Data Structures

#2 แถวลำดับ (Array)

เนื้อหา

- ความหมาย
- คุณสมบัติ
- การอ้างอิงตำแหน่งในอาเรย์
- ขอบเขตของอาเรย์
- การจัดเก็บอาเรย์ในหน่วยความจำ

Array :แถวลำดับ

- การรวมกลุ่มกันของตัวแปร ใช้ชื่อเดียวแทนข้อมูลสมาชิก (element) ได้หลายตัว
- ใช้ดรรชนี (Index) อ้างอิงตำแหน่งสมาชิกแต่ละตัว

- Ex.
  - Score[1] คะแนนสอบของนักศึกษาคนแรก
  - Score[6] คะแนนสอบของนักศึกษาคนสุดท้าย

คุณสมบัติอาเรย์

- อาเรย์เป็นตัวแทนของกลุ่มข้อมูลที่มีความสัมพันธ์กัน
  - เดือน 12 เดือน : มกราคม – ธันวาคม
  - รายชื่อนักศึกษาในห้องเรียนจำนวน 6 คน
- สมาชิกในอาเรย์มีคุณสมบัติเหมือนกัน
  - ชนิดข้อมูลเหมือนกัน เช่น อาเรย์เก็บจำนวนเต็ม (Integer) จะไม่สามารถเก็บข้อมูลแบบทศนิยมผสมเข้าไปได้ (Float)
- ขนาดของอาเรย์มีขนาดคงที่
- อ้างอิงข้อมูลได้ทันที (Direct Access) โดยใช้ดรรชนี

การอ้างอิงตำแหน่งในอาเรย์

- เริ่มต้นด้วยชื่ออาเรย์
- ตามด้วยเลขดรรชนี ซึ่งต้องระบุภายในวงเล็บ ( ) ,[ ]
- Ex.
  - Month[1] : แทนเดือนที่ 1 มกราคม
  - Month[2] : แทนเดือนที่ 2 กุมภาพันธ์
  - :
  - Month[12] : แทนเดือนที่ 12 ธันวาคม

การอ้างอิงตำแหน่งในอาเรย์

- สำหรับภาษา c, java จะเริ่มสมาชิกตัวแรก ที่ดรรชนี 0
- ดังนั้น
  - Month[0] : แทนเดือนที่ 1 มกราคม
  - Month[1] : แทนเดือนที่ 2 กุมภาพันธ์
  - :
  - Month[11] : แทนเดือนที่ 12 ธันวาคม

ขอบเขตของอาเรย์ (Bounds)

- ช่วงขอบเขตของค่าประกอบด้วย
  - ขอบเขตล่างสุด (Lower Bound) โดยทั่วจะไปจะไม่กำหนดแต่เข้าใจตรงกันว่าเป็นค่า 0 ในภาษา c/c++, java
  - ขอบเขตบนสุด (Upper Bound)
- อาเรย์ต้องประกาศจำนวนสมาชิกของอาเรย์ก่อนการใช้งาน

