บทที่ 6 อาร์เรย์ (Array) (ต่อ)

6.4 ตัวแปรชุดแบบหลายมิติ

จากพื้นฐานที่ผ่านมาเรื่องตัวแปรชุดจะเป็นลักษณะของตัวแปรชุดมิติเดียว แต่ตัวแปรชุดอาจจะมีมากกว่า 1 มิติก็ได้ ข้อมูลตัวแปรชุด 2 มิติเป็นข้อมูลที่สามารถพบเห็นได้บ่อยในชีวิตประจำ วัน เช่น ตารางค่าโดยสารรถไฟจากเมืองงหนึ่งไปยังอีกเมืองหนึ่ง ตารางระยะทางจากเมืองหนึ่งไปยังอีกเมืองหนึ่ง ตารางสูตรคูณ เป็นต้น

จะสังเกตว่าข้อมูลในลักษณะของตัวแปรชุด 2 มิติจะเป็นในลักษณะของตาราง พิจารณาจากตัวอย่างการเก็บข้อมูลคะแนนสอบของนักศึกษาวิชาหนึ่ง ซึ่งแบ่งการเก็บคะแนนออกเป็น 5 ครั้ง จะพบว่าหากต้องการเก็บข้อมูลคะแนนสอบของนักศึกษาแต่ละคนสามารถใช้ตัวแปรชุดมิติเดียว แสดงดังรูปที่ 6.6 ด้านล่าง และใช้คำสั่ง

```
#define NUMBER_OF_PAPERS
float score[ NUMBER_OF_PAPERS ];
```

score[0]	score[1]	score[2]	score[3]	score[4]
5.6	8.5	12.6	24.1	16.0

รูปที่ 6.6 แสดงการเก็บคะแนนสอบของนักศึกษาคนหนึ่ง

แต่หากเพิ่มเติมว่าให้เก็บข้อมูลของนักศึกษาทุกคนในชั้นนั้น จะต้องใช้ตัวแปรชุดหลายมิติเข้ามาเกี่ยวข้องตัวอย่างของข้อมูลที่เก็บดังรูปที่ 6.7

	ครั้งที่ 1	ครั้งที่ 2	ครั้งที่ 3	ครั้งที่ 4	ครั้งที่ 5
นาย ก	5.6	8.5	12.6	24.1	16.0
นาย ข	6.0	7.2	15.0	25.0	18.0
.....	.....	.....	.....	.....	.....

รูปที่ 6.7 แสดงตัวอย่างการเก็บข้อมูลคะแนนของนักศึกษา

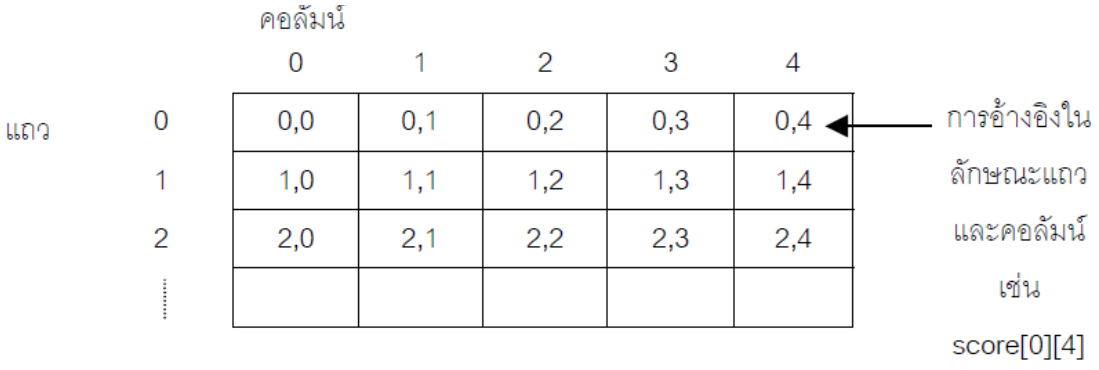
ข้อมูลที่จะจัดเก็บในตัวแปรชุดจะต้องเป็นข้อมูลชนิดเดียวกันเสมอ ในที่นี้จะจัดเก็บเฉพาะส่วนของคะแนนสอบของนักศึกษาเท่านั้น จากลักษณะความต้องการเก็บข้อมูลดังกล่าวจะต้องเตรียมตัวแปรชุดเพื่อเก็บข้อมูลในลักษณะ 2 มิติ สามารถประกาศตัวแปรชุดดังนี้

```
#define NUMBER_OF_PAPERS 5
#define NUMBER_OF_STUDENTS 50
float score[NUMBER_OF_STUDENTS][NUMBER_OF_PAPERS];
```

