w13-Lab

2D Array

Assembled for 204111

by Ratsameetip Wita

Two-Dimensional Arrays (2D-Arrays)

- **2DArray หรืออาเรย์แบบสองมิติ จะมีการเก็บข้อมูลในลักษณะคล้ายตาราง**
  - **มีการอ้างอิงในรูปแบบ แถว และหลัก (Row/Column)**
  - **ในการประกาศตัวแปร คล้ายกับการประกาศอาเรย์ 1 มิติ แต่มีการระบุจำนวน Row และ Column ใน [][] ดังนี้**
    - **int 2Darray[row][column];**

Two-Dimensional Arrays (2D-Arrays)[2]

- **เช่น int val[3][4]; เป็นการประกาศตัวแปรชื่อ val เป็นอาเรย์แบบสองมิติ ที่มี 3 แถว และ 4 หลัก**

เริ่มนับที่ แถวที่ 0 และหลักที่ 0 เสมอ

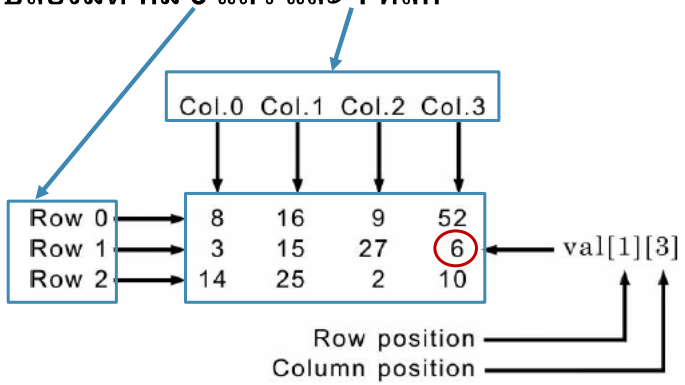


Figure 8.9 Each array element is identified by its row and column

2D Array Initialization

- **Initialization**

ระบุสมาชิกในอาเรย์ เรียงลำดับตามแถว ระบุโดย { {แถว 1}, {แถว2}, {แถว 3} }

```
#define NUMROWS 3
#define NUMCOLS 4
int val[NUMROWS][NUMCOLS] = { {8,16,9,52},
                              {3,15,27,6},
                              {14,25,2,10} };
```

- **The inner braces can be omitted:**

```
int val[NUMROWS][NUMCOLS] =
    {8,16,9,52,3,15,27,6,14,25,2,10};
```

val[0][] val[1][] val[2][]

- **Initialization is done in row order**

Initialization Order

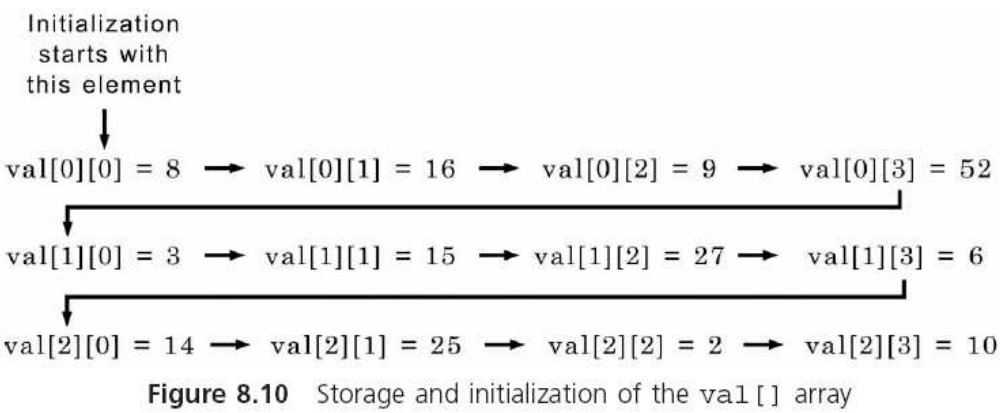


Figure 8.10 Storage and initialization of the `val[]` array


```c
#include <stdio.h>
int main(){
#define NUMROWS 3
#define NUMCOLS 4

int val[NUMROWS][NUMCOLS] = {8,16,9,52,3,15,27,6,14,25,2,10};
int i, j;
printf("Display val array by explicit element\n");
printf("%2d %2d %2d %2d\n", val[0][0],val[0][1],val[0][2],val[0][3]);
printf("%2d %2d %2d %2d\n", val[1][0],val[1][1],val[1][2],val[1][3]);
printf("%2d %2d %2d %2d\n", val[2][0],val[2][1],val[2][2],val[2][3]);

printf("Display val array using nested loop\n");
for(i=0;i< NUMROWS;i++){    //วนทีละแถว
   printf("\n");
    for(j=0;j<NUMCOLS;j++){   //พิมพ์ทีละหลักในแถวนั้น ๆ
        printf("%2d", val[i][j]);
    }
}
printf("\n");
return 0;
}
```

```
Display of val array by explicit
 element
8 16 9 52
3 15 27 6
14 25 2 10

Display of val array using a nested
 for loop
8 16 9 52
3 15 27 6
14 25 2 10
```



```c
int val[NUMROWS][NUMCOLS] = {8,16,9,52,3,15,27,6,14,25,2,10};

for(i=0;i< NUMROWS;i++){    //วนทีละแถว
   printf("\n");
   for(j=0;j<NUMCOLS;j++){   //พิมพ์ทีละหลักในแถวนั้น ๆ
     printf("%2d", val[i][j]);
   }
}
```

```
NUMROWS = 3
NUMCOLS = 4

 8  16  9  52
 3  15  27  6
14  25  2  10
```

