02204171 Structured Programming

Chapter 10 : Two-dimensional Arrays

Outline

- Review: 1 Dimensional Array
- 2-Dimensional Array
  - Overview
  - Declaration & Initialization
  - Accessing Element of 2D-Array
  - Accessing Element of 2D-Array by using Nested Loop
- Arrays of Strings

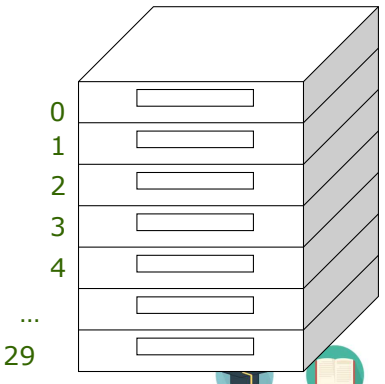
Review: 1 Dimensional Array

- **อาเรย์ (Arrays)** ตัวแปรชุด ที่เก็บข้อมูล**ประเภทเดียวกัน**

อาเรย์ช่วยอำนวยความสะดวกในการประกาศและใช้งานตัวแปรจำนวนมาก

ตัวอย่าง ประกาศตัวแปร array ชนิดจำนวนเต็ม 30 ตัว

```
int score[30];
```



Review: 1 Dimensional Array

- **อาเรย์เก็บคะแนนวิชา C programming ของนิสิต 30 คน**

```
int score[30];
```

- **แสดงค่าทั้งหมดในอาเรย์ Score**

```
int i;
for (i=0;i<30;i++)
    printf("%d\n",score[i]);
```

- **บวกผลรวมคะแนนทั้งหมดในอาเรย์ Score**

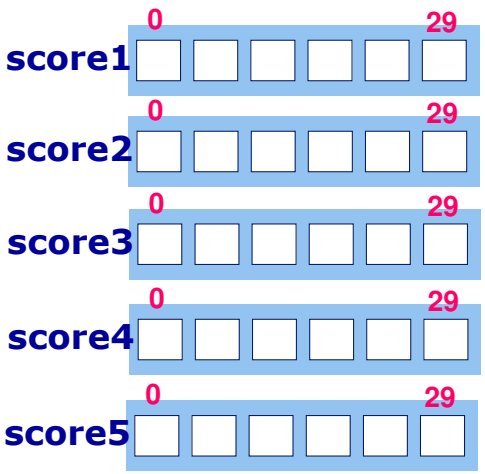
```
int sum=0;
for (i=0;i<30;i++)
        sum+=score[i];
printf("%d\n",sum);
```

ถ้าต้องการเก็บคะแนนวิชาอื่น ๆ เพิ่มอีก 4 วิชา ต้องทำอย่างไร

Review: 1 Dimensional Array

- อาเรย์เก็บคะแนน วิชา C, Eng, Chem, Phy, Math ของนิสิต 30 คน

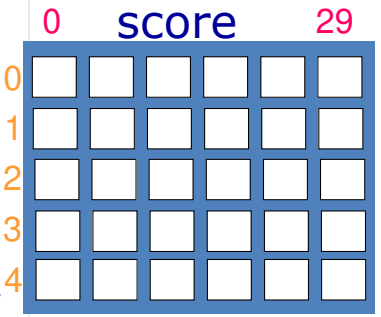
```
int score1[30],score2[30],
score3[30],score4[30],
score5[30];

for (i=0;i<30;i++) {
  scanf("%d",&score1[i]);
  scanf("%d",&score2[i]);
  scanf("%d",&score3[i]);
  scanf("%d",&score4[i]);
  scanf("%d",&score5[i]);
}
```



Overview: 2 Dimensional Array

- อาเรย์เก็บคะแนน วิชา C, Eng, Chem, Phy, Math ของนิสิต 30 คน

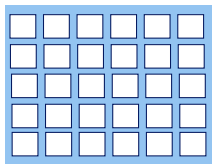
```
int score[5][30];
int i,j;

for (i=0;i<5;i++)
{
   for (j=0;j<30;j++)
    scanf("%d",&score[i][j]);
}
```