ตัวแปรชุด 2 มิติจะมองข้อมูลในลักษณะแถวและคอลัมน์ แถวของข้อมูลในที่นี้จะเป็นคะแนนที่นักศึกษาแต่ละคนได้รับ ส่วนคอลัมน์จะเป็นคะแนนสอบแต่ละครั้งของนักศึกษา เพราะฉะนั้นเมื่อเราอ้างถึงจุดใดจุดหนึ่งในตัวแปรชุด 2 มิติ ก็จะเป็นคะแนนที่นักศึกษาแต่ละคนได้รับในการสอบครั้งที่ระบุ เช่น อ้างถึงแถวที่ 0 คอลัมน์ที่ 4 จะเป็นคะแนนสอบครั้งที่ 5 ของนักศึกษาคนที่ 1 การอ้างอิงข้อมูลในตัวแปรชุด 2 มิติใช้วิธีดัชนี (Index) ดังนี้

score[row][column] เช่น score[0][4]

การทำงานของตัวแปรชุด 2 มิติจะเหมือนกับตัวแปรชุดมิติเดียว โดยจะดูว่าเป็นตัวแปรชุดที่มีสมาชิกเป็นตัวแปรชุด 1 มิติ สามารถแสดงรูปจำ ลองของตัวแปรชุด 2 มิติดังรูปที่ 6.8



รูปที่ 6.8 แสดงแบบจำลองของตัวแปรชุด 2 มิติ

การจัดเก็บตัวแปรชุด 2 มิติในหน่วยความจำ เนื่องจากการเก็บข้อมูลในหน่วยความจำ จะเก็บเรียงตามลำดับ ภาษาโปรแกรมแต่ละภาษาจะมีการจัดเก็บตัวแปรชุดหลายมิติในหน่วยความจำ แตกต่างกัน โดยผ่านทาง Storage Mapping Function ตัวอย่างการจัดเก็บตัวแปรชุด 2 มิติ

int table[3][4];

จะมีการจัดเก็บข้อมูลเรียงลำดับภายในหน่วยความจำ ดังรูปที่ 6.9 เมื่อต้องการเข้าถึงสมาชิกตัวแปรชุด **table[1][3]** จะตรงกับตำแหน่งที่ 7 ในหน่วยความจำ (สมมติให้เริ่มนับจาก 0) ฟังก์ชันที่ใช้ในการหาคือ table[i][j] จะตรงกับตำแหน่ง **4 * i + j** ฟังก์ชันที่ใช้หาตำแหน่งของตัวแปรชุดหลายมิติในหน่วยความจำ จะขึ้นอยู่กับค่าคงที่ภายในสูตร ในกรณีของตัวแปรชุด 2 มิติค่าที่ใช้คือ 4 หากเป็นตัวแปรชุดหลายมิติที่ไม่ใช่ 2 จะต้องหาค่าคงที่ที่จะเป็นตัวกำหนดนี้ ทั้งนี้คอมไพเลอร์จะทำ หน้าที่ในการหาตำแหน่งในหน่วยความจำ ให้กับผู้เขียนโปรแกรมโดยอัตโนมัติ

table[0][0]	0
table[0][1]	1
table[0][2]	2
table[0][3]	3
table[1][0]	4
table[1][1]	5
table[1][2]	6
table[1][3]	7