# of Loop	i	j	val[i][j]
1	0	0	8
2	0	1	16
3	0	2	9
4	0	3	52
5	1	0	3
6	1	1	15
7	1	2	27
8	1	3	6
9	2	0	14
10	2	1	25
11	2	2	2
12	2	3	10

2D Array Address Ordering

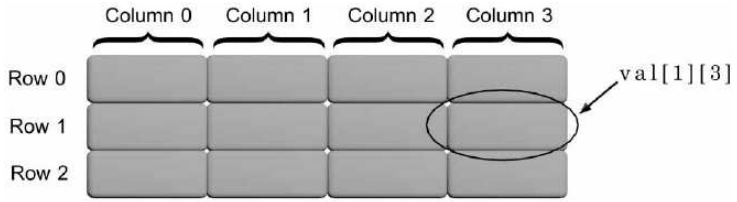


Figure 8.11 Storage of the `val` array

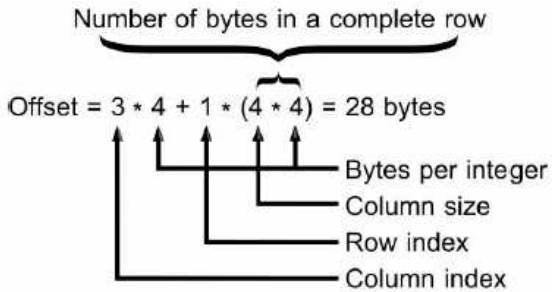


Figure 8.12 Determining an element's offset

การหาตำแหน่งของ **element** ใน **array 2** มิติ

ตัวอย่าง การหาตำแหน่งของ **val[1][3]**

n-dimension Array

- ตัวแปรแบบอาเรย์ สามารถกำหนดจำนวนมิติ ได้มากกว่า 2 โดยมีการประกาศตัวแปรในรูปแบบดังนี้

```
int 3Darray[10][10][10]; // 3D-Array
double records[100][250][250][250] //4D-Array
```

- โดยในการพิจารณาอาเรย์แบบหลายมิติ จะมีการมองข้อมูลเป็นเซ็ตของตาราง

Function Calling with Arrays (Revisited)

- ในภาษา C เราสามารถ ส่งผ่าน array เป็นค่า parameter ไปยัง function ได้ดังนี้
  - void printArray(char s1[],…)
  - void printArray(char s1[30],…)
  - void printArray(char *s1,…) //204112
- ทั้งสามวิธีเป็นการส่ง array ไปยัง function ในรูปแบบการส่ง memory address ทั้งสิ้น

Function Calling with 2DArrays

- ในการส่งอาเรย์สองมิติ เป็นอากิวเมนต์ของฟังก์ชัน จะมีรูปแบบของการใช้งานดังนี้

```
void readarray(int a[ROW][COL], int m, int n);
                        หรือ
void readarray(int a[][COL], int m, int n);
```

- **ROW** และ **COL** คือขนาดของอาเรย์ที่จองไว้
- **m , n** คือจำนวนสมาชิกที่มีการเก็บข้อมูลจริงในอาเรย์

Practice I

- ให้เขียนโปรแกรมเพื่อรับ Matrix ขนาด m x n 2 ชุด (1≤m≤ 20) (1≤n≤ 20) และแสดงผลบวกของ Matrix โดยรับค่า Input ผ่าน IO Redirection และแสดงผลลัทธ์ที่หน้าจอ

data.txt

```
row    3
col    4
matrix 1
       1 2 3 4
       5 6 7 8
       9 10 11 12
matrix 2
       10 11 12 13
       14 15 16 17
       18 19 20 21
```

Output

```
11 13 15 17
19 21 23 25
27 29 31 33
```

Practice II

- ให้เขียนโปรแกรมเพื่อแสดง **transpose** ของ **Matrix** ขนาด **m x n (1≤m≤ 20) (1≤n≤ 20)** โดยรับค่า **Input** ผ่าน **IO Redirection** และแสดงผลลัทธ์ที่หน้าจอ

data.txt

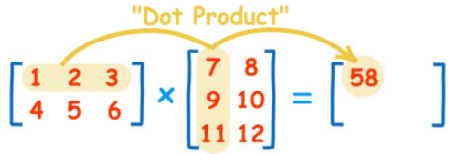
```
row      3
col      4
matrix 1 1 2 3 4
         5 6 7 8
         9 10 11 12
```

Output

```
1 5 9
2 6 10
3 7 11
4 8 12
```


Practice III : Matrix dot product

- ให้เขียนโปรแกรมเพื่อรับค่า **Matrix** ขนาด **mxn** และ **nxo** และทำการคำนวนและแสดงผลการคูณแบบ **dot product** ของ **Matrix** โดยรับค่า **Input** ผ่าน **IO Redirection** และแสดงผลลัทธ์ที่หน้าจอ



data.txt

```
row col of A 3 2
row col of B 2 3
matrix A     1 2 3
             4 5 6
matrix B     7 8
             9 10
             11 12
```

Output

```
58 64
139 154
```