บทที่ 9

แถวลำดับหลายมิติ

(Multidimensional Arrays)

ภาษา C ได้จัดเตรียมโครงสร้างข้อมูลแถวลำดับหลายมิติให้นักเขียนโปรแกรมนำมาใช้จัดเก็บข้อมูลที่มีโครงสร้างซับซ้อนได้และรูปแบบแถวลำดับหลายมิติที่ซับซ้อนน้อยที่สุด คือ แถวลำดับสองมิติ

9.1 แถวลำดับสองมิติ (two-dimensional arrays)

แถวลำดับสองมิติมีลักษณะคล้ายกับแถวลำดับมิติเดียว เพียงแต่ว่าสมาชิกแต่ละตัวเป็นแถวลำดับมิติเดียวเช่นกัน

9.1.1 การประกาศตัวแปรแถวลำดับสองมิติ

รูปแบบการประกาศตัวแปรแถวลำดับสองมิติคือ

```
type  array_name[1st dimension size][2nd dimension size]
```

1st dimension size คือ จำนวนแถว
2nd dimension size คือ จำนวนสดมภ์

ตัวอย่างการประกาศตัวแปรแถวลำดับสองมิติ 'd' ที่มีขนาดเป็น 10 แถว และ 20 สดมภ์ เพื่อเก็บจำนวนเต็ม 200 จำนวน สามารถประกาศได้ดังนี้

```
int   d[10][20]
```

9.1.2 การอ้างถึงสมาชิกแถวลำดับสองมิติ

การอ้างถึงสมาชิกแถวที่ 'i' และ สดมภ์ที่ 'j' ของแถวลำดับ 'd' สามารถทำได้โดยกำหนดชื่อแถวลำดับ แล้วตามด้วยดัชนีระบุเลขที่ของแถวและสดมภ์ ดังนี้

```
d[i][j]
```

ตัวอย่าง

d[3][5] หมายถึง สมาชิกแถวที่ 3 และสดมภ์ที่ 5 ของแถวลำดับ d

ตัวอย่างที่ 9.1 ส่วนของโปรแกรมต่อไปนี้ แสดงการนำจำนวนเต็ม 1-12 ไปเก็บไว้ในแถวลำดับสองมิติ 'num' ที่มี 3 แถว และ 4 สดมภ์

```
#include <stdio.h>
main( )
{
        int             i, j, num[3][4] ; /* i, j are integers,
                                             num is an array of 12 integers  */

        for     (i = 0 ; i < 3; ++i)            /* i is row index */
                for   (j = 0 ; j <4; ++j)       /* j is column index */
                        num[i][j] = (i*4) + j + 1;
        return 0;
}
```

ผลลัพธ์ที่ได้คือ

```
num[0][0] = 1
num[0][1] = 2
num[0][2] = 3
num[0][3] = 4
num[1][0] = 5
num[1][1] = 6
num[1][2] = 7
num[1][3] = 8
num[2][0] = 9
num[2][1] = 10
num[2][2] = 11
num[2][3] = 12
```

ดัชนีเริ่มต้นของแถวลำดับสองมิติ จะคล้ายกันกับดัชนีเริ่มต้นของแถวลำดับมิติเดียว นั่นคือ ดัชนีเริ่มต้นที่ระบุเลขที่ของแถวและสดมภ์ จะเป็น '0' ทั้งคู่

เมื่อกำหนด

```
int  num[3][4];
```

หมายถึง การประกาศแถวลำดับสองมิติเพื่อจัดเก็บจำนวนเต็ม ที่มีทั้งหมด 12 ช่อง ดังนี้

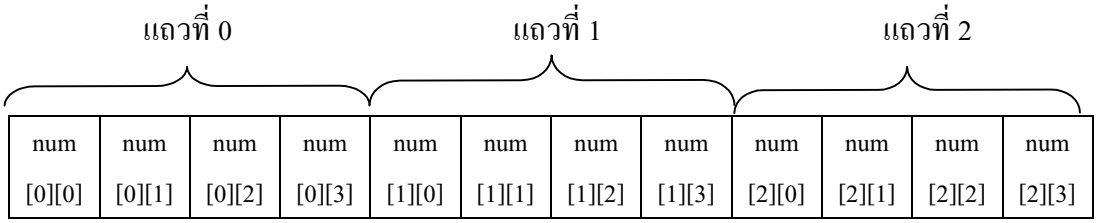
```
num[0][0], num [0][1], num[0][2], num [0][3], …, num[2][3]
```