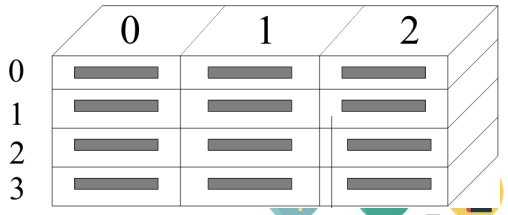
อาเรย์ 2 มิติ เก็บข้อมูลในรูปแบบ**ตาราง**
- **แถว** แทน จำนวนวิชา
- **คอมลัมภ์** แทน จำนวนนิสิต

Overview: 2 Dimensional Array

- **อาเรย์ 2 มิติ ตัวแปรชุดที่เก็บข้อมูลชนิดเดียวกันทั้งหมดในลักษณะตาราง**



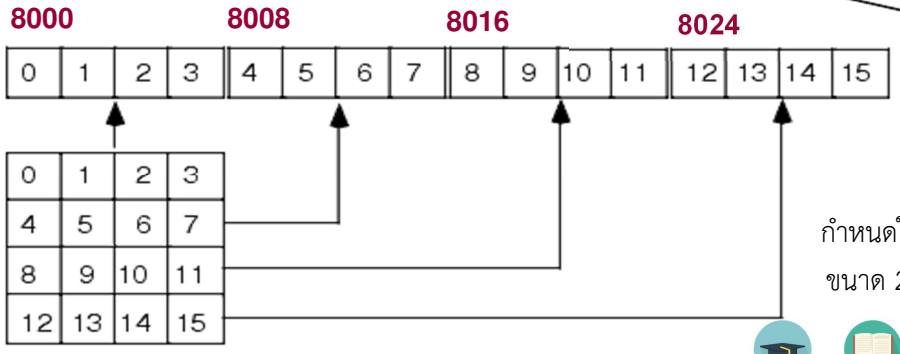
- **ตัวอย่าง:** การใช้ อาเรย์ 2 มิติเก็บข้อมูล
  - การเก็บข้อมูลคะแนน 5 วิชาของนักเรียน 30 คน
  - การเก็บข้อมูลของเมตริกซ์เพื่อใช้ในการคำนวณและการแก้สมการ
  - การเก็บข้อมูลการจองที่นั่งในโรงหนัง



Overview: 2 Dimensional Array

- **การเก็บอาเรย์ 2 มิติ ในหน่วยความจำ**

ในหน่วยความจำ ภาษา C เก็บอาเรย์ตามแนวแถว นั่นคือ แถวแรก ตามด้วยแถวที่สองจนถึงแถวสุดท้ายตามลำดับ



กำหนดให้ int มีขนาด 2 bytes

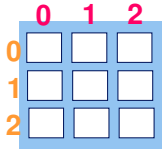
1. 2D-Array Declaration and Initialization

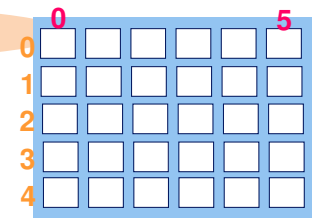
- ไวยกรณ์ (Syntax):

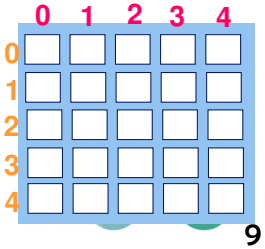
ชนิดข้อมูล ชื่อตัวแปร[จำนวนแถว] [จำนวนหลัก];

int char float double bool

- ตัวอย่าง
  - int a[5][6];
    //สมาชิก 30 ตัว
  - double b[3][3];
    //สมาชิก 9 ตัว
  - char c[5][5];
    //สมาชิก 25 ตัว







1. 2D-Array Declaration and Initialization

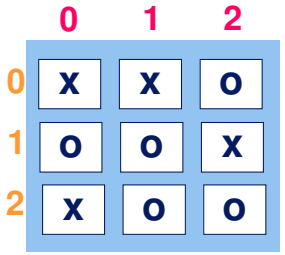
- ไวยกรณ์ (Syntax):

คั่นแต่ละค่าด้วย comma

ชนิดข้อมูล ชื่อตัวแปร[จำนวนแถว][จำนวนหลัก]={{ค่าที่11,ค่าที่12,..}
,{ค่าที่21,ค่าที่22,..}};

- ตัวอย่าง

```
char ticTacToeBoard[3][3]= {{'x', 'x', 'o'},
                            {'o', 'o', 'x'},
                            {'x', 'o', 'o'} };
```