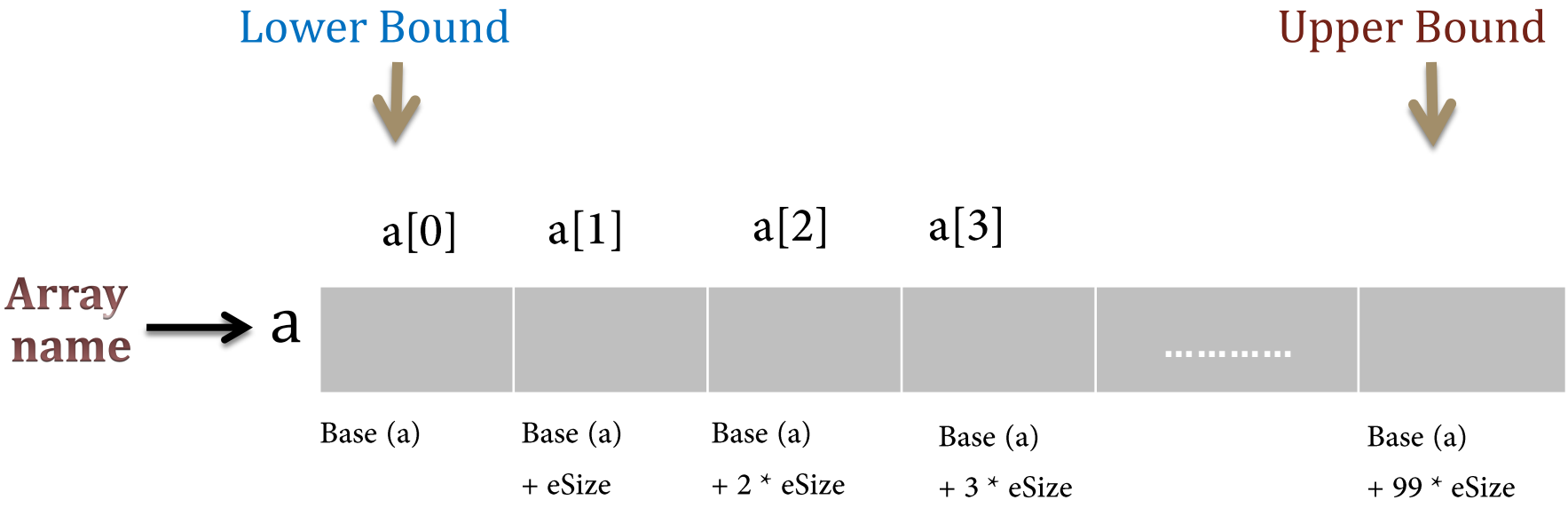
การแทนค่าอาเรย์

int a[100];

- คอมพิวเตอร์จะจัดเตรียมหน่วยความจำที่สามารถเก็บเลขจำนวนเต็ม 100 ค่าติดกัน
- ตำแหน่งแรก เรียกว่า ตำแหน่งฐาน (Base Address)
- สมาชิกของอาเรย์ จะมีขนาดเท่ากันทั้งหมด คือ eSize
- สมาชิกของอาเรย์ ต้องเป็นข้อมูลชนิดเดียวกัน คือ eSize
- อาเรย์ตำแหน่งที่ 1 คือ a[0] : อ้างถึงข้อมูลในตำแหน่ง Base(a)
- อาเรย์ตำแหน่งที่ 2 คือ a[1] : อ้างถึงข้อมูลในตำแหน่ง Base(a) + eSize
- อาเรย์ตำแหน่งที่ i คือ a[i] : อ้างถึงข้อมูลในตำแหน่ง Base(a) + (i) × eSize

ตัวอย่าง

- int a[100]



การคำนวณหาจำนวนสมาชิกอาเรย์

- จำนวนสมาชิก = U – L +1
  - U = ขอบเขตบนสุด (Upper Bound)
  - L = ขอบเขตล่างสุด (Lower Bound)

- int a[4]
  - จำนวนสมาชิก = 4 - 0 +1

    = 5

การจัดเก็บในหน่วยความจำ

- ข้อมูลต่อเนื่องกันไป
- ใช้เนื้อที่ขนาดเท่ากัน ในการจัดเก็บสมาชิกของอาเรย์
- ต้องเป็นข้อมูลชนิดเดียวกัน
- แบ่งประเภทได้ดังนี้
  - อาเรย์ 1 มิติ
  - อาเรย์หลายมิติ

ประเภทของอาเรย์

- อาเรย์ 1 มิติ
- อาเรย์หลายมิติ
  - 2 มิติ
  - 3 มิติ

การประกาศตัวแปร อาเรย์ 1 มิติ

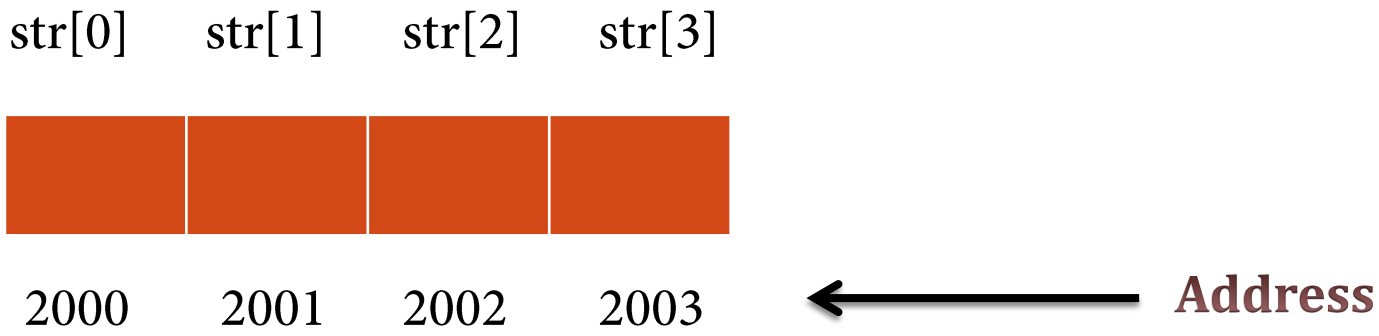
- รูปแบบ

  data-type array-name[จำนวนสมาชิก]

