บทที่ 2
โครงสร้างข้อมูลแบบ Array และ String

สุนทรี คุ้มไพโรจน์

Array(แถวลำดับ)

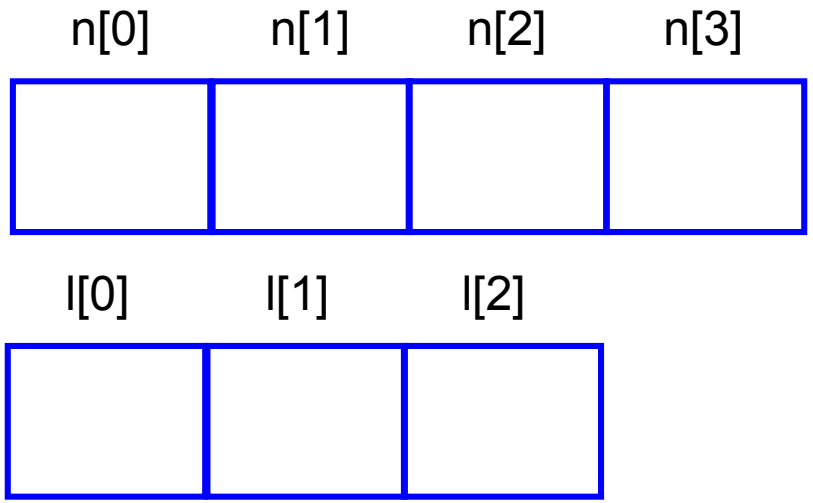
- โครงสร้างข้อมูลที่เก็บข้อมูลได้เป็นชุด
- โดยจัดเก็บใน Memory (หน่วยความจำ) แบบต่อเนื่อง
- ข้อมูลอยู่ในตำแหน่งที่ต่อเนื่องกันไปตามลำดับ แต่ละช่องข้อมูลจะเก็บ**ข้อมูลชนิดเดียวกัน**
- การ Access (เข้าถึงข้อมูล) ใด ๆ ใน array อ้างถึงได้โดยใช้ Index (ดัชนี) เป็นการระบุหมายเลขกำกับช่องข้อมูล
- Length(ความยาว) หรือ size(ขนาด)ของ array คือจำนวนสมาชิกใน array

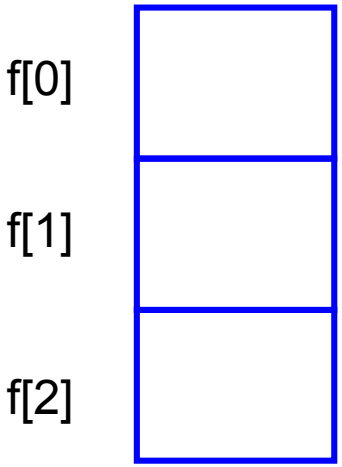
แถวลำดับ (Array)

- แถวลำดับ 1 มิติ (one dimensional array)
- แถวลำดับ 2 มิติ (two dimensional array)
- แถวลำดับหลายมิติ (multidimensional array)

อาร์เรย์หนึ่งมิติ (One Dimension Array)

- จัดเก็บข้อมูลในลักษณะต่อเนื่องกันเป็นแถว
- รูปแบบการจองพื้นที่ในภาษา C : type array_name[size]
- เช่น int n[4];
  float f[3];
  char l[3];