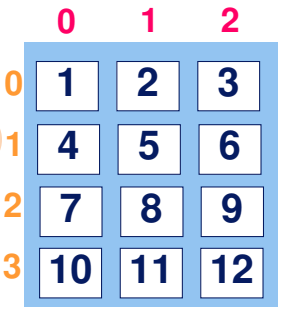


1. 2D-Array Declaration and Initialization

- การประกาศอาเรย์ พร้อมกำหนดค่าเริ่มต้น (เพิ่มเติม)

```
int A[][3] = {
 {1, 2, 3},
 {4, 5, 6},
 {7, 8, 9},
 {10,11,12}};
```

หากไม่มีการกำหนดจำนวนแถว compiler จะใช้จำนวนค่าเริ่มต้น เป็นจำนวนแถว




```
int B[2][3]= {{4},
              {6,7}};
```

ค่าเริ่มต้นของสมาชิกที่เหลือจะมีค่าเป็น 0

```
int C[2][3]= {{4,0,0},
              {6,7,0}};
```

Quick Check1

- จงหาจำนวนสมาชิกของอาเรย์ต่อไปนี้
  - int x[98];
  - int y[20][10];
  - int z[100][2];
- หากประกาศ อาเรย์ w ดังนี้ w[2][1] มีค่าเท่าไร

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

Quick Check1

- จงวาดโครงสร้างของอาเรย์ พร้อมทั้งเขียนค่าที่จัดเก็บในสมาชิกแต่ละตัว เมื่อประกาศอาเรย์ matrix ดังนี้

```
int matrix[5][5] = {
  {1, 2, 3, 4, 5},
  {2, 3, 4, 5},
  {3, 4, 5},
  {4, 5},
  {5}  };
```


2. Accessing Element of 2D-Array

- การเข้าถึงตัวแปรอาเรย์ 2 มิติ จะต้องระบุ **ดัชนี** ของสมาชิกอาเรย์ ทั้งแนว **แถว** และ **คอลัมน์**

	0	1	2
0	b[0][0]	b[0][1]	b[0][2]
1	b[1][0]	b[1][1]	b[1][2]
2	b[2][0]	b[2][1]	b[2][2]

- ตัวอย่าง 1:

```
double b[3][3]={0};
b[0][0] = 10.2;
b[1][1] = 12.0;
b[2][2] = 7.8;

b[0][2] = b[1][1]+b[2][2];
b[2][0] = b[0][0]+b[2][2];

printf( "%.2f",b[2][0]+b[1][1]);
```

	0	1	2
0	0.0	0.0	0.0
1	0.0	0.0	0.0
2	0.0	0.0	0.0

2. Accessing Element of 2D-Array

- ตัวอย่าง 2:

```
int a[2][10],b[5][2];
int x,y,c;
for(x=0;x<2;x++)
    b[x][x]=x;

for(y=0;y<10;y++)
   a[x-1][y]=x*y;

c=a[1][5]+b[1][1];

printf("%d",c);
```

x

0
1
2

b[0][0] = 0

b[1][1] = 1

Array: b

0	
	1

Array: a

0	2	4	6	8	10	12	14	16	18

2. Accessing Element of 2D-Array

- ตัวอย่าง 3: การบวก Matrix

$$A=\begin{bmatrix}5 & 0\\ 1 & -2\end{bmatrix}$$

$$B=\begin{bmatrix}10 & 6\\ 1 & 2\end{bmatrix}$$

```
int A[2][2]= {{5},
              {1,-2}};

int B[2][2]= {{10, 6},
              {1, 2}};

printf("%d %d\n", A[0][0] + B[0][0], A[0][1] + B[0][1]);

printf("%d %d\n", A[1][0] + B[1][0], A[1][1] + B[1][1]);
```