9.1.3 การจัดเก็บแถวลำดับสองมิติในหน่วยความจำ

การจัดเก็บแถวลำดับสองมิติในหน่วยความจำนั้น เก็บแบบเรียงตามแถว (row major) เช่น เมื่อประกาศตัวแปรว่า

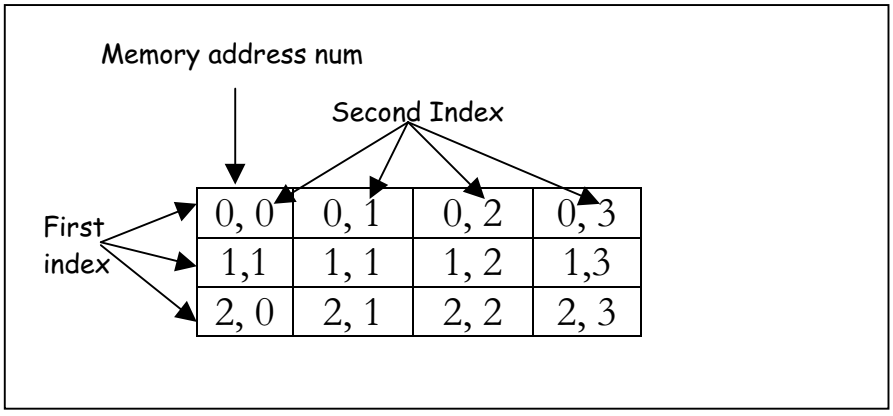
```
int  num[3][4]
```

แถวลำดับ 'num' จะถูกเก็บในหน่วยความจำดังรูปที่ 9.1



รูปที่ 9.1 แสดงการเก็บแถวลำดับสองมิติในหน่วยความจำ

เราอาจจำลองภาพการจัดเก็บแถวลำดับสองมิติ num[3][4] ในหน่วยความจำในรูปของตาราง ดังรูปที่ 9.2



รูปที่ 9.2 ตารางการจัดเก็บแถวลำดับสองมิติ num[3][4] ในหน่วยความจำ

เนื่องจากสมาชิกของแถวลำดับได้รับการจัดสรรเนื้อที่ด้วยปริมาณที่เท่ากัน ดังนั้น เราสามารถคำนวณหาจำนวนไบต์ (bytes) ที่ใช้จัดเก็บแถวลำดับสองมิติได้จากสูตร

```
byte = row* column * number of bytes in data type
```

ตัวอย่างเช่น ถ้าข้อมูลประเภท 'int' ใช้เนื้อที่ 2 ไบท์ เนื้อที่ทั้งหมดที่ใช้จัดเก็บแถวลำดับสองมิติ num[3][4] คือ 3*4*2 = 24 ไบต์

ถ้ากำหนดให้ ตัวแปรแถวลำดับ 'num' ถูกจัดเก็บ ณ ตำแหน่งเริ่มต้น เป็น '1000' ดังนั้น เลขที่อยู่ของ 'num [2][2]' คือ

```
(&num[0][0])
    + ( จำนวนไบต์สำหรับ 4 * 2 ช่องในแถวที่ 0 และแถวที่ 1)
    + (จำนวนไบต์ของสมาชิก 2 ตัวที่อยู่ก่อนหน้า)
      = 1000 + 8 * size of(int) + 2 * size of(int)
      = 1000 + 8 * 2 + 2 * 2
      = 1000 + 16 + 4  = 1020
```

นั่นคือ ตำแหน่งเลขที่อยู่ของ 'd[i][j]' ซึ่งถูกประกาศด้วย

```
int   d[max_row][max_cols]
```

คือ ตำแหน่ง

```
&d[0][0] +  i *max_cols * sizeof(int)
         +  j * sizeof(int)
```

การที่เราสนใจโครงสร้างภายในการจัดเก็บแถวลำดับสองมิติ ก็เพราะว่า เราสามารถใช้ตัวชี้ท่องเข้าไปในโครงสร้างแถวลำดับสองมิตินี้ได้ เหมือนกับว่าแถวลำดับสองมิตินี้เป็นแถวลำดับมิติเดียว ซึ่งทำให้สามารถเข้าถึงข้อมูลได้เร็วขึ้น

