

Array

by Dr. Sethavidh Gertphol

Outline

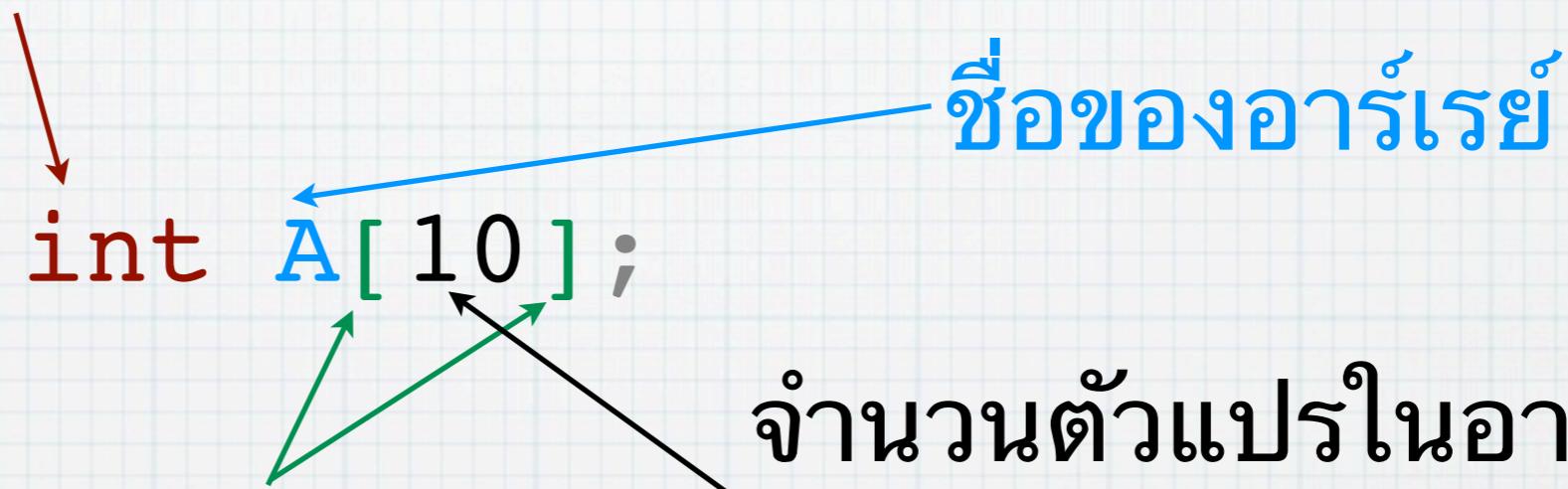
- * เรื่องของอาร์เรย์ (Array)
- * ประการและเรียกใช้งานอาร์เรย์
- * ส่งอาร์เรย์ไปยังฟังก์ชัน
- * ข้อควรระวัง
- * อาร์เรย์สองมิติ

อาร์เรย์คืออะไร

- * กลุ่มของตัวแปรที่ถูกจัดการร่วมกัน
- * จัดการอัตโนมัติโดยคอมไพล์เลอร์
- * จัดการ: สร้าง, เรียกใช้
- * ตัวแปรต้องเป็นประเภทเดียวกัน
- * เช่น `int 10 ตัว` หรือ `char 8 ตัว`
- * จะใช้ชื่อเดียวกัน มีเบอร์ประจำแต่ละตัวแปร

การประกาศอาร์เรย์

ประเภทของอาร์เรย์ ตัวแปรทุกตัว
ในอาร์เรย์จะเป็นประเภทเดียวกัน



เครื่องหมายก้ามปูเปิดและปิด
[] เป็นการบอกว่าเป็นอาร์เรย์

- * ประกาศอาร์เรย์ชื่อ A ซึ่งประกอบไปด้วย
ตัวแปรประเภท int 10 ตัว

การเรียกใช้อาร์เรย์

- * เราต้องการเรียกใช้ตัวแปรในอาร์เรย์
- * ตัวแปรแต่ละตัวจะมีเบอร์กำกับ (**index**) อยู่

```
int A[ 10 ];    A[ 2 ] = 5;
```

- * A[2] เป็นตัวแปรประเภท **int** รองรับมูลค่า
- * ใช้งานเหมือน **int** ปกติ

```
int b;    b = A[ 2 ] + 14;
```

!!!! ระวัง !!!