รูปที่ 6.9 แสดงตัวอย่างการจัดเก็บข้อมูลตัวแปรชุด 2 มิติในหน่วยความจำ

6.5 การกำหนดค่าเริ่มต้นให้กับตัวแปรชุด 2 มิติ

เนื่องจากเมื่อมีการประกาศตัวแปรชุดจะมีการจองพื้นที่ให้ตัวแปรชุด โดยที่ค่าที่อยู่ในพื้นที่หน่วยความจำที่จองอาจจะมีค่าใด ๆ ที่เราไม่ต้องการอยู่ การกำหนดค่าเริ่มต้นให้กับตัวแปรชุดเป็นการทำงานเพื่อให้สมาชิกของตัวแปรชุดมีค่าเริ่มต้นที่เราต้องการ สามารถกำหนดค่าเริ่มต้นได้ดังนี้

```
int number[ ][5]={{1, 2, 3, 4, 5}, {2, 4, 6, 8, 10}, {1, 3, 5, 7, 9}};
```

การกำหนดค่าเริ่มต้นจะทำพร้อมกับการประกาศตัวแปรชุดเท่านั้น โดยไม่จำเป็นต้องกำหนดขนาดจำนวนแถวของตัวแปรชุดก็ได้ เนื่องจากเมื่อมีการกำหนดค่าเริ่มต้นจะมีการจองพื้นที่ตัวแปรชุดให้เท่ากับข้อมูลที่ใช้ในการกำหนดค่าเริ่มต้นนั้น ในที่นี้จะจอง **ตัวแปรชุด 2 มิติขนาด 3 แถว 5 คอลัมน์** เครื่องหมาย { } จะเป็นส่วนของการกำหนดค่าในแต่ละแถว ในที่นี้จะสามารถจำลองการเก็บข้อมูลได้ดังรูปที่ 6.10

number	[0]	[1]	[2]	[3]	[4]
[0]	1	2	3	4	5
[1]	2	4	6	8	10
[2]	1	3	5	7	9

รูปที่ 6.10 แสดงการเก็บข้อมูลในตัวแปร number

6.5 การใช้งานตัวแปรชุด 2 มิติ

ดังที่กล่าวมาแล้วว่าการส่งผ่านตัวแปรชุดมิติเดียวไปยังฟังก์ชันเป็นการทำงานในลักษณะเดียวกับพอยน์เตอร์ การใช้งานตัวแปรชุด 2 มิติหรือหลายมิติกับฟังก์ชันก็เป็นการทำงานในลักษณะเดียวกัน การเปลี่ยนแปลงค่าใด ๆ ในฟังก์ชัน จะมีผลกระทบมายังตัวแปรชุดที่ส่งเข้าไปยังฟังก์ชันนั้นด้วย การทำงานภายในตัวแปรชุด 2 มิติ ผู้เขียนต้องสังเกตความสัมพันธ์ระหว่างสมาชิกในแถวและคอลัมน์ให้ถูกต้อง ก็จะสามารถเข้าถึงข้อมูลที่ต้องการได้

การส่งตัวแปรชุด 2 มิติไปเป็นอาร์กิวเมนท์ของฟังก์ชัน จะต้องประกาศขนาดความยาวของคอลัมน์ ดังตัวอย่างฟังก์ชันการกำหนดค่าเริ่มต้นให้กับตัวแปรชุด 2 มิติ แสดงดังตัวอย่างที่ 6.11