ตัวอย่างที่ 9.2 แสดงโปรแกรม 'init.c' ซึ่งเรียกใช้ฟังก์ชัน 'init_2d' ในการกำหนดค่าเริ่มต้น '0' ให้กับสมาชิกทุกตัวของแถวลำดับสองมิติ 'table'

```
/* init.c
 * Initialize two-d array to zero
 */

#include    <stdio.h>
#include    "init.h"

main()
{
  int table[MAXROWS][MAXCOLS];

  printf("Initiualizing ... \n");
  init_2d(table, 0);
  printf(" ... done\n");

  return 0;
}
```

```
/*
 * init.h -- Header file for program to initialize two-d array
 */

#define MAXROWS 150
#define MAXCOLS 200

extern void init_2d(int table[ ][MAXCOLS], int value);
```

ตัวอย่างที่ 9.3 โปรแกรม 'init1.c' ซึ่งบรรจุฟังก์ชัน 'init_2d' โดยใช้ดัชนี กำหนดเลขที่อยู่ของสมาชิกของแถวลำดับแต่ละตัว

```
/* init1.c
 * Array-substring version.
 */
#include "init.h"
void init_2d(int table[ ][MAXCOLS], int value)

{
  int  x, y;

  for ( x = 0; x < MAXROWS; x++)
    for (y = 0; y < MAXCOLS; y++)
      table[x][y] = value;
}
```

ตัวอย่างที่ 9.4 โปรแกรม 'init2.c' ซึ่งบรรจุฟังก์ชัน 'init_2d' โดยใช้ตัวชี้ เพื่อเข้าถึงสมาชิกของแถวลำดับแต่ละตัว

```
/* init2.c
 * Pointer-indexing version.
 */
#include "init.h"
void init_2d(int table[ ][MAXCOLS], int value)
{
     int  *ptr = &table[0][0];
     int  *endptr = &table[MAXROWS - 1][MAXCOLS - 1];

     for ( ; ptr <= endptr; *ptr++ = value)
}
```

จากโปรแกรมจะเห็นว่า 'ptr' เป็นตัวชี้ ซึ่งถูกกำหนดให้เป็นเลขที่อยู่ของสมาชิกตัวแรกของแถวลำดับดังคำสั่ง

```
ptr =&table[0][0];
```

หลังจากนั้นกำหนดให้ค่าข้อมูล ณ ตำแหน่งที่ 'ptr' ชี้เป็น '0' แล้วเพิ่มค่า 'ptr' ทีละ '1' ด้วยคำสั่ง

```
*ptr++ = value;
```

แล้ว กำหนดค่าข้อมูล ณ ตำแหน่งที่ 'ptr' ชี้เป็น '0' และเพิ่มค่า 'ptr' ทำเช่นนี้เรื่อย ๆ ไปจนกระทั่ง 'ptr' ชี้ไปยังสมาชิกตัวสุดท้ายของแถวลำดับที่มีเลขที่อยู่เป็น

```
&table[MAXROWS - 1][MAXCOLS - 1];
```

9.1.4 การกำหนดค่าเริ่มต้นของแถวลำดับสองมิติ

การกำหนดค่าเริ่มต้นของตัวแปรแถวลำดับสองมิติ สามารถกำหนดไว้ในบรรทัดเดียวกันกับคำสั่งประกาศตัวแปรได้ ซึ่งสามารถกำหนดได้ 2 แบบ ดังนี้

(1) ประกาศตัวแปรแถวลำดับสองมิติ แล้วตามด้วยรายการค่าเริ่มต้นของสมาชิกซึ่งเขียนไว้ภายในวงเล็บปีกกา และคั่นระหว่างค่าของสมาชิก แต่ละตัวด้วยเครื่องหมายจุลภาค

ตัวอย่างเช่น

```
int     square[3][3] = {1, 1, 1, 2, 2, 2, 3, 3, 3 }
```

ในที่นี้กำหนดให้

ค่าของสมาชิกในแถวที่	0	เป็น	1	ทุกตัว
ค่าของสมาชิกในแถวที่	1	เป็น	2	ทุกตัว
และค่าของสมาชิกในแถวที่	2	เป็น	3	ทุกตัว ตามลำดับ

จะเห็นว่าการกำหนดค่าเริ่มต้นวิธีนี้ จะทำให้มองเห็นภาพไม่ชัดเจนว่ามีค่าใดถูกเก็บอยู่ในแถวใดบ้าง โดยเฉพาะเมื่อแถวลำดับมีจำนวนแถวและจำนวนสดมภ์มาก

(2) เขียนวงเล็บปีกกาล้อมรอบกลุ่มของค่าของสมาชิกในแต่ละแถวของแถวลำดับสองมิติ วิธีนี้จะทำให้มองเห็นภาพชัดเจนยิ่งขึ้น ดังตัวอย่าง

```
int     square[3][3] = {{1, 1, 1}, {2, 2, 2}, {3, 3, 3}};
```

