแนกตามกลุ่มอายุ และความรู้สึกต่อ การใช้สินค้า
Product "A"

ตัวอย่างที่ 3 จะแตกต่างจาก ตัวอย่างที่ 1 และที่ 2 ในแง่ที่ว่า ตัวอย่างที่ 1 และ 2 จะกำหนดให้ข้อมูลที่รับจาก external file นั้น เข้าไปเก็บใน สมองเครื่อง ตามรูปแบบของ Array ประเภท 2 มิติ ในขณะที่ ตัวอย่างที่ 3 นั้นเราไม่จำเป็นต้องเก็บข้อมูลของลูกค้าทุกคนไว้ในรูปของ Array เนื่องจากเราต้องการสารสนเทศ ในรูปของตารางสรุปดังที่กล่าวมาแล้วข้างต้นเพื่อใช้งานเท่านั้น และเหตุผลอีกประการหนึ่งก็คือ ถ้าเรามีข้อมูลลูกค้าจำนวนเป็นเป็นหมื่นคน เราก็ไม่สามารถจัดสรรพื้นที่ให้เพียงพอกับความ ต้องการได้ ดังนั้นโครงสร้างข้อมูลที่เราจะออกแบบให้กับระบบงานนี้ก็คือ ให้รับข้อมูลเข้าไปเก็บ โดยใช้ตัวแปรอิสระธรรมดา เช่น อ่านข้อมูลของลูกค้า ทีละคนแล้วให้ไปเก็บที่ ตัวแปรชื่อ sex, age, per โดยที่เราออกแบบรหัสเพื่อความเหมาะสมดังนี้คือ

```
sex   :    1: ชาย
        2:   หญิง

age   :    integer 2  หลัก

per   :    1:  ชอบ
        2:  ไม่ชอบ
        3:  เฉยๆ
```

ภายหลังเมื่อเรารับข้อมูลเข้าไปเก็บในตัวแปรดังกล่าวแล้วจึงนำไปเก็บในตารางที่ ออกแบบ ในตัวอย่างนี้เราจะออกแบบตารางทั้งสองดังนี้

ตารางที่ 1 จำแนกตามเพศ และความรู้สึกต่อสินค้าของลูกค้า

เพศ	ความรู้สึกต่อสินค้าของลูกค้า		
	ชอบ	ไม่ชอบ	เฉยๆ
ชาย 1	xx	xx	xx
หญิง 2	xx	xx	xx

ตารางที่ 1 นี้เทียบได้เท่ากับ Array ประเภท 2 มิติ โดยมีมิติแรกคือ sex มิติที่สองคือความรู้สึกต่อการบริโภคสินค้า

ตารางที่ 2 จำแนกตามอายุ และความรู้สึกต่อสินค้าของลูกค้า

กลุ่มอายุ	ความรู้สึกต่อสินค้าของลูกค้า		
	ชอบ	ไม่ชอบ	เฉยๆ
1	xx	xx	xx
2	xx	xx	xx
3	xx	xx	xx

ตารางที่ 2 นี้เทียบได้เท่ากับ Array ประเภท 2 มิติ โดยมีมิติแรกคือกลุ่มอายุ มีสองคือความรู้สึกต่อการบริโภคสินค้า