3. Accessing Element of 2D-Array by using Nested Loop

- ตัวอย่าง 4: การบวก Matrix

$$\begin{pmatrix} a_{00} & a_{01} & a_{02} & a_{03} & a_{04} \\ a_{10} & a_{11} & a_{12} & a_{13} & a_{14} \\ a_{20} & a_{21} & a_{22} & a_{23} & a_{24} \\ a_{30} & a_{31} & a_{32} & a_{33} & a_{34} \end{pmatrix} + \begin{pmatrix} b_{00} & b_{01} & b_{02} & b_{03} & b_{04} \\ b_{10} & b_{11} & b_{12} & b_{13} & b_{14} \\ b_{20} & b_{21} & b_{22} & b_{23} & b_{24} \\ b_{30} & b_{31} & b_{32} & b_{33} & b_{34} \end{pmatrix} = \begin{pmatrix} a_{00}+b_{00} & a_{01}+b_{01} & a_{02}+b_{02} & a_{03}+b_{03} & a_{04}+b_{04} \\ a_{10}+b_{10} & a_{11}+b_{11} & a_{12}+b_{12} & a_{13}+b_{13} & a_{14}+b_{14} \\ a_{20}+b_{20} & a_{21}+b_{21} & a_{22}+b_{22} & a_{23}+b_{23} & a_{24}+b_{24} \\ a_{30}+b_{30} & a_{31}+b_{31} & a_{32}+b_{32} & a_{33}+b_{33} & a_{34}+b_{34} \end{pmatrix}$$

$$\begin{pmatrix} 1 & 2 & 3 & 4 & 5 \\ 6 & 7 & 8 & 9 & 10 \\ 1 & 2 & 3 & 4 & 5 \\ 6 & 7 & 8 & 9 & 10 \end{pmatrix} + \begin{pmatrix} 1 & 1 & 1 & 1 & 1 \\ 2 & 2 & 2 & 2 & 2 \\ 3 & 3 & 3 & 3 & 3 \\ 4 & 4 & 4 & 4 & 4 \end{pmatrix} = \begin{pmatrix} 1+1 & 2+1 & 3+1 & 4+1 & 5+1 \\ 6+2 & 7+2 & 8+2 & 9+2 & 10+2 \\ 1+3 & 2+3 & 3+3 & 4+3 & 5+3 \\ 6+4 & 7+4 & 8+4 & 9+4 & 10+4 \end{pmatrix}$$

3. Accessing Element of 2D-Array by using Nested Loop

- ตัวอย่าง 4: การบวก Matrix

```
int i,j;
int a[4][5],b[4][5],c[4][5];

//assign a,b as value defined in
previous slide

for(i=0;i<4;i++)
{
  for(j=0;j<5;j++)
    c[i][j]=a[i][j]+b[i][j];
}
```

จำนวนรอบของลูปนอก เท่ากันกับ จำนวนแถว

จำนวนรอบของลูปใน เท่ากันกับ จำนวนคอลัมภ์

การดำเนินการกับอาเรย์ 2 มิติ ทำได้โดยการใช้คำสั่งลูปซ้อนลูป(Nested Loop)

โดยกำหนดให้
- ตัวนับของลูปนอก เป็นดัชนีในแนวแถวของอาเรย์
- ตัวนับของลูปใน เป็นดัชนีในแนวคอลัมภ์ของอาเรย์

3. Accessing Element of 2D-Array by using Nested Loop

- ตัวอย่าง 4: การบวก Matrix

```
int i,j;
int a[4][5],b[4][5],c[4][5];
//assign a,b as value defined in
previous slide

for(i=0;i<4;i++)
{
  for(j=0;j<5;j++)
    c[i][j]=a[i][j]+b[i][j];
}
```

$$c=\begin{pmatrix} 2 & 3 & 4 & 5 & 6 \\ 8 & 9 & 10 & 11 & 12 \\ 4 & 5 & 6 & 7 & 8 \\ 10 & 11 & 12 & 13 & 14 \end{pmatrix}$$

i	j	c[i][j]
0	0	C[0][0]=1+1
	1	C[0][1]=2+1
	2	C[0][2]=3+1
	3	C[0][3]=4+1
	4	C[0][4]=5+1
…	…	…
3	0	C[3][0]=6+4
	1	C[3][1]=7+4
	2	C[3][2]=8+4
	3	C[3][3]=9+4
	4	C[3][4]=10+4

3. Accessing Element of 2D-Array by using Nested Loop

- ตัวอย่าง 5: กำหนดให้สมาชิกทุกตัวในในอาเรย์ A มีค่า เป็น 1

```
int i,j;
int A[3][3];

for(i=0;i<3;i++)
{
  for(j=0;j<3;j++)
    A[i][j]=1;
}
```