เนื่องจากว่า คอมไพเลอร์ภาษา C สามารถกำหนดขนาดและโครงสร้างของแถวลำดับจากค่าที่กำหนดให้ได้ ดังนั้นในการประกาศไม่จำเป็นต้องระบุดัชดีตัวแรกของแถวลำดับ (จำนวนแถว) ก็ได้

นั่นคือ เราสามารถประกาศตัวแปร 'square' ได้อีกแบบหนึ่งดังนี้

```
int     square[ ][3] = {{1, 1, 1}, {2, 2, 2}, {3, 3, 3}};
```

ถ้ามีค่าใดค่าหนึ่งขาดหายไปคอมไพเลอร์จะกำหนดให้มีค่าเริ่มต้น เป็น '0' เช่น ถ้าประกาศตัวแปร 'square' ดังต่อไปนี้

```
int     square[3][3] = {{1, 1}, {1}};
```

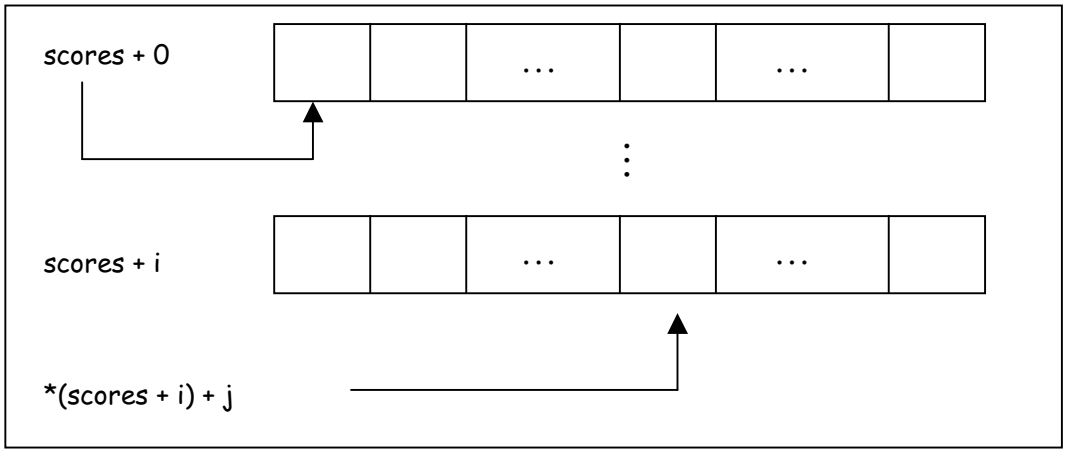
คอมไพเลอร์จะกำหนดให้สมาชิก 2 ตัวแรกของแถวที่ 0 และ สมาชิกตัวแรกของแถวที่ 1 มีค่าเป็น '1' ส่วนสมาชิกอื่น ๆ มีค่าเป็น '0'

9.2 ตัวชี้และแถวลำดับสองมิติ

ชื่อของแถวลำดับไม่ว่าจะเป็นแถวลำดับมิติเดียว หรือ แถวลำดับสองมิติ จะหมายถึงเลขที่อยู่ของสมาชิกของแถวลำดับนั้นเสมอ ดังนั้น เมื่อมีการอ้างอิงถึงชื่อของแถวลำดับตามด้วยดัชนีเพื่อระบุตำแหน่งของสมาชิก คอมไพเลอร์ภาษา C จะแปลการอ้างอิงนั้นให้อยู่ในรูปของนิพจน์ตัวชี้ที่สมนัยกันเสมอ

ตัวอย่างเช่น เมื่อกำหนด	scores[i][j]
นิพจน์ จะถูกแปลงให้อยู่ในรูป	*(*(scores + i) + j)

ซึ่งแผนภาพแสดงตำแหน่งของ scores[i][j] ปรากฏในรูปที่ 9.3 ต่อไปนี้



รูปที่ 9.3 แสดงการใช้ *(*(scores + i) + j) เพื่อเข้าถึง scores[i][j]

9.3 การท่องเข้าไปในสดมภ์ของแถวลำดับ

การท่องเข้าไปในสดมภ์ของโครงสร้างแถวลำดับนั้นสามารถทำได้ 2 วิธี คือ ใช้ดัชนีและตัวชี้ ดังในตัวอย่าง ที่ 9.5 – 9.7 ต่อไปนี้