ตัวอย่างที่ 6.11 โปรแกรมเพื่อสร้างข้อมูลเข้าไปเก็บยังตัวแปรชุด 2 มิติที่มีขนาดแถวและคอลัมน์เท่ากัน และให้พิมพ์ค่าดังกล่าว ซึ่งมีลักษณะดังรูป

```
+ + + + +
+ - - - +
+ - - - +
+ - - - +
```

```c
#include <stdio.h>
#define MAX 5
void createArray(char [ ][MAX]);
void printArray(char [ ][MAX]);
void main() {
    char arr[MAX][MAX];
    createArray(arr);
    printArray(arr);
}
void createArray(char parr[ ][MAX]) {
    int i, j;
    for (i=0; i<MAX; i++) {
        for (j=0; j<MAX; j++) {
            if (i==0 || j==0 || i==MAX-1 || j==MAX-1)
                parr[i][j] = '+';
            else
                parr[i][j] = '-';
        }
    }
}
void printArray(char parr[ ][MAX]) {
    int i, j;
    printf("\nShow array : \n");
    for (i=0; i<MAX; i++) {
        for (j=0; j<MAX; j++) {
            printf("%4c", parr[i][j]);
        }
        printf("\n");
    }
}
```

จากตัวอย่างจะต้องพิจารณาความสัมพันธ์ที่เกิดขึ้นจากรูปที่โจทย์กำหนด โดยแทนตำแหน่งของสมาชิกลงในรูปจะได้ดังรูปที่ 6.11

0,0	0,1	0,2	0,3	0,4
1,0	1,1	1,2	1,3	1,4
2,0	2,1	2,2	2,3	2,4
3,0	3,1	3,2	3,3	3,4
4,0	4,1	4,2	4,3	4,3

รูปที่ 6.11 แสดงการแทนตำแหน่งลงในรูปที่ระบุ

เนื่องจากข้อมูลที่แสดงเป็นเครื่องหมาย + และ – เพราะฉะนั้นจึงประกาศให้ตัวแปรชุดมีเป็นชนิด char ตำแหน่งที่เป็น + จะตรงกับตำแหน่งที่เป็นสีทึบ **ให้ i แทนค่าแถวและ j แทนค่าคอลัมน์** จะพบว่าด้านที่เป็นสีทึบนั้นคือ ด้านที่ i มีค่าเป็น 0 (แนวนอนด้านบนสุด) ด้านที่ i มีค่าเท่ากับขนาดตัวแปรชุดลบไป 1 (แนวนอนด้านล่างสุด) ด้านที่ j มีค่าเท่ากับ 0 (แนวตั้งด้านซ้ายสุด) และด้านที่ j มีค่าเท่ากับขนาดตัวแปรชุดลบไป 1 (แนวตั้งด้านขวาสุด) ส่วนตำแหน่งอื่น ๆ จะมีค่าเป็น - โดยอาศัยการสังเกตค่า i และ j ดังกล่าว จะสามารถทำ ให้เลือกกำหนดค่าสมาชิกของตัวแปรชุดที่ต้องการได้ ถึงแม้ขนาดของรูปจะเปลี่ยนไป ก็จะได้ผลลัพธ์ที่เหมือนเดิม ให้ทดลองโดยการกำหนดค่า MAX ขึ้นใหม่ และทดลองรันโปรแกรมอีกครั้ง

ส่วนการส่งผ่านตัวแปรชุด ไปในฟังก์ชันสามารถทำ ได้ดังตัวอย่าง เช่น หากในฟังก์ชัน main( ) มีตัวแปรชุด ชื่อ arr ดังคำสั่ง

char arr[MAX][MAX];

createArray(arr);

การส่งตัวแปรชุด arr ไปยังฟังก์ชัน createArray( ) สามารถอ้างชื่อของตัวแปรชุด arr โดยตรง ซึ่งจะเหมือนการอ้างในลักษณะพอยน์เตอร และการอ้างถึงตัวแปรชุดมิติเดียว การประกาศฟังก์ชัน createArray( ) สามารถทำ ได้ด้วยคำสั่ง

void createArray(char parr[ ][MAX])