- data-type ประเภทข้อมูล
- array-name ชื่ออาเรย์

ตัวอย่าง อาเรย์ 1 มิติ

- char *str*[4]
  - คอมพิวเตอร์จองเนื้อที่ในหน่วยความจำ สำหรับตัวแปร ***str***
  - ให้เป็นตัวแปรชนิด character (ขนาดในหน่วยความจำ 1 byte)
  - ขนาดสมาชิก 4 สมาชิก



การกำหนดค่าให้กับตัวแปรอาเรย์ 1 มิติ

- การกำหนดค่า

```
char a[4];
a[0] = 'v';
a[1] = 'w';
:
a[3] = 'y';
```

- หรือ

```
char a[4] = {'v', 'w', 'x', 'y'};
```

การใช้งานตัวแปรอาเรย์ 1 มิติ

- การใช้งาน
  - อ้างถึงสมาชิกที่ต้องการโดยใช้ดรรชนี (index)

- ตัวอย่าง

  char a[4] = {‘v’, ‘w’, ‘x’, ‘y’};

  char temp := a[1];

  ดังนั้น temp จะมีค่าเท่ากับ ‘w’

ตัวอย่าง

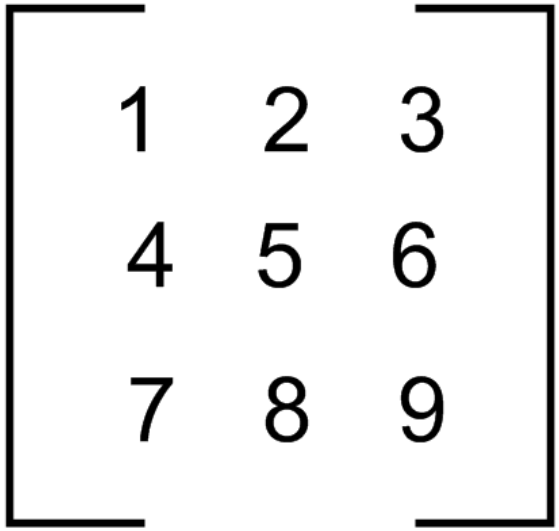
```
int score[5] = {2, 8, 16, 24, 9};

int temp = score[3] ;
printf("%d \n", temp); //แสดงค่า 24


// แสดงค่าทุกค่าใน อาเรย์
int i;
for(i=0; i<5; i++){
    printf("score[%d] = %d \n", i, score[i]);
}
```

อาเรย์สองมิติ (Two Dimension Array)

- รูปแบบการเก็บแบบ Matrix
- โครงสร้างประกอบด้วย
  - แถว (Row)
  - คอลัมน์ (Column)
- นิยมใช้ เนื่องจากการเก็บข้อมูลทั่วไปอยู่ในรูปแบบตารางสองมิติ



3 * 3

อาเรย์สองมิติ (Two Dimension Array)

- รูปแบบ

  data-type array-name[n, m]

- data-type ประเภทข้อมูล
- array-name ชื่ออาเรย์
- n ตัวเลขที่แสดงตำแหน่งของแถว (Row)
- m ตัวเลขที่แสดงตำแหน่งของคอลัมน์ (Column)

ตัวอย่าง อาเรย์สองมิติ

- int a[3][4]

	Column 0	Column 1	Column 2	Column 3
Row 0	a[0][0]	a[0][1]	a[0][2]	a[0][3]
Row 1	a[1][0]	a[1][1]	a[1][2]	a[1][3]
Row 2	a[2][0]	a[2][1]	a[2][2]	a[2][3]

- อาเรย์ a สามารถเก็บข้อมูลได้ ดังนี้

7	8	1	-3
2	-1	5	10
0	6	3	9

การกำหนดค่าตัวแปรอาเรย์สองมิติ

- การกำหนดค่า

```
int a[2][3] = {{2, 4, 6} , {7, 5 ,1}}
```

- หรือ

```
int a[2][3];
a[0][0] = 2;
a[0][1] = 4;
a[0][2] = 6;
a[1][0] = 7;
a[1][1] = 5;
a[1][2] = 1;
```

การใช้งานอาเรย์สองมิติ

```
int a[2][3] = {{2, 4, 6} , {7, 5 ,1}}

temp = a[0][2];

printf("%d \n", temp);  //แสดงค่า 6


temp2 = a[1][2];

printf("%d \n", temp);  //แสดงค่า 1
```