	0	1	2
0	1	1	1
1	1	1	1
2	1	1	1

i	j	A[i][j]
0	0	A[0][0]=1
	1	A[0][1]=1
	2	A[0][2]=1
1	0	A[1][0]=1
	1	A[1][1]=1
	2	A[1][2]=1
2	0	A[2][0]=1
	1	A[2][1]=1
	2	A[2][2]=1

3. Accessing Element of 2D-Array by using Nested Loop

- **ตัวอย่าง 6:** กำหนดให้สมาชิกทุกตัวในในอาเรย์ m เป็นผลคูณของดัชนีของแถว กับ ดัชนีของคอมลัมภ์

```
int i,j;
int m[3][4];
for (i=0; i<3; i++)
{
   for (j=0; j<4; j++)
      m[i][j] = i*j;
}
```

0	0	0	0
0	2	4	6

i	j	m[i][j]
0	0	m[0][0]=0
	1	m[0][1]=0
	2	m[0][2]=0
	3	m[0][3]=0
2	0	m[2][0]=0
	1	m[2][1]=2
	2	m[2][2]=4
	3	m[2][3]=6

3. Accessing Element of 2D-Array by using Nested Loop

- **ตัวอย่าง 7:** แสดงผลลัพธ์ทางจอภาพ จากส่วนของโปรแกรมนี้

```
int i,j;
char m[3][4]={{'a', 'e','i','o'},
              {'+', '-','*','/'},
              {'$','#','%','@'}};

for (i=0; i<3; i++)
{
   for (j=0; j<4; j+=2)
      printf("%c",m[i][j]);
   printf("\n");
}
```

a	e	i	o
+	-	*	/
$	#	%	@

i	j	m[i][j]
0	0	m[0][0]→ a
	2	m[0][2]→ i
1	0	m[1][0]→ +
	2	m[1][2]→*
2	0	m[2][0]→ $
	2	m[2][2]→%

Quick check2

- จงวาดโครงสร้างของอาเรย์ m พร้อมทั้งเขียนค่าที่จัดเก็บในสมาชิกแต่ละตัว เมื่อประมวลผลส่วนของโปรแกรมนี้

```
int r,c;
int m[4][4];
for (r=0; r<4; r++)
{
   for (c=0; c<4; c++)
      m[r][c] = r+c;
}
```

r	c	m[r][c]
0	0	
	1	
	2	
	3	
3	0	
	1	
	2	
	3	

Quick check 3

- ให้เขียนโปรแกรมรับค่าอุณหภูมิ หน่วย Celsius ในช่วงเวลา 8.00 น., 12.00 น. , 16.00 น. , 20.00 น. ,24.00 น. และ 4.00 น. ในเวลา 1 อาทิตย์
  - อาเรย์ 2 มิติ ประกาศขนาดเท่าใด

Quick check 3 (ต่อ)

- เขียนโปรแกรมคำนวณค่าเฉลี่ยอุณหภูมิ หน่วย Celsius ในช่วงเวลา 8.00 น.,12.00 น. , 16.00 น. , 20.00 น. ,24.00 น. และ 4.00 น. ในแต่ละวันใน 1 อาทิตย์

4. Arrays of Strings

- อาเรย์ของสายอักขระ (array of C-strings) สามารถเก็บได้โดยใช้ **อาเรย์ 2 มิติ ชนิด char**
  - **จำนวนแถว** แทน จำนวนสายอักขระ
  - **จำนวนคอลัมน์** แทน จำนวนอักขระในสายอักขระ

char name1[11]="John Snow";

J	o	h	n		S	n	o	w	\0	?

char name2[**3**][**11**]= {"John Snow", "Arya Stark", "Terion Lan" };

J	o	h	n		S	n	o	w	\0	?
A	r	y	a		S	t	a	r	k	\0
T	e	r	i	o	n		L	a	n	\0

4. Arrays of Strings

- การเข้าถึง สายอักขระ(String) และ อักขระ(Char) ใน อาเรย์ของสายอักขระ
  - **การเข้าถึงสายอักขระ** ทำได้โดยระบุ**ชื่ออาเรย์**ของสายอักขระ พร้อม**ดัชนีของแถว**

```
printf("%s", name2[0]);
```