เมื่อ parr คือชื่อตัวชุดที่มารับค่าตัวแปรชุด arr โดยที่ไม่ต้องระบุขนาดแถว แต่ให้ระบุขนาดของคอลัมน์เสมอ การอ้างถึงตัวแปร parr เช่น การกำหนดค่า หรือการรับค่า จะเป็นเหมือนกับการอ้างถึงตัวแปรชุด arr ในฟังก์ชัน main( ) เมื่อค่าใน parr เปลี่ยนแปลง ค่า arr จะเปลี่ยนแปลงด้วยทันที การใช้งานตัวแปรชุด 2 มิติมักจะใช้คำ สั่งวนรอบซ้อนกันในการทำงานเสมอ ดังตัวอย่างเพิ่มเติมในตัวอย่างที่ 6.18 และ 6.19

ตัวอย่างที่ 6.12 ข้อมูลค่าจ้างของพนักงานในบริษัทแห่งหนึ่ง ซึ่งแบ่งพนักงานออกเป็น 5 ระดับ ในแต่ละระดับจะแบ่งออกเป็น 4 ขั้น ซึ่งแต่ละขั้นจะได้รับเงินเดือนไม่ไม่เท่ากัน แสดงดังตารางค่าจ้าง

	ขั้น 1	ขั้น 2	ขั้น 3	ขั้น 4
ระดับ 1	4,200	4,400	4,650	4,950
ระดับ 2	5,200	5,400	5,650	5,950
ระดับ 3	6,200	6,400	6,650	6,950
ระดับ 4	7,200	7,400	7,650	7,950
ระดับ 5	8,200	8,400	8,650	8,950

ให้เขียนโปรแกรมซึ่งมีการกำหนดค่าเริ่มต้นของค่าจ้างเหล่านี้ในโปรแกรม โดยที่โปรแกรมจะต้องมีความสามารถต่าง ๆ ดังนี้

- คำนวณหาค่าจ้างเฉลี่ยในแต่ละระดับ

```
#include <stdio.h>
#define NO_LEVEL 5
#define NO_STEP 4
void calAvgLevel(float [] [NO_STEP]);
void calAvgStep(float [][NO_STEP]);
void calAvgAll(float [][NO_STEP]);
void increaseSalary(float [][NO_STEP], float [] [NO_STEP], float);
void printSalary(float [][NO_STEP]);
void main() {
    float salary[][NO_STEP]={{4200.0, 4400.0, 4650.0, 4950.0}, {5200.0, 5400.0, 5650.0,
            5950.0},{6200.0, 6400.0, 6650.0, 6950.0}, {7200.0, 7400.0, 7650.0, 7950.0},
            {8200.0, 8400.0, 8650.0, 8950.0}};
    float newSalary[NO_LEVEL][NO_STEP];
    calAvgLevel(salary);
    calAvgStep(salary);
    calAvgAll(salary);
    increaseSalary(salary, newSalary, 7.5);
    printf("\n\nNew salary table is\n");
    printSalary(newSalary);
}
void calAvgLevel(float sal[][NO_STEP]) {
    float sum, avg;
    int i, j;
    printf("\n\nAverage salary for level : ");
    for (i=0; i<NO_LEVEL; i++) {
        sum = 0.0;
        for (j=0; j<NO_STEP; j++)
            sum = sum + sal[i][j];
        avg = sum / NO_STEP;
        printf("\n\t\t%d is %.2f", i, avg);
    }
}
void calAvgStep(float sal[] [NO_STEP]) {
    float sum, avg;
    int i, j;
    printf("\n\nAverage salary for step : ");
    for (i=0; i<NO_STEP; i++) {
        sum = 0.0;
        for (j=0; j<NO_LEVEL; j++)
            sum = sum + sal[j][i];
```