L17.c

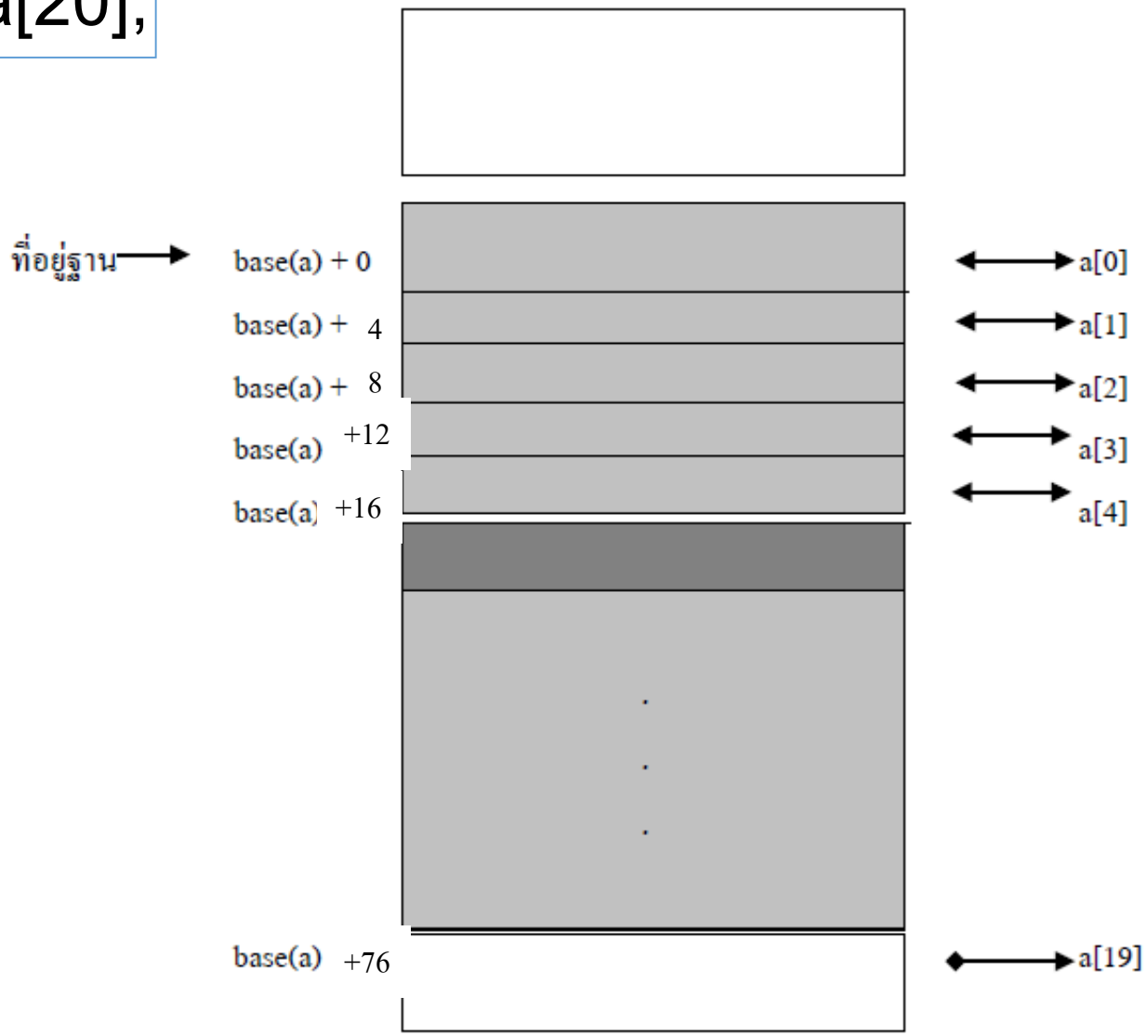
```c
#include <stdio.h>
   int main() {
        int   n[4] = {9,-7,10,-2};
        float f[4] = {-0.5,1,0.044,-3.1111};
        char  l[4] = {'t','e','s','t'};
        for(int i=0; i<4;i++)
            printf("%d  %f  %c\n",n[i],f[i],l[i]);
   return 0;
}
```

ให้สังเกต format การพิมพ์

(address) (memory) (array member)

ที่อยู่ หน่วยความจำ สมาชิกของแถวลำดับ

```
int  a[20];
```



สตริง(String)

- array of character หรือชุดของตัวอักขระหลายๆตัวที่มาต่อกัน สิ่งที่ได้อาจเป็นคำ หรือประโยค แล้วจะปิดท้ายคำหรือประโยคด้วย '\0'**หรือ** NULL เสมอ