  - **การเข้าถึงอักขระ** ทำได้โดยระบุ**ชื่ออาเรย์**ของสายอักขระ พร้อม**ดัชนีของแถว และ ดัชนีของคอมลัมภ์**

```
printf("%c", name2[1][3]);
```

J	o	h	n		S	n	o	w	\0	?
A	r	y	a		S	t	a	r	k	\0
T	e	r	i	o	n		L	a	n	\0

4. Arrays of Strings

- **ตัวอย่าง 8**

J	o	h	n		S	n	o	w	\0	?	?
A	r	y	a		S	t	a	r	k	\0	?
T	e	r	i	o	n		L	a	n	\0	?
S	a	n	s	a		S	t	a	r	k	\0

```
int i;
char *p;
char name2[4][12]={"John Snow",
                   "Arya Stark",
                   "Terion Lan",
                   "Sansa Stark"};

puts("== House of Stark ==");
for(i=0;i<4;i++)
{
   p=strstr(name2[i],"Stark");
   if(p)
     printf("%s\n",name2[i]);
}
```

4. Arrays of Strings

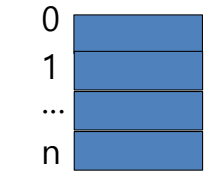
- **ตัวอย่าง 9:** รับชื่อวิชาจากคีย์บอรด์ 5 วิชา แล้วเก็บในอาเรย์ชื่อ subject

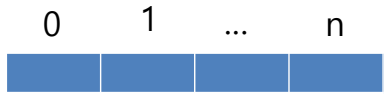
```
int i,j;
char subject[5][12];
for(i=0;i<5;i++) {
  printf("Enter subject:");
  scanf("%s",subject[i]);
}

for(i=0;i<5;i++) {
  for(j=0; j<strlen(subject[i]); j++)
    printf("%c", toupper(subject[i][j]));
  printf("\n");
}
```

```
Enter subject:Math
Enter subject:Physics
Enter subject:Chem
Enter subject:Programming
Enter subject:English
MATH
PHYSICS
CHEM
PROGRAMMING
ENGLISH
```

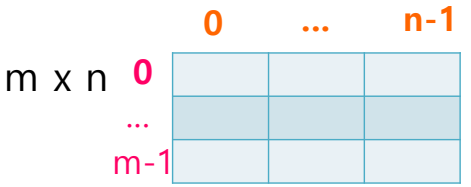

Summary

- อาเรย์ 2 มิติ ตัวแปรชุดที่เก็บข้อมูลชนิดเดียวกันทั้งหมดในลักษณะตาราง





Array 1 **มิติ** มีเพียงแถวหรือคอลัมน์



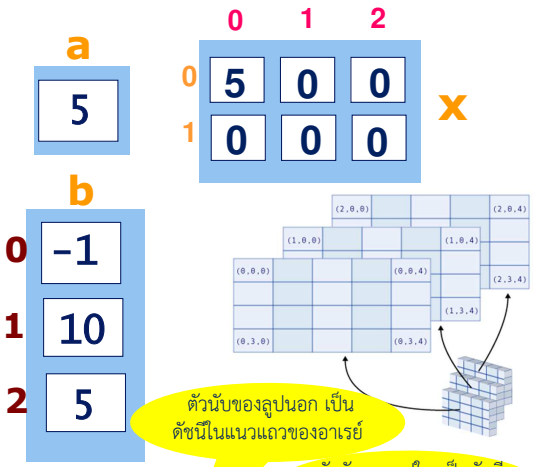
Array 2 **มิติ** มีทั้งแถวและคอลัมน์

การเข้าถึงสมาชิกแต่ละตัวในอาเรย์ 2 มิติ ทำได้โดยการระบุดัชนีของทั้งแถวและคอลัมน์

Summary

```
int a=5;
int b[3]={-1,10,5};
int x[2][3]={5};
```

```
printf("%d", a);
printf(" %d", b[2]);
printf(" %d", x[0][0]);
```




```
int r,c;
for (r=0; r<2; r++)
   for (c=0; c<3; c++)
      printf("%d",x[r][c]);
```

การดำเนินการกับอาเรย์ 2 มิติ ทำได้โดยการใช้คำสั่งลูปซ้อนลูป(Nested Loop)