การใช้งานอาเรย์สองมิติ (2)

```
int a[2][3] = {{2, 4, 6} , {7, 5 ,1}}


int i, j;
for (i=0; i < 2; i++){
    for (j=0; j < 3; j++){
            printf("a[%d][%d] = %d", i, j, a[i][j]);
    }
}
```

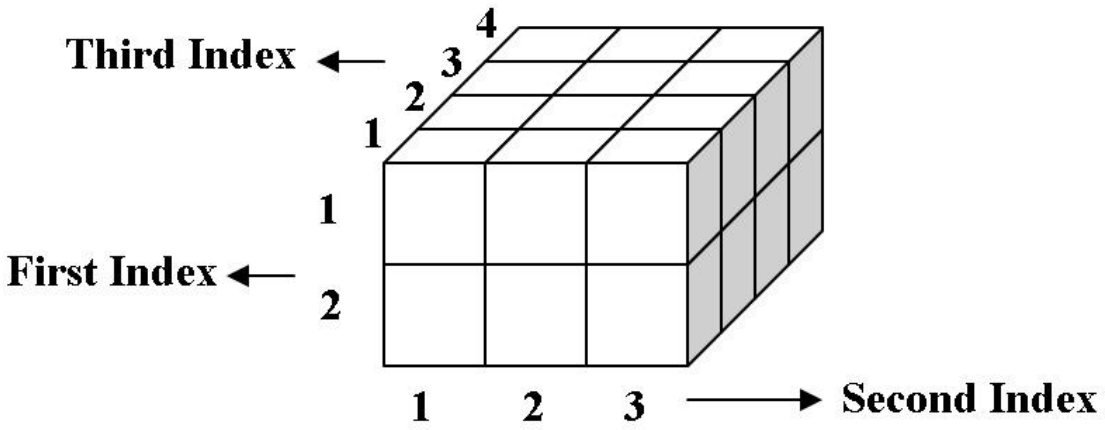
การใช้งานในชีวิตจริง

- ตารางสรุปยอดขายของแต่ละไตรมาส
  - int income[3][4]

ปี พ.ศ.	ไตรมาศที่ 1	ไตรมาศที่ 2	ไตรมาศที่ 3	ไตรมาศที่ 4
2553	3,000,564	2,550,000	2,853,271	2,799,853
2555	4,255,844	-500,875	-1,100,181	1,500,800
2556	1,913,086	2,230,510	4,500,990	3,000,655

อาเรย์สามมิติ (Three Dimension Array)

- เป็นการนำอาเรย์สองมิติมาซ้อนกัน หลาย ๆ ชั้น (Page)



Three-dimensional array with twenty four elements

ตัวอย่างอาเรย์ 3 มิติ

a[2][0][0]	a[2][0][1]	a[2][0][2]
a[2][1][0]	a[2][1][1]	a[2][1][2]

Page 3

a[1][0][0]	a[1][0][1]	a[1][0][2]
a[1][1][0]	a[1][1][1]	a[1][1][2]

Page 2

a[0][0][0]	a[0][0][1]	a[0][0][2]
a[0][1][0]	a[0][1][1]	a[0][1][2]

Page 1

รูปแบบการประกาศตัวแปรอาเรย์ 3 มิติ

- รูปแบบ

  data-type array-name [l, n, m]

- data-type ประเภทข้อมูล
- array-name ชื่ออาเรย์
- l ตัวเลขที่แสดงชั้น (Page)
- n ตัวเลขที่แสดงตำแหน่งของแถว (Row)
- m ตัวเลขที่แสดงตำแหน่งของคอลัมน์ (Column)

การใช้งานในชีวิตจริง

- แต่ละpage แทนห้องแต่ละห้อง
- แต่ละแถวแทนข้อมูลของ นศ. 1 คน
- แต่ละคอลัมน์แทนด้วยคะแนน

30	11	12
37	29	28

Room 3

25	22	20
20	25	30

Room 2

30	40	20
25	35	15

Room 1

ข้อดีของ Array

- จัดการง่าย
- เข้าถึงข้อมูลได้รวดเร็ว

ข้อจำกัดของ Array

- ต้องกำหนดขนาดของอาเรย์ที่แน่นอน
- **ประเด็น** ขนาดของอาเรย์ที่เหมาะสม ควรเป็นเท่าไหร่ ?
  - สมมติ ต้องการสร้างอาเรย์เพื่อรับข้อมูลนักเรียน n คน
  - n ควรจะเป็นสองเท่าของจำนวนนักเรียนจริง
  - สมมุติ(อีกครั้ง) ในตอนนักเรียนเข้าลงทะเบียนจริง มีจำนวน 2n+1 จะเป็นอย่างไร ?

End of Array