- * เบอร์ของตัวแปรในอาร์เรย์เริ่มจากศูนย์ (0)
-

int A[10];

A[0] ถึง A[9]

- * ประกาศ เตต A[10] แต่เรียกใช้ A[0] ถึง A[9] เท่านั้น!!!! ไม่มี A[10]

!!! อีกครั้ง !!!

* ประกาศ

```
int A[10];
```

ขนาด (size) ของอาร์เรย์

* เรียกใช้

```
A[0] ถึง A[9]
```

เบอร์ประจำตัวแปร (index)

การตั้งค่าเริ่มต้นของอาร์เรย์

* การตั้งค่าเริ่มต้น (Initialization)

```
int b = 5;
```

* สำหรับอาร์เรย์

```
int A[10] = {0, 1, 2, 3, 4, 5,  
6, 7, 8, 9};
```

```
int A[10] = {0, 1, 2, 3, 4};
```

ที่เหลือเป็น 0

ตัวอย่าง

- * พิมพ์ค่าของตัวแปรทุกตัวในอาร์เรย์
- * ประกาศว่า `int A[10];`
- * ต้องพิมพ์ค่าของ
 - * `A[0], A[1], A[2], ..., A[9]`
- * สังเกตว่า `index` (เบอร์ของตัวแปร) เท่านั้นที่
เปลี่ยน เพิ่มทีละ 1
- * วนลูปพิมพ์ได้

การวนลูปเพื่อจัดการกับตัวแปรทุกตัว ในอาร์เรย์

* ทำได้ดังนี้

```
for (i = 0; i < 10; i++)
```

↑
Isルルン
↑
スルルン
step = i++

* หรือแบบนี้

```
for (i = 0; i <= 10 - 1; i++)
```

* เลือกแบบใดแบบหนึ่ง และจำไว้ใช้เลย

* อย่าใช้ปนกัน เดียวงง

ตัวอย่าง

```
#include <stdio.h>

int main() {
    int A[10] = {0, 1, 2, 3, 4, 5, 6, 7, 8, 9};
    int i;

    for (i = 0; i < 10; i++)
        printf("A[%d] = %d\n", i, A[i]);
}
```

```
113:C akepooh$ ./array_class
A[0] = 0
A[1] = 1
A[2] = 2
A[3] = 3
A[4] = 4
A[5] = 5
A[6] = 6
A[7] = 7
A[8] = 8
A[9] = 9
113:C akepooh$ █
```

กำหนดขนาดอาร์เรย์ตามจำนวนค่าตั้งต้น

- * ทำแบบนี้ก็ได้

```
int A[] = {0, 1, 2, 3, 4};
```

- * สร้างอาร์เรย์ขนาด 5 ตัวแปรและตั้งค่า
- * แต่ต้องระวังเรื่องขนาดของอาร์เรย์เอง

รายละเอียดการใช้งาน

- * ขนาดของอาร์เรย์ตอนประกาศควรเป็น
ค่าคงที่ (constant)
- * ค่าคงที่คือค่าที่ไม่มีการเปลี่ยนแปลง^{ตลอดการทำงานของโปรแกรม}
- * ใช้ `#define directive` ช่วย
- * หรือ `const` คีย์เวิร์ด

#define

- * #define เป็นคำสั่ง (directive) ที่สั่งให้ compiler เปลี่ยนอักษรทั้งหมดในโปรแกรมก่อนคอมไพล์

```
#define arraySize 5
```

- * เปลี่ยนคำว่า arraySize ให้เป็น 5 ให้หมด

```
int A[arraySize];
```

จะกลายเป็น

```
int A[5];
```

ตัวอย่าง

```
#include <stdio.h>
#define arraySize 5

int main() {
    int A[arraySize] = {0,1,2,3,4};
    int i;

    for (i = 0; i < arraySize; i++)
        printf("A[%d] = %d\n", i, A[i]);
}
```

```
113:C akepooh$ ./array_class
A[0] = 0
A[1] = 1
A[2] = 2
A[3] = 3
A[4] = 4
113:C akepooh$
```

```
#include <stdio.h>
#define arraySize 6

int main() {
    int A[arraySize] = {0,1,2,3,4,5};
    int i;

    for (i = 0; i < arraySize; i++)
        printf("A[%d] = %d\n", i, A[i]);
}
```

```
113:C akepooh$ ./array_class
A[0] = 0
A[1] = 1
A[2] = 2
A[3] = 3
A[4] = 4
A[5] = 5
113:C akepooh$
```

const

- * เป็นคีย์เวิร์ดที่บังคับว่าตัวแปรนี้จะไม่มีการเปลี่ยนแปลงตลอดการทำงานของโปรแกรม
- * ต้องตั้งค่าทันทีตอนประกาศ

```
const int arraySize = 5;
```