ตัวอย่างที่ 9.5 โปรแกรม 'testavg.c' แสดงการคำนวณค่าเฉลี่ยของคะแนนสอบของแต่ละวิชา ซึ่งถูกจัดเก็บอยู่ในสดมภ์ต่าง ๆ ของแถวลำดับ 'scores'

```
/* Program name:  testavg.c
 *            Read scores and print average on each test
 */

#include        <stdio.h>
#define         MAXLEN 80
#define         MAX_TESTS 3
#define         MAX_STUDENTS 3



main()
{
        int scores[MAX_STUDENTS][MAX_TESTS];
        int i;
        double test_avg(int s[ ][MAX_TESTS], int rows, int n);
        int read_scores(int s[ ][MAX_TESTS], int maxrow);


        if (read_scores(scores, MAX_STUDENTS) == MAX_STUDENTS)
                for (i = 0; i < MAX_TESTS; i++)
                        printf("Average on test  %d is  %6.2f\n", i+1,
                                test_avg(scores, MAX_STUDENTS, i));
         else
        printf("Couldn't read all students successfully\n");
         return 0;
}

int read_scores(int s[ ][MAX_TESTS], int maxrow)
{
         int student, test;
         char line[MAXLEN + 1];

         for (student = 0; student < maxrow; student++)
         {
                printf("student %d:\n",student+1);
                for (test =0; test < MAX_TESTS; test++)
                {
                    printf("\tscores on test %d =",test+1);
                    scanf("%d", &s[student][test]) ;
                }
        }
        printf("************************************\n");
        return maxrow;                  /* read all scores we're supposed to */
}
```

โปรแกรม 'testavg.c' ในตัวอย่างที่ 9.5 นั้นเรียกใช้ฟังก์ชัน 'test_avg' ซึ่งได้กำหนดการท่องเข้าในสดมภ์ที่ต้องการ 2 วิธี คือ

1. วิธีแรก แสดงในตัวอย่างที่ 9.6 ซึ่งใช้ดัชนีของแถวลำดับ
2. วิธีที่สอง แสดงในตัวอย่างที่ 9.7 ซึ่งใช้ตัวชี้ในการท่องเข้าไปในแถวลำดับ

ตัวอย่างที่ 9.6 ฟังก์ชันแสดงการคำนวณค่าเฉลี่ยของคะแนนสอบของแต่ละวิชา โดยการท่องเข้าไปในสดมภ์โดยใช้ดัชนี

```
/*
 * Array-subscripting version of our function to compute
 * the average of a column in a 2d array.
 */

#define MAX_TESTS 3

double test_avg(int s[ ][MAX_TESTS], int rows, int n)

{
  double sum = 0.0;                    /* column total  */
  int i;                            /* will point to the last row */

  for (i = 0; i < rows; i++)
    sum += s[i][n];
  return sum / rows;

}
```

ผลลัพธ์ที่ได้คือ

```
student 1:
        scores on test 1 =90
        scores on test 2 =86
        scores on test 3 =70
student 2:
        scores on test 1 =50
        scores on test 2 =42
        scores on test 3 =70
student 3:
        scores on test 1 =99
        scores on test 2 =88
        scores on test 3 =75
************************************
Average on test  1 is   79.67
Average on test  2 is   72.00
Average on test  3 is   71.67
```

ตัวอย่างที่ 9.7 ฟังก์ชันแสดงการคำนวณค่าเฉลี่ยของคะแนนสอบของแต่ละวิชา โดยการท่องเข้าไปสดมภ์โดยใช้ตัวชี้

```
/*
 *          Pointer-indexing version of our function to compute
 *          the average of a column in a 2d array.
 */

#define MAX_TESTS 3

double test_avg(int (*rowptr)[MAX_TESTS], int rows, int n)

{
  double sum = 0.0;                    /* column total */
  int (*endptr)[MAX_TESTS];            /* will point to the last row */
  for (endptr = rowptr + rows; rowptr < endptr; rowptr++)
    sum += (*rowptr)[n];
  return sum / rows;
}
```

ผลลัพธ์ที่ได้ เหมือนกับผลลัพธ์ในตัวอย่างที่ 9.6

เมื่อประกาศแถวลำดับสองมิติ เราสามารถใช้ชื่อของตัวแปรเป็นตัวชี้ชี้ไปที่แถวแรกของแถวลำดับสองมิตินั้นได้เสมอ เช่น จากตัวอย่างที่ 9.5 เมื่อประกาศคำสั่ง