ที่เพิ่มขึ้น

```
        avg = sum / NO_LEVEL;
        printf("\n\t\t%d is %.2f", i, avg);
    }
}
void calAvgAll(float sal[] [NO_STEP]) {
    float sum=0.0, avg;
    int i, j;
    for (i=0; i<NO_LEVEL; i++)
        for (j=0; j<NO_STEP; j++)
            sum = sum + sal[i][j];
        avg = sum / (NO_LEVEL * NO_STEP);
   printf("\n\nAverage salary for all levels and steps is %.2f", avg);
}
void increaseSalary(float baseSal[][NO_STEP], float newSal[] [NO_STEP], float percentInc) {
    int i, j;
    for (i=0; i<NO_LEVEL; i++)
        for (j=0; j<NO_STEP; j++)
            newSal[i][j] = baseSal[i][j] + (baseSal[i][j]*percentInc/100);
}
void printSalary(float sal[][NO_STEP]) {
    int i, j;
    for (i=0; i<NO_LEVEL; i++) {
        for (j=0; j<NO_STEP; j++)
            printf("%12.2f", sal[i][j]);
        printf("\n");
}
}
```

ปัญหาหลักเมื่อพบในการเขียนโปรแกรมในลักษณะตัวแปรชดุ 2 มิติ คือ ไม่รู้ว่าจะต้องเขียนคำ สั่ง for ของแถวหรือคอลัมน์ไว้ที่ตำแหน่งใด สังเกตได้ว่า

- ถ้าเป็นการเข้าถึงสมาชิกทุกตัวจะเขียนคำสั่ง for ของแถวหรือคอลัมน์ก่อนก็ได้ เช่น การคำนวณหาค่าเฉลี่ยของค่าจ้างในทุกระดับและทุกขั้น ต้องมีการหาผลรวมของค่าจ้างของสมาชิกทุกตัวในตัวแปรชุดซึ่งจะต้องเข้าถึงสมาชิกทุกตัว เพราะฉะนั้นจะเขียนคำสั่งวนซ้ำของแถวหรือคอลัมน์ก่อนก็ได ดังตัวอย่างในฟังก์ชัน calAvgAll( ) จะต้องงมีการหาผลรวมทั้งหมดก่อนหาค่าเฉลี่ย และฟังก์ชัน increaseSalary( ) ที่จะต้องงมีการเพิ่มค่าจ้างให้ในทุกระดับและทุกขั้น

- หากต้องการหาผลรวมในแนวแถว จะต้องตีความปัญหาให้ได้ก่อน เช่น หาค่าจ้างเฉลี่ยในแต่ละระดับ ในที่นี้ระดับคือแถว เพราะฉะนั้นต้องใช้คำสั่ง for ควบคุมแถวอยู่ลูปนอก ต้องมีการกำหนดให้ค่าตัวแปร sum เป็น 0 ก่อน เพื่อให้ผลรวมในแต่ละขั้นเริ่มต้นที่ 0 เสมอ ในแต่ละแถวจึงจะมีการหาผลรวมของแต่ละขั้น (คอลัมน์) เมื่อได้ผลรวมแต่ละขั้นในระดับนั้นแล้ว จึงจะหาค่าเฉลี่ยในระดับนั้นได้ ดังตัวอย่างในฟังก์ชัน calAvgLevel( ) ในคำ สั่ง for ด้านในสังเกตว่าค่า i ที่เป็นค่าแถวจะคงที่เสมอ

- หากต้องการหาผลรวมในแนวคอลัมน์ เช่น หาค่าจ้างเฉลี่ยในแต่ละขั้น ในที่นี้ขั้นเป็นคอลัมน์ ในการคำนวณจะต้องหาว่าผลรวมในแต่ละระดับในขั้นนั้นมีอะไรบ้าง เพราะฉะนั้นคำ สั่ง for ควบคุมคอลัมน์จึงอยู่ด้านนอก แล้วจึงหาผลรวมของแต่ละระดับ ก็จะได้ค่าผลรวมเพื่อนำ มาหาค่าเฉลี่ยในขั้นนั้น ๆดังตัวอย่างในฟังก์ชัน calAvgCol( ) ให้สังเกตว่าในคำ สั่ง for ด้านในค่าของ i ที่เป็นค่าของคอลัมน์จะคงที่เสมอ