- * จานี้จะไม่สามารถเปลี่ยนค่าได้
`arraySize++;`

```
array_class.c: In function 'main':  
array_class.c:9: error: increment of read-only variable 'arraySize'
```

ตัวอย่าง

```
#include <stdio.h>

int main() {
    const int arraySize = 5;
    int A[arraySize];
    int i;

    for (i = 0; i < arraySize; i++) {
        A[i] = arraySize - i;
        printf("A[%d] = %d\n", i, A[i]);
    }
}
```

```
113:C akepooh$ ./array_class
A[0] = 5
A[1] = 4
A[2] = 3
A[3] = 2
A[4] = 1
113:C akepooh$
```

```
#include <stdio.h>

int main() {
    const int arraySize = 7;
    int A[arraySize];
    int i;

    for (i = 0; i < arraySize; i++) {
        A[i] = arraySize - i;
        printf("A[%d] = %d\n", i, A[i]);
    }
}
```

```
113:C akepooh$ ./array_class
A[0] = 7
A[1] = 6
A[2] = 5
A[3] = 4
A[4] = 3
A[5] = 2
A[6] = 1
113:C akepooh$
```

ความแตกต่าง

- * `#define` จะเปลี่ยนอักษรทั้งหมดในโปรแกรม เมื่อเป็นการเขียนโปรแกรมใหม่
- * `const int` เป็นการประกาศตัวแปร
- * ชื่อตัวแปรต้องถูกตามข้อกำหนด
- * ตัวแปรส่งผลตาม `scope`
- * ตั้งค่าแต่ต้นไม่ได้ (?)

การใช้ตัวแปรกำหนดขนาดอาร์เรย์

- * ใช้ได้กับคอมไพล์เวอร์ที่ทำงานมาตรฐาน C99
- * คอมไพล์เวอร์บางตัวไม่ยอม

```
int arraySize;
```

```
printf("Enter Array Size: ");  
scanf("%d", &arraySize);
```

```
int A[arraySize];
```

- * !!!ระวัง!!! อย่าเปลี่ยนค่า arraySize เด็ดขาด

ตัวอย่าง

```
#include <stdio.h>

int main() {
    int arraySize, i;
    printf("Enter Array Size: ");
    scanf("%d", &arraySize);

    int A[arraySize];
    for (i = 0; i < arraySize; i++) {
        A[i] = arraySize - i;
        printf("A[%d] is %d\n", i, A[i]);
    }
}
```

```
113:C akepooh$ ./array_class
Enter Array Size: 5
A[0] is 5
A[1] is 4
A[2] is 3
A[3] is 2
A[4] is 1
113:C akepooh$
```

```
113:C akepooh$ ./array_class
Enter Array Size: 7
A[0] is 7
A[1] is 6
A[2] is 5
A[3] is 4
A[4] is 3
A[5] is 2
A[6] is 1
113:C akepooh$
```

ข้อควรระวัง

- * โปรแกรมนี้มี bug
- * แต่ compile ผ่าน!!!
- * แต่อาจมีปัญหาตอนรัน
- * C ไม่มีการเช็คขนาดของอาร์เรย์ให้!!!
- * คนเขียนต้องเช็คเอง
- * ข้อผิดพลาดหลักของโปรแกรมภาษา C

```
#include <stdio.h>

int main() {
    int x = 10;
    int i; |
    int A[100]; 0-99

    A[100] = 10;
    printf("%d\n", A[1000]);
}
```

ลิ้งก์เกิดขึ้น

- * ประกาศ `int A[100]`
- * ควรจะเรียกใช้ได้ตั้งแต่ `A[0]` ถึง `A[99]`
- * แต่ถ้าเรียกใช้ `A[100]` ก็จะไปเรียกใช้ตำแหน่งถัดจาก `A[99]` ไป
- * คอมpileอร์ ของ C ไม่รู้ว่าขนาดของอาร์เรย์ A เป็นเท่าไหร่!!