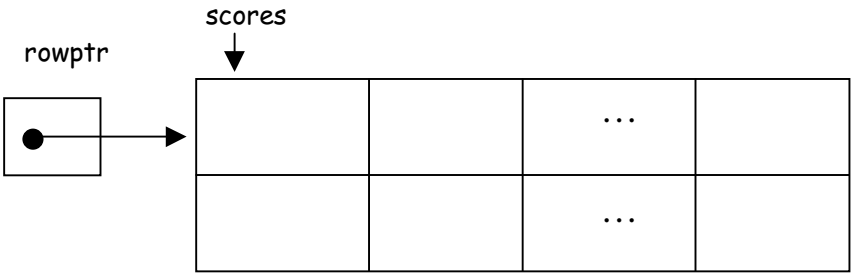
```
int  scores[MAX_STUDENTS][MAX_TESTS];
```

คอมไพเลอร์จะทำการจัดสรรเนื้อที่สำหรับแถวลำดับสองมิติ และกำหนดให้ 'scores' เป็นตัวชี้ชี้ไปที่แถวแรกของแถวลำดับนั้น เมื่อเราส่งตัวแปร 'scores' เป็นพารามิเตอร์ไปยังฟังก์ชัน 'test_avg' นั้น เราเพียงแต่ส่งตัวชี้ชี้ไปยังแถวแรกของแถวลำดับ 'scores' เท่านั้น และในฟังก์ชัน 'test_avg' ซึ่งใช้ตัวชี้ในการรับค่าพารามิเตอร์จะประกาศพารามิเตอร์ดังปรากฏในตัวอย่างที่ 9.7 โปรแกรม 'tstavg2.c' ดังนี้

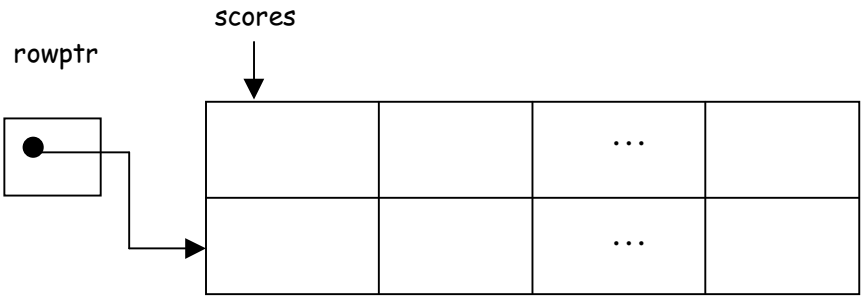
```
int   (*rowptr) [MAX-TESTS]
```

ในที่นี้ '*rowptr' เป็นตัวชี้ชี้ไปที่แถวแรกของแถวลำดับ 'scores' ซึ่งจะเห็นว่า เราสามารถใช้ '*rowptr' เหมือนกับเป็นชื่อของแถวลำดับ ดังนั้น (*rowptr)[n] คือ สมาชิกตัวที่ n ในแถวที่ rowptr

ชี้อยู่ ณ ขณะนั้น นอกจากนี้ การเพิ่มค่า rowptr ขึ้นอีก 1 นั้น rowptr จะชี้ไปที่แถวถัดไปของแถวลำดับ 'scores' ดังปรากฏในรูปที่ 9.4



รูปที่9.4 (a) เมื่อเริ่มต้น rowptr จะชี้ไปที่แถวแรกของแถวลำดับ scores



รูปที่ 9.4 (b) เมื่อเพิ่มค่า rowptr อีก 1 (rowptr ++) มีผลทำให้ rowptr ชี้ไปที่แถวถัดไปของแถวลำดับ scores

9.4 การท่องเข้าไปในแถวของแถวลำดับ

เราสามารถใช้ตัวชี้เพื่อช่วยให้สามารถท่องเข้าไปในแต่ละแถวของแถวลำดับได้ในทำนองเดียวกันกับการท่องเข้าไปในสตมภ์

โปรแกรม 'rowavg.c' ในตัวอย่างที่ 9.8 เรียกใช้ฟังก์ชัน 'student_avg' ซึ่งทำการคำนวณค่าเฉลี่ยของคะแนนสอบของนักเรียนแต่ละคน (ค่าเฉลี่ยในแต่ละแถว) โดยฟังก์ชัน'student_avg' นั้นสามารถกำหนดได้โดยใช้ดัชนีของแถวลำดับในการเข้าถึงสมาชิกในแต่ละแถว ดังในตัวอย่างที่ 9.9 โปรแกรม 'rowavg1.c' หรือใช้ตัวชี้ในตัวอย่างที่ 9.10 โปรแกรม 'rowavg2.c'