ดังนั้นคำสั่งในการประกาศของพื้นที่ให้กับตารางทั้งสองใน เครื่องคอมพิวเตอร์ก็คือ

```

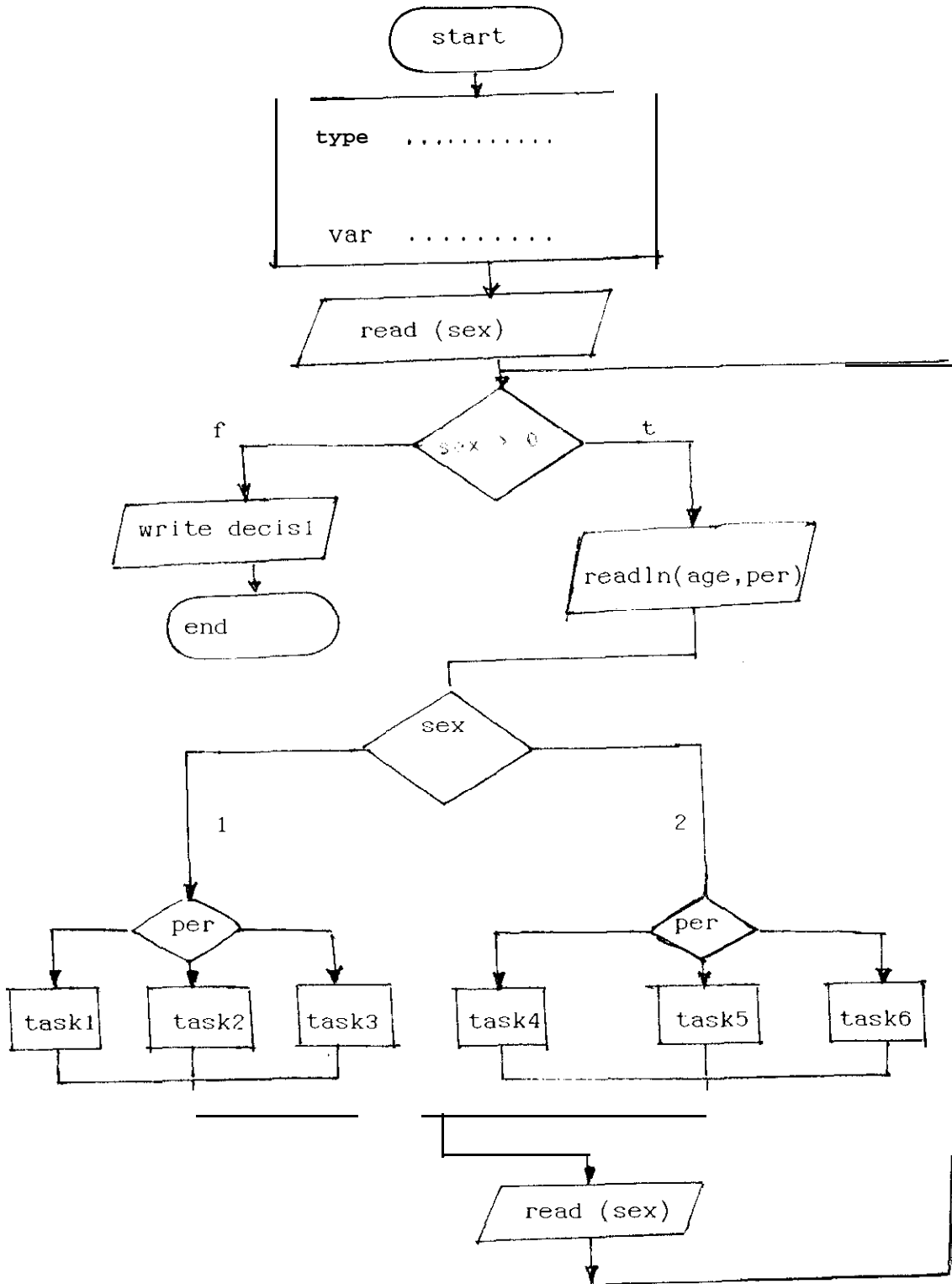
type
    tab1 = array [1..2,1..3] of integer;
    tab2 = array [1..3,1..3] of integer ;
var
    decis1 : tab1;
    decis2 : tab2;

```

เพื่อที่จะชี้ประโยชน์ของการออกแบบรหัส ให้สอดคล้องกับการใช้ Array และการใช้คำสั่งใน FOR Loop จะเขียน Program Flow เพื่อตอบปัญหาเฉพาะในการสร้างตารางที่ 1 ให้ดู ส่วนตารางที่ 2 จะขอเว้นไว้ให้เป็นแบบฝึกหัดของนักศึกษา

การออกแบบ Program Flow ในงานนี้จะทำให้ดู 2 แบบ แล้วให้พิจารณาว่าแบบใดมีข้อดี ข้อเสียแตกต่างกัน