ชื่อตัวแปรที่จ่อง

ข้อมูล

x	10
i	985710
A[0]	0
A[1]	0
.	0
.	0
A[99]	0
b	10

buffer overflow!

โปรแกรมที่ผิดพลาด

- * จะเกิดอะไรขึ้นเมื่อรัน??
- * ไม่มีครบอกได้

```
#include <stdio.h>

int main() {
    int i;
    int A[10] = {0};

    for (i = 0; i < 100; i++) {
        A[i] = A[i] + 1;
        printf("%d\n", A[i]);
    }
}
```

```
1348422995
1380405075
1094536514
1701605489
1919243360
1634625902
1163133037
2017283411
1836213621
1819239214
1392538224
1280066889
1768042302
1633824623
1409312884
1229213774
1982807379
1714385506
1701080176
1127183219
1329803088
1831939415
Segmentation fault
113:C akepooh$
```

การส่งค่าจากอาร์เรย์ไปยังฟังก์ชัน

- * ส่งค่าตัวแปรตัวหนึ่งในอาร์เรย์
- * ส่งตามปกติ
- * เป็นการ pass by value
- * ส่งค่าไปเพื่อบวก 1

```
113:C akepooh$ ./array_class
0
113:C akepooh$
```

```
#include <stdio.h>

int addOne(int x);

int main() {
    int A[10]={0};

    addOne(A[2]);

    printf("%d\n", A[2]);
}

int addOne(int x) {
    x = x+1;
}
```

ส่งทั้งอาร์เรย์เลยจะ

- * ต้องกำหนดพังก์ชันให้รับตัวแปรประเภทอาร์เรย์ของ int
- * ใช้ int array[] เพื่อบอกว่าเป็นอาร์เรย์

```
int addOne( int array[ ] );
```

- * ตอนส่งให้ส่งชื่ออาร์เรย์ไป addOne(A);
- * สังเกตุว่าการส่งอาร์เรย์ ไม่มีการนับขนาดไปด้วย (คอมไพล์เตอร์ไม่รู้ขนาดของอาร์เรย์)
- * โดยทั่วไปแล้วจะส่งขนาดแยกไปต่างหาก

```
int addOne( int array[ ] , int size );
```

ตัวอย่าง

* บวก 1 ให้ทุกตำแหน่งในอาร์เรย์

* รันปุ๊ป...จะพิมพ์ค่าอะไร

* 0 หรือ 1 ?

```
113:C akepooh$ ./array_class
1
113:C akepooh$
```

0_0 อื๊ะ... ทำไมล่ะ??