ตัวอย่างที่ 6.13 โปรแกรมเพื่อรับค่าข้อมูลคะแนนของนักศึกษาในกระบวนวิชาหนึ่งซึ่งมีนักศึกษา 25 คน แบ่งคะแนนของนักศึกษาออกเป็น 2 ส่วน ได้แก่ คะแนนสอบกลางภาค และคะแนนสอบปลายภาค ให้คำนวณหาคะแนนรวมที่นักศึกษาแต่ละคนได้รับ และเกรดที่นักศึกษาแต่ละคนจะได้รับเมื่อกำหนดให้ช่วงคะแนน เกรดที่ได้รับ

น้อยกว่า 50 F

50 ถึง 59 D

60 ถึง 69 C

70 ถึง 79 B

80 ขึ้นไป A

```
#include <stdio.h>
#define MAX_STD 3
#define MAX_SCR 2
void readScore(float [][MAX_SCR]);
void findTotalScore(float [][MAX_SCR] , float []);
void calGrade(float [], char []);
void printResult(float [][MAX_SCR], float [], char []);
void main() {
    float score[MAX_STD][MAX_SCR], totalScore[MAX_STD];
    char grade[MAX_STD];
    readScore(score);
    findTotalScore(score, totalScore);
    calGrade(totalScore, grade);
    printResult(score, totalScore, grade);
}
void readScore(float pScore[][MAX_SCR]) {
    int i, j;
    for (i=0; i<MAX_STD; i++) {
        printf("\nEnter score for student no.%d : \n", i+1);
        for (j=0; j<MAX_SCR; j++) {
            printf("\tscore no.%d : ", j+1);
            scanf("%f", &pScore[i][j]);
        }
    }
}
void findTotalScore(float pScore[][MAX_SCR], float pTotal[]) {
    int i, j;
    for (i=0; i<MAX_STD; i++) {
        pTotal[i] = 0.0;
        for (j=0; j<MAX_SCR; j++)
            pTotal[i] = pTotal[i] + pScore[i][j];
   }
}
void calGrade(float pTotal [], char pGrade[]) {
    int i;
    for (i=0; i<MAX_STD; i++) {
        if (pTotal[i] < 50)
            pGrade[i] = 'F';
        else if (pTotal[i] < 60)
            pGrade[i] = 'D';
        else if (pTotal[i] < 70)
            pGrade[i] = 'C';
        else if (pTotal[i] < 80)
            pGrade[i] = 'B';
```

```
        else
            pGrade[i] = 'A';
    }
}
void printResult(float pScore[][MAX_SCR], float pTotal[], char pGrade[]) {
    int i, j;
    printf("\nStudent score report\n");
    for (i=0; i<MAX_STD; i++) {
        printf("\t%-2d", i+1);
        for (j=0; j<MAX_SCR; j++)
            printf("%8.2f", pScore [i][j]);
        printf("%8.2f %c\n", pTotal[i], pGrade[i]);
    }
}
```

ให้สังเกตการรับข้อมูลเข้ามายังตัวแปรชุดหากเป็นการอ้างชื่อของสมาชิกในตัวแปรชุด จะต้องใช้เครื่องหมาย & เพื่อบอกให้รู้ว่าต้องการอ่านข้อมูลเข้ามาเก็บยังสมาชิกตัวที่ระบุ ทั้งนี้ใช้ได้กับข้อมูลพื้นฐานทุกประเภท ยกเว้นข้อมูลประเภทสตริงที่สามารถอ้างชื่อสมาชิกของตัวแปรชุดโดยไม่จำเป็นต้องใช้เครื่องหมาย &