แบบที่ 1



โดยที่ task แต่ละอันมีความหมายดังนี้คือ

task1 : $\text{decis1}[1,1] := \text{decis1}[1,1]+1$

task2 : $\text{decis1}[1,2] := \text{decis1}[1,2]+1$

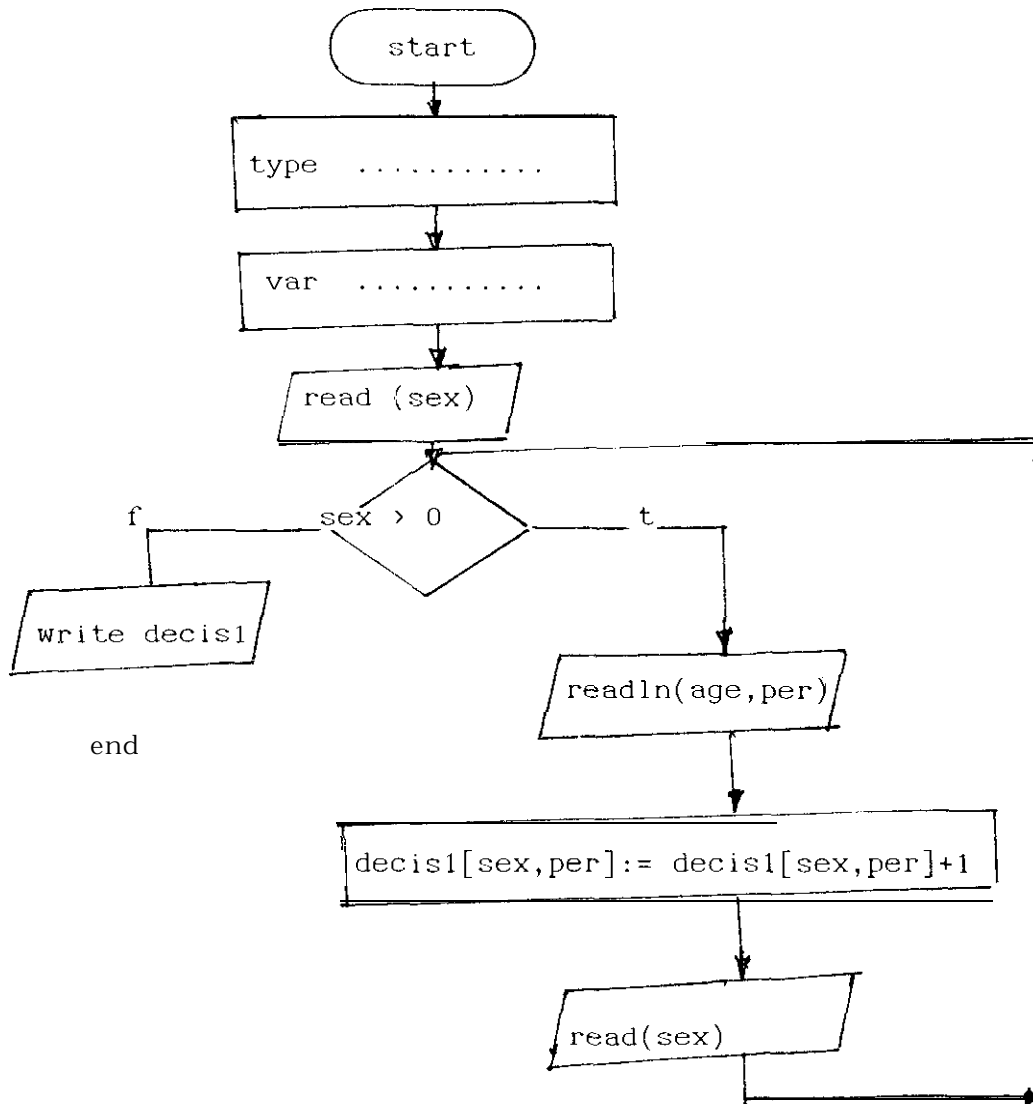
task3 : $\text{decis1}[1,3] := \text{decis1}[1,3]+1$

task4 : $\text{decis1}[2,1] := \text{decis1}[2,1]+1$

task5 : $\text{decis1}[2,2] := \text{decis1}[2,2]+1$

task6 : $\text{decis1}[2,3] := \text{decis1}[2,3]+1$

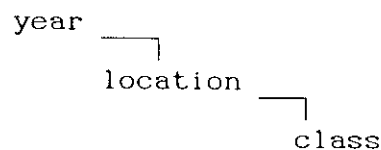
แบบที่ 2



แบบที่ 2 ที่ปรากฏ คุณจะคิดว่าในแง่ที่สั้นกว่า ทั้งนี้สืบเนื่องมาจากการออกแบบระบบรหัสให้ตรงกับตัวชี้ของ Array ที่เราสร้างขึ้น และในแง่ของการนำ Array ที่ได้ไปตอบคำถาม ก็ดูเข้าใจได้ง่ายกว่า