pass by reference!!

```
#include <stdio.h>
#define arraySize 10

int addOne(int array[], int size);

int main() {
    int A[arraySize]={0};
    int i;

    addOne(A, arraySize);

    printf("%d\n", A[2]);
}

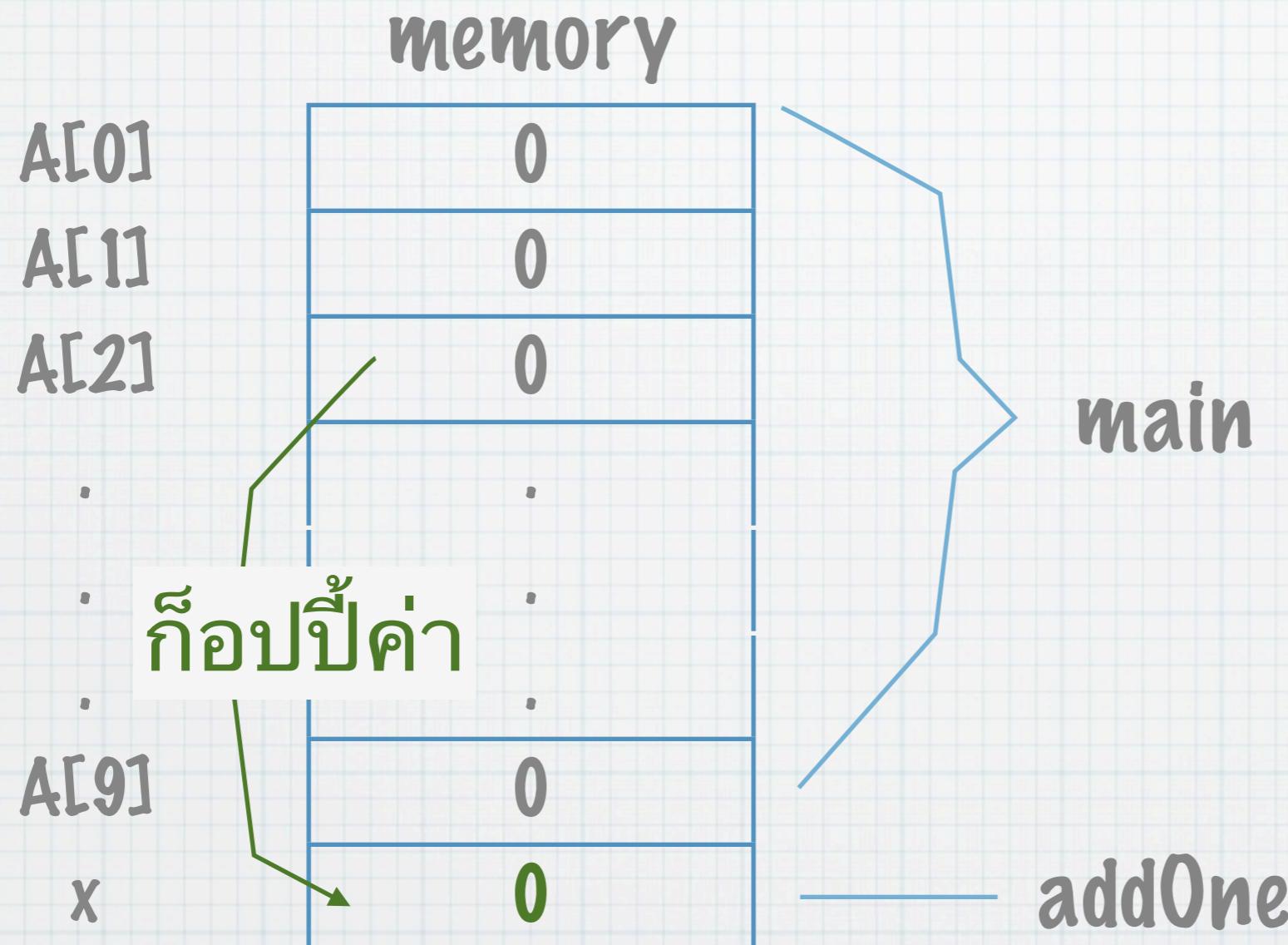
int addOne(int array[], int size) {
    int i;

    for (i = 0; i < size; i++)
        array[i]++;
}
```

pass by value v.s. reference

pass by value	pass by reference
ก็อปปีค่าของตัวแปรจากผู้เรียกไปให้ฟังก์ชัน	ส่งตำแหน่งของตัวแปรจากผู้เรียกไปให้ฟังก์ชัน
สร้างตัวแปร local variable ใหม่ในฟังก์ชัน	ตัวแปรในฟังก์ชันเป็นตัวเดียวกันกับในผู้เรียก
การเปลี่ยนแปลงค่าในฟังก์ชันไม่ส่งผลถึงตัวแปรต้นที่รับ	การเปลี่ยนแปลงค่าในฟังก์ชันเปลี่ยนตัวแปรต้นที่รับด้วย

pass by value



```
#include <stdio.h>

int addOne(int x);

int main() {
    int A[10]={0};
    addOne(A[2]);
    printf("%d\n", A[2]);
}

int addOne(int x) {
    x = x+1;
}
```

pass by reference

ข้อมูล ตำแหน่ง (ในเมม莫รี)

A[0]	0	100
A[1]	0	104
A[2]	0	108
.	.	
.	.	

main

ส่งตำแหน่งของ A[] มา

A[9]	.	136
i	248637	140
array	100	144
size	10	148

addOne

```
#include <stdio.h>
#define arraySize 10

int addOne(int array[], int size);

int main() {
    int A[arraySize]={0};
    int i;

    addOne(A, arraySize);
    printf("%d\n", A[2]);
}

int addOne(int array[], int size) {
    int i;

    for (i = 0; i < size; i++)
        array[i]++;
}
```