- การประกาศตัวแปรต้องจองเนื้อที่สำหรับ '\0' **ด้วย**

  เช่น char a[6]; //**ถ้ารับคำว่า** "School" **จะตัดตัวอักษรสุดท้าย มีความยาว** =5

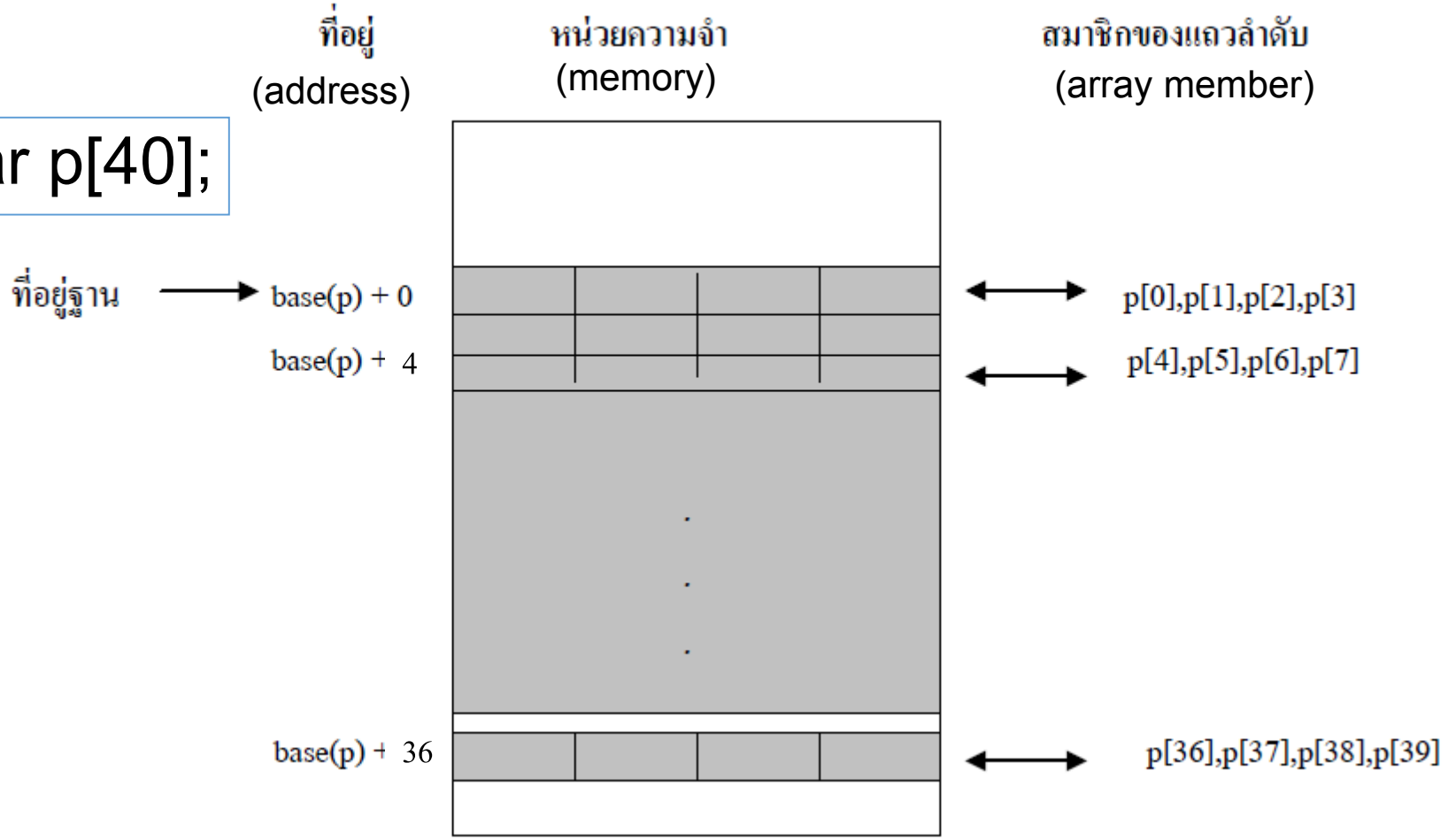
a	'S'	'c'	'h'	'o'	'o'	'\0'

char c[ ] = "cs 