แนวคิดและการดำเนินงานอันสืบเนื่องมาจากการใช้ Array

- การตั้งตัวชี้ในกรณีของ Array ที่มีขนาดเกิน 2 มิติขึ้นไปควรจะลำดับตามศักดิ์ (Hierarchy) ลักษณะของสารสนเทศที่กำหนด ยกตัวอย่างเช่น ถ้าจะเก็บข้อมูลสินค้าของบริษัทในลักษณะเช่นเดียวกับตารางที่แสดงในตัวอย่างที่ 1 แต่มีการเพิ่มเงื่อนไขให้เก็บเป็นรายปีด้วยเริ่มเก็บตั้งแต่ปี 1980,...,1994 ดังนั้นจาก Array เดิม 2 มิติก็จะกลายมาเป็น Array 3 มิติ โดยที่ มิติแรกควรจะเป็นการจำแนกตาม ปี.ศ. มิติที่สองจำแนกตามที่ตั้งของคลังสินค้า และมิติที่สามจำแนกตามประเภทของสินค้าแต่ละชนิด ตัวอย่างคำสั่งในการประกาศของตัวแปร Array ในกรณีนี้ก็กล่าวถึงจะปรากฏดังนี้



type

```

year = 1980..1994 ;
locat = (central,east,north,south);
class = (corn,sugar,rice,bean);
stock = array [year,locat,class];
  
```

var

```

onhand : stock;
  
```

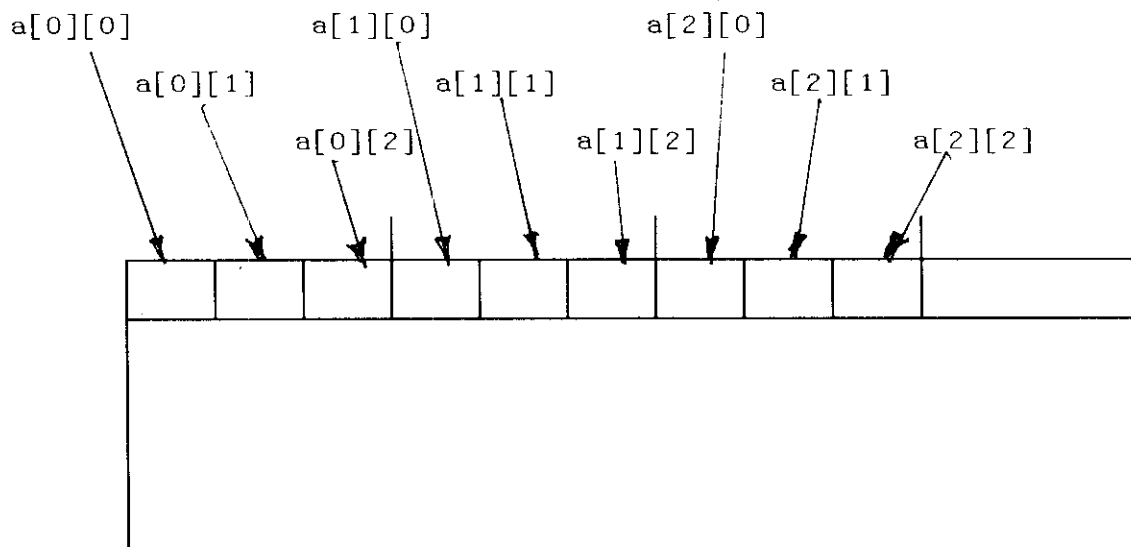

- การจัดสรรพื้นที่ในสมองของคอมพิวเตอร์ให้กับ Array ที่มีลักษณะหลายมิติ นั้นจะแตกต่างกันไปตามแต่ละภาษา เราจะจำแนกการจัดสรรได้ออกเป็น 2 แบบ คือ

1. แบบ row major

เป็นการจัดสรรพื้นที่ให้ Array โดยยึดเอาแถวเป็นหลัก ตัวอย่าง เช่น ภาษา C ตัวอย่างเช่น ถ้าเราเขียนคำสั่งในภาษา C ว่า

```
int a[3][3] ;
```

ถ้าคำสั่งได้รับการตอบสนองแล้ว เครื่องจะจัดสรรพื้นที่ ให้กับตัวแปร a จำนวน 3 x 3 cell (element) โดยที่แต่ละ cell จะมีขนาด 2 bytes ลักษณะโครงสร้างที่ปรากฏ จะเป็นดังนี้คือ

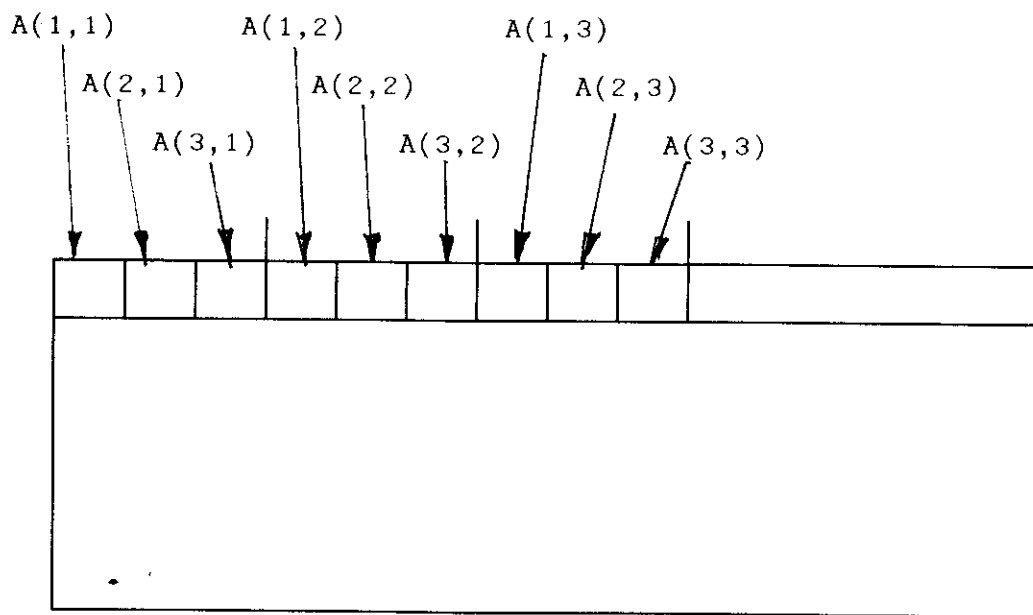


1. แบบ column major

เป็นการจัดสรรพื้นที่ให้ Array โดยยึดเอาสดมภ์เป็นหลัก ตัวอย่างเช่น ภาษา FORTRAN ตัวอย่างเช่น ถ้าเราเขียนคำสั่งในภาษา FORTRAN ว่า

DIMENSION A(3,3)

ถ้าคำสั่งได้รับการตอบสนองแล้ว เครื่องจะจัดสรรพื้นที่ ให้กับตัวแปร A จำนวน 3 x 3 cell (element) โดยที่แต่ละ cell จะมีขนาด 2 bytes ลักษณะโครงสร้างที่ปรากฏ จะเป็นดังนี้คือ



การเรียนรู้โครงสร้างการจัดเก็บข้อมูลในเรื่องของ Array จะส่งผลในเรื่องของการส่งต่อพารามิเตอร์ให้กับ Module อื่นที่ติดต่อกับ ตัวอย่างเช่นถ้าเราต้องการผลบวกของ row ใด row หนึ่งไปให้ Module หนึ่งเพื่อหาผลรวม แต่ไม่ทราบวิธีการจัดเก็บ เราก็สามารถดำเนินการได้เพราะไม่ทราบที่ตั้งของข้อมูลที่ต้องการ

- การจัดสรรพื้นที่ให้กับ Array นั้นจะเห็นได้ว่าจำเป็นจะต้องใช้พื้นที่ต่อเนื่องกันตลอด ดังนั้น ในกรณีของการประกาศของ Array ขนาดใหญ่ก็อาจจะล้มเหลวได้ถ้าหากเกิดเหตุการณ์ที่เราเรียกว่า overflow error เกิดขึ้น ดังนั้นทางออกของเราก็คือจะต้องพยายามยืดหลักการของการออกแบบมาใช้ พื้นที่เท่าที่จำเป็นจริงๆ เท่านั้น การประกาศของพื้นที่ในตัวอย่างต่อไปนี้จะทำให้เราเห็นรายละเอียดปลีกย่อยที่ทำให้เกิดขึ้น

นักเรียนคนที่	คะแนนแต่ละประเภท		
	คะแนนการบ้าน	คะแนนสอบกลางภาค	คะแนนสอบปลายภาค
1	67	82	69
2	43	65	72
:			
100	56	76	74

สมมติว่าเราประกาศของพื้นที่ให้ Array คือ

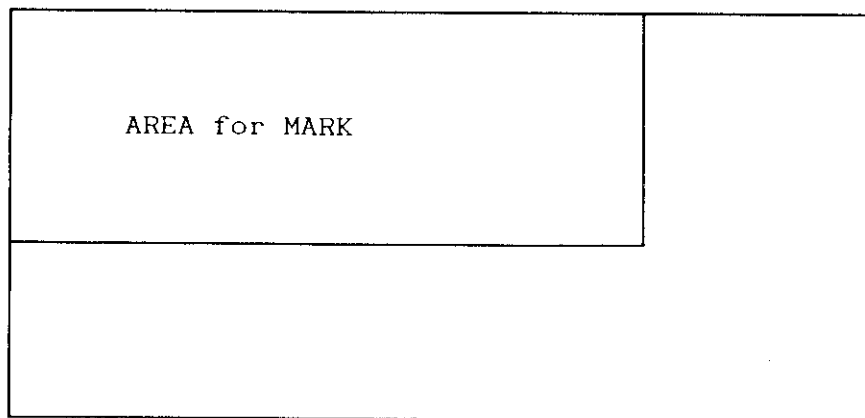
```
var
```

```
MARK : array [1..10,1..3] of integer ;
```

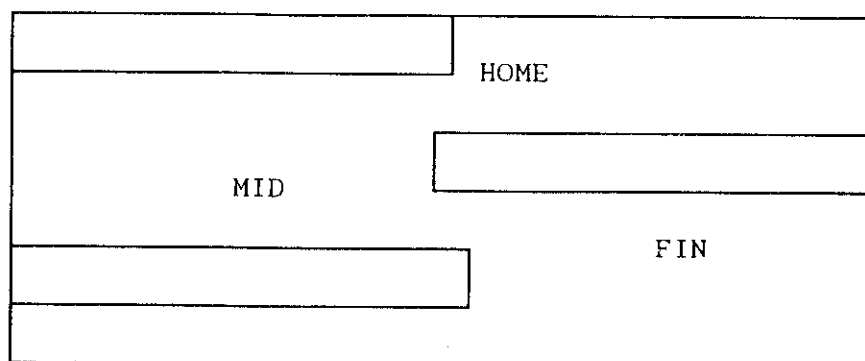
ผลที่เกิดขึ้นก็คือเครื่องจะต้องจัดสรรพื้นที่จำนวน $100 \times 3 \times 2$ bytes (กำหนดให้ 1 cell ใช้พื้นที่เท่ากับ 2 bytes) ดังนั้นพื้นที่ที่ใช้จะเท่ากับ 600 bytes ประมาณ 1/2 ถ้าหากว่าคอมพิวเตอร์มีที่เหลืออยู่มากกว่า 1/2 K จริง ก็อาจจะเป็นไปได้ว่า MARK อาจจะไม่ได้รับการจัดสรรก็ได้ เพราะว่าพื้นที่ยังมีปรากฏนั้นอาจจะอยู่ไม่ต่อเนื่องก็ได้ ซึ่งขัดกับ

ข้อตกลง สิ่งที่น่าจะเป็นไปได้ก็คือเราอาจจะออกแบบเสียใหม่โดยการมอตารางดังกล่าวในรูปของ Array มิติเดียวจำนวนทั้งสิ้น 3 Array ดังนั้น $MARK = \{HOME, MID, FIN\}$ โดยที่ HOME, MID, FIN ก็คือ Array มิติเดียวที่มีจำนวน Array ละ 100 element ซึ่งรวมแล้วก็ยังใช้ พื้นที่ 600 bytes เท่าเดิม แต่การออกแบบวิธีนี้อาจจะได้รับการจัดสรรสำเร็จก็ได้สืบเนื่องมาจากสาเหตุที่ว่า Array ทั้งสามอาจจะอยู่คนละส่วนพื้นที่ไม่จำเป็นจะต้องอยู่ต่อเนื่องกันไป ดังรูปเปรียบเทียบต่อไปนี้

รูปที่ 1 ออกแบบให้ MARK เป็น 2 มิติ ($MARK[100,3]$)



รูปที่ 2 ออกแบบให้ MARK เป็น องค์ประกอบของ HOME, MID, FIN



- การออกแบบ Array ประเภทหลายมิตินั้น แต่ละ element ของ Array จะต้องมี type (numeric or string) size (size of string or size of numeric (may be integer or longinteger or real or double)) เป็นอย่างเดียวกัน ดังนั้น ในกรณีของตัวอย่างของข้อมูลของพนักงานต่อไปนี้เราจะออกแบบตารางนี้ให้เป็นแบบ Array หลายมิติไม่ได้

ชื่อคนงาน	อายุ	เพศ	เงินเดือน
string	int	char	real

ดังนั้นจากตารางนี้ เราจะต้องออกแบบให้เป็น Array มิติเดียวจำนวน 4 Array โดยที่ Array แต่ละอันก็จะมี element แตกต่างกันไป ตัวอย่างการเขียนคำสั่งดังนี้

```

const
    size = 100;

type
    str = array[1..10] of char;

var
    name : array [1..size] of str ;
    age  : array [1..size] of integer;
    sex  : array [1..size] of char;
    sala : array [1..size] of real;

```

- ดังที่ได้กล่าวมาแล้วว่า การจัดสรรพื้นที่ให้กับ Array นั้นเราจำเป็นต้องใช้พื้นที่ที่ต่อเนื่องกันไป ดังนั้นเราจึงจำเป็นต้องประหยัคพื้นที่ วิธีการที่เราจะช่วยประหยัคพื้นที่นั้น เราสามารถดำเนินการได้ดังนี้คือ

1. reused area

หลักการที่กล่าวถึงในข้อนี้ก็คือนำพื้นที่ที่ใช้แล้วและไม่จำเป็นต้องใช้อีก มาใช้เก็บข้อมูลของ Array ынต่อไป ดังตัวอย่างนี้

นักเรียนคนที่	คะแนนแต่ละประเภท		
	คะแนนการบ้าน	คะแนนสอบกลางภาค	คะแนนสอบปลายภาค
1	67	82	69
2	43	65	72
:			
100	56	76	74

สมมติว่าเราต้องการที่จะคำนวณหาคะแนนรวมของนักศึกษาแต่ละคน โดยที่

$$\text{คะแนนรวม} = \text{คะแนนการบ้าน} + \text{คะแนนสอบกลางภาค} + \text{คะแนนสอบไล่}$$

ดังนั้น เราอาจจะนำคะแนนรวมที่ได้ไปจัดเก็บไว้ในตำแหน่งของการบ้านก็ได้ ถ้าหากเราไม่มีความจำเป็นต้องเก็บข้อมูลเรื่องเกี่ยวกับคะแนนการบ้านไว้อีกแล้ว ภายหลังเมื่คำนวณหาคะแนนรวมได้แล้ว

2. reduced representation form

การจัดการในลักษณะนี้มักจะเกิดขึ้นในกรณีของการจัดเก็บข้อมูลคศาสตร์ในรูปของ Array ดังตัวอย่างต่อไปนี้

ตัวอย่างที่ 1 Sparse Matrix หมายถึงเมทริกซ์ที่ element ส่วนใหญ่จะมีค่าเป็น 0 ส่วนใหญ่ ดังตัวอย่างนี้คือ

$$A_{5 \times 5} \begin{vmatrix} 0 & 0 & 0 & 1 & 0 \\ 3 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \\ 7 & 0 & 0 & 0 & -22 \end{vmatrix}$$

สามารถจะจัดโครงสร้างในการจัดเก็บให้มีประสิทธิภาพได้ดังนี้คือ

$$A_{4 \times 3} \begin{vmatrix} 1 & 4 & 1 \\ 2 & 1 & 3 \\ 5 & 1 & 7 \\ 5 & 5 & -22 \end{vmatrix}$$

ตัวอย่างที่ 2 Digonal Matrix หมายถึงเมทริกซ์ที่ element บน main diagonal จะมีค่า ส่วน element ที่อยู่บน off diagonal matrix มีค่าเป็น 0 ดังตัวอย่างคือ

$$B_{5 \times 5} = \begin{bmatrix} 1 & 0 & 0 & 0 & 0 \\ 0 & 3 & 0 & 0 & 0 \\ 0 & 0 & 5 & 0 & 0 \\ 0 & 0 & 0 & 7 & 0 \\ 0 & 0 & 0 & 0 & 14 \end{bmatrix}$$

สามารถจะจัดโครงสร้างในการจัดเก็บให้มีประสิทธิภาพได้ดังนี้คือ

$$B_{5 \times 1} = \begin{bmatrix} 1 \\ 3 \\ 5 \\ 7 \\ 14 \end{bmatrix}$$