ชื่อของอาร์เรย์นั้นสำคัญไฉน

- * ชื่อของอาร์เรย์บอก **ตำแหน่งเริ่มต้น** ของอาร์เรย์นั้น

`int A[10];`

- * **A[0]** คือค่า (**value**) ที่เก็บไว้ในตัวแปรตัวแรก
- * **A[1]** คือค่าที่เก็บไว้ในตัวแปรตัวถัดไป
- * **A** คือ **ตำแหน่งเริ่มต้น** ของอาร์เรย์ (**เชื่งเท่ากับตำแหน่งของ A[0] ด้วย**)

การส่งตัวแหน่ง

	ข้อมูล	ตำแหน่ง
A	100	96
A[0]	0	100
A[1]	0	104
A[2]	0	108
⋮	⋮	⋮
ก้อบปีค่า	⋮	⋮
A[9]	0	136
i	248637	140
array	100	144
size	10	148

- เรียกตัวแปรที่เก็บ
ตำแหน่งว่า **pointer**
- * array[2] อยู่ที่ไหน??
 - * มาดูก่อนว่า array[0] อยู่ที่ไหน
 - * array เก็บตำแหน่งที่ 100 (ส่ง
มาจาก A)
 - * array[0] ก้อยู่ที่ 100
 - * array[2] นับไป 2 ตัว... ก้อยู่ที่ 108
 - * อยู่ที่เดียวกับ A[2]

สรุป

- * `int A[10];`
- * `A` คือชื่อของอาร์เรย์ เก็บตำแหน่งเริ่มต้นของอาร์เรย์ หรือตำแหน่ง `A[0]`

- * ตัวแปรต่อๆไปในอาร์เรย์ ใช้วิธีนับจากจุดเริ่มต้น เช่น `A[8]` ก็นับไป 8 ตัวจากตำแหน่งเริ่มต้น
- * ตัวแปรหนึ่งตัวกินเนื้อที่ตามประเภท
- * เช่น `int` ใช้ 4 bytes, `char` ใช้ 1 byte

สรุป

- * **pass by value:** ส่งค่าไปยังฟังก์ชัน ใช้งานได้เลย
- * **pass by reference:** ส่งตำแหน่งไปยังฟังก์ชัน
 - * ดังนั้น array กับ A ชี้ไปยังตำแหน่งเดียวกัน
 - * A[2] กับ array[2] ก็อยู่ตำแหน่งเดียวกัน
 - * เปลี่ยนค่าที่เก็บไว้ใน array[2] ก็เหมือนเปลี่ยนค่าของ A[2]

2D array

- * อาร์เรย์ปกติเป็น 1 มิติ (เป็นเลี้น)

- * อาร์เรย์สองมิติเป็นตาราง ประกาศแบบ

สองชั้น

int A[2][3];

row column

column

A[0][0]	A[0][1]	A[0][2]
A[1][0]	A[1][1]	A[1][2]

อาร์เรย์ช้อนอาร์เรย์

- * ลังเกตุว่าเราสามารถมองอาร์เรย์สองมิติ เป็นอาร์เรย์ของอาร์เรย์ได้

$\text{int } A[2]; \text{int } B[3]; \text{int } C[3];$

- * ให้ $A[0] = B$ และ $A[1] = C$

column

$A[0]$	$B[0]$	$B[1]$	$B[2]$
$A[1]$	$C[0]$	$C[1]$	$C[2]$

ตั้งค่าเริ่มต้นของอาร์เรย์สองมิติ

* ตั้งค่าแบบอาร์เรย์ช้อนอาร์เรย์

int A[2][3] = { {1,2,3} ,
C → {4,5,6} }

B

1	2	3
4	5	6

int A[2][3] = { {1,2} ,
{3} }

• 1

1	2	0
3	0	0

int A[2][3] = {{1,2,3}}

• 1 2

1	2	3
0	0	0

เบื้องหลังอาร์เรย์สองมิติ

ข้อมูล ตำแหน่ง (ในเมม莫รี)

A	100	96
A[0][0]	1	100
A[0][1]	2	104
A[0][2]	3	108
A[1][0]	4	112
A[1][1]	5	116
A[1][2]	6	120

- * ในเมม莫รีเก็บเป็นอาร์เรย์หนึ่งมิติ
- * ตำแหน่งของตัวแปรอาร์เรย์สองมิติเรียงอย่างไร
- * ภาษา C เป็น **row major** (เรียงตาม **row** ก่อนคือ ยึด **row** เป็นหลัก)