ตัวอย่างที่ 9.8 โปรแกรมคำนวณหาค่าเฉลี่ยของนักเรียนแต่ละคน

```
/* Program name:  rowavg.c
 *             Read scores and print average for each student
 */

#include <stdio.h>
#define MAX_TESTS       3
#define MAX_STUDENTS    3

main()
{
          int scores[MAX_STUDENTS][MAX_TESTS];
           int i, n;
           double student_avg(int s[ ][MAX_TESTS], int n);

           if ((n = read_scores(scores, MAX_STUDENTS)) == MAX_STUDENTS)
                    for (i = 0; i < MAX_STUDENTS; i++)
                    printf("Average for student  %d is  %.2f\n", i+1, student_avg(scores,i));
           else
                    printf("Only %d students read successfully\n", n);
  return 0;
}

int read_scores(int s[ ][MAX_TESTS], int maxrow)
{
           int student, test;

           for (student = 0; student < maxrow; student++)
           {
                    printf("Student %d\n",student+1);
                    for (test =0; test < MAX_TESTS; test++)
                             scanf("%d", &s[student][test]) ;
            }
          return maxrow;                    /* read all scores we're supposed to */

}
```

ตัวอย่างที่ 9.9 ฟังก์ชันในการคำนวณหาค่าเฉลี่ยโดยการท่องเข้าไปในแถวลำดับของดัชนี

```
/* Program name: rowavg1.c
 *            Compute row average using array-subscripting
 */
#define MAX_TESTS 3
double student_avg(int s[][MAX_TESTS], int n)
{
  int       i = 0;                  /* index to next row  */
 long       sum = 0;                /* total so far */
  for ( ; i < MAX_TESTS; sum += s[n][i++]) ;
   return (double)sum / MAX_TESTS;
}
```

ผลลัพธ์ที่ได้คือ

```
Student 1
95
62
78
Student 2
50
42
95
Student 3
74
68
81
Average for student  1 is  78.33
Average for student  2 is  62.33
Average for student  3 is  74.33
```

ตัวอย่างที่ 9.10 ฟังก์ชันในการคำนวณค่าเฉลี่ยโดยใช้ตัวชี้ท่องเข้าไปในแถวลำดับ

```
/* Program name: rowavg2.c
 *            Compute row average using array subscripting
 */
#define MAX_TESTS 3
double student_avg(int s[][MAX_TESTS],int n)
{
    int *colptr  = &s[n][0];                        /* ptr to first element */
    int *endptr  = colptr + MAX_TESTS;  /* ptr to last element */
   long sum     = 0;                       /* total so far */

  for (; colptr < endptr; sum += *colptr++);
    return (double)sum / MAX_TESTS;
}
```

ผลลัพธ์ที่ได้เหมือนกับผลลัพธ์ในตัวอย่างที่ 9.9

9.5 แถวลำดับสามมิติ

การประกาศตัวแปร และการกำหนดค่าเริ่มต้นตัวแปรแถวลำดับสามมิติทำได้เช่นเดียวกันกับประกาศและการกำหนดค่าเริ่มต้นตัวแปรแถวลำดับสองมิติ

ตัวอย่างเช่น

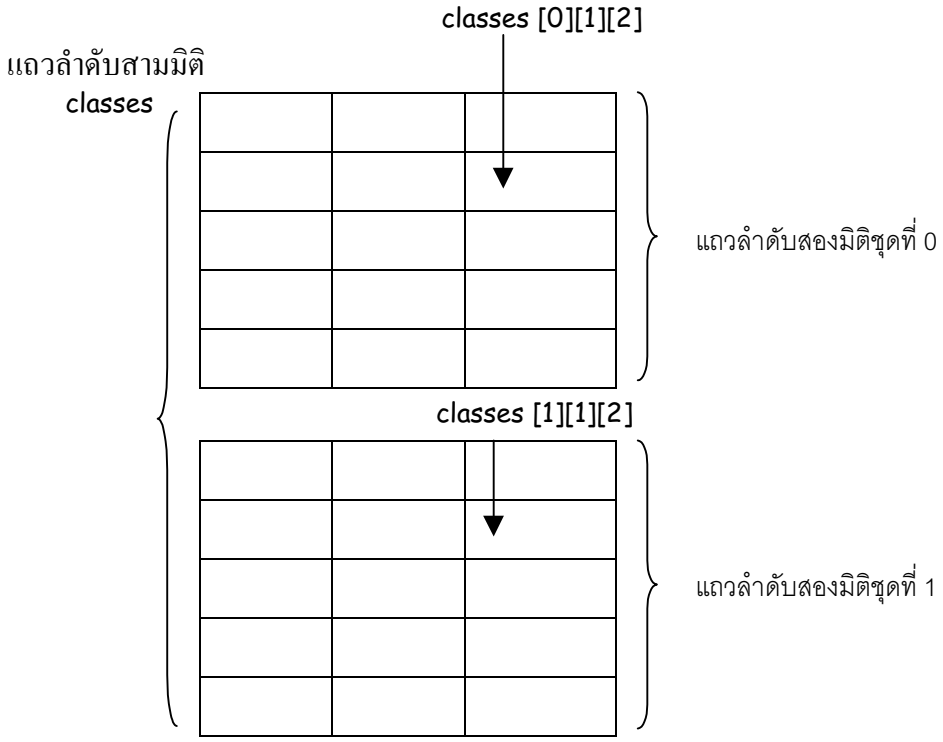
```
#define MAX_CLASSES     2
#define MAX_STUDENTS    5
#define MAX_TESTS       3
…
int     classes[MAX_CLASSES][MAX_STUDENTS] [MAX_TESTS];
```

จากตัวอย่างการประกาศข้างต้น เป็นการประกาศตัวแปรแถวลำดับสามมิติ ‘classes’ ที่มีจำนวนสมาชิก 30 ตัว ซึ่งถูกจัดเก็บอยู่ในแถวลำดับมิติ 2 แถวลำดับ โดยแต่ละแถวลำดับสองมิติขนาด 5 แถว และ 3 สดมภ์

การเข้าถึงสมาชิกแต่ละตัวของแถวลำดับ สามารถเข้าถึงได้โดยกำหนดตำแหน่งที่ระบุโดยค่าของดัชนีในมิติต่าง ๆ เช่น

```
classes[0][1][2]
```

เป็นการอ้างถึงสมาชิกในแถวลำดับชุดที่ 0 ในสดมภ์ที่ 3 แถวที่ 2 ในแถวลำดับสองมิติอันแรก ดังปรากฏในรูปที่ 9.5



รูปที่ 9.5 ภาพจำลองแถวลำดับสามมิติ classes

แบบฝึกหัดบทที่ 9

1. จงตอบคำถามต่อไปนี้ เมื่อกำหนดให้ 'table' เป็นชื่อของแถวลำดับสองมิติ
   ก. ประกาศตัวแปรแถวลำดับ 'table' เพื่อเก็บจำนวนเต็มที่มีสามแถว และสามสดมภ์ โดยสมมติว่าค่าคงตัว 'SIZE' ถูกกำหนดเป็น 3
   ข. แถวลำดับ 'table' มีจำนวนสมาชิกทั้งหมดกี่ตัว
   ค. จงเขียนส่วนของโปรแกรมเพื่อกำหนดค่าเริ่มต้นให้สมาชิกแต่ละตัวของแถวลำดับมี ค่าเป็น x + y เมื่อ x คือ ดัชนีของแถวและ y คือดัชนีของสดมภ์
   ง. สมมติว่าแถวลำดับ 'table' ถูกกำหนดค่าเริ่มต้นในบรรทัดเดียวกับการประกาศดังนี้

```
int table[SIZE][SIZE]={{1,8},{2,4,6},{5,}};
```

   จงแจกแจงค่าของสมาชิกของแถวลำดับ 'table'

2. จงเขียนโปรแกรมเพื่อหาผลรวมของทุก ๆ สมาชิกทุกตัวที่ถูกเก็บอยู่ในแถวลำดับสองมิติใด ๆ
3. จงเขียนฟังก์ชันเพื่อตรวจสอบว่าแถวลำดับสองมิติที่มีจำนวนแถวและจำนวนสดมภ์เท่ากันนั้น เป็นเมตริกเอกลักษณ์ (สมาชิกทุกตัวบนแนวทะแยงมุมหลักมีค่าเป็น 1 ส่วนตัวอื่น ๆ เป็น 0 ทั้งหมด)
4. จงเขียนฟังก์ชัน '`sort_scores`' ซึ่งมีอากิวเมนต์ 2 ตัว คือ '`scores`' ซึ่งเป็นตัวแปรแถวลำดับสองมิติที่ใช้เก็บจำนวนเต็ม (int) และ '`n`' ซึ่งเป็นเลขที่ของสดมภ์ที่ต้องการเรียงลำดับคะแนน เมื่อฟังก์ชันทำงานเสร็จสิ้น คะแนนในสดมภ์ที่ '`n`' ต้องเรียงลำดับจากน้อยไปหามาก