ดังนั้น

- * เราสามารถตั้งค่า **2D array** เหมือน
อาร์เรย์ธรรมดائد้วย

`int A[2][3] = {1,2,3,4,5,6}`

1	2	3
4	5	6

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

1	2	0
0	0	0

- * เราแปลงอาร์เรย์ธรรมด้าให้เป็นอาร์เรย์
สองมิติได้

* ถ้ารู้จำนวน **column!!**

ตัวอย่าง

```
#include <stdio.h>
#define rowSize 2
#define colSize 3

int main() {

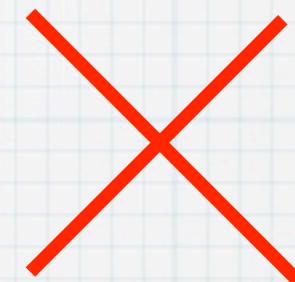
    int A[rowSize][colSize]={{1,2,3}, {4,5,6}};
    int i=0, j=0;

    for (i = 0; i < rowSize; i++) {
        for (j = 0; j < colSize; j++)
            printf("%d ",A[i][j]);
        printf("\n");
    }
}
```

```
113:C akepooh$ ./array_class
1 2 3
4 5 6
113:C akepooh$
```

กำหนดขนาดอาร์เรย์สองมิติตามจำนวนค่าตั้งต้น

```
int A[ ][ ] = { {1,2,3} ,  
                 {4,5,6} }
```



- * ประกาศแบบนี้ไม่ได้ เพราะคอมไพล์ร์
ไม่รู้ว่าจะขึ้น row ใหม่เมื่อไหร่
- * แต่แบบนี้ได้ ครบ 3 column ก็ขึ้น row ใหม่

```
int A[ ][3] = { {1,2,3} ,  
                 {4,5,6} }
```



ใช้ตัวแปรกำหนดขนาด

- * ทำได้เมื่อ compile ด้วยคอมไพล์เลอร์ตามมาตรฐาน C99

```
int rowSize, colSize;
```

```
printf("Enter Row Size: ");
scanf("%d", &rowSize);
printf("Enter Column Size: ");
scanf("%d", &colSize);
```

```
int A[rowSize][colSize];
```

- * !!อย่าเปลี่ยนค่า rowSize / colSize!!

ส่งอาร์เรย์สองมิติไปให้ฟังก์ชัน

- * ผู้เรียกส่งชื่ออาร์เรย์ตามปกติ addOne(A);
- * ฟังก์ชันต้องรู้ว่าที่ส่งมาเป็นอาร์เรย์สองมิติ

```
int addOne(int array[ ][ ], int row, int col);
```

- * ข้างบนไม่เวิร์ค เพราะฟังก์ชันไม่รู้ว่าจะขึ้น row ใหม่เมื่อไหร่
- * ต้องใช้แบบนี้

```
int addOne(int array[ ][ 3 ], int row, int col);
```

ตัวอย่าง

```
#include <stdio.h>
#define rowSize 2
#define colSize 3

int main() {
    int A[rowSize][colSize] = {{1,2,3} , {4,5,6}};
    int i,j;

    addOne(A, rowSize, colSize);

    for (i = 0; i < rowSize; i++) {
        for (j = 0; j < colSize; j++)
            printf("%d ", A[i][j]);
        printf("\n");
    }
}

int addOne(int array[][colSize], int row, int col) {
    int i,j;
    for (i = 0; i < row; i++)
        for (j = 0; j < col; j++)
            array[i][j]++;
}
```

```
113:C akepooh$ ./array_class
2 3 4
5 6 7
113:C akepooh$ █
```

✓

อาร์เรย์มากกว่าสองมิติ

- * สามารถประกาศและเรียกใช้ได้เช่นกัน
- * `int A[2][3][4];`
- * แต่ในเมมโมรีก็ยังเป็น 1 มิติเช่นเดิม
- * นอกนั้นก็คล้ายกับอาร์เรย์สองมิติทุกอย่าง
- * เป็นอาร์เรย์ช้อนอาร์เรย์ช้อนอาร์เรย์ ไปเรื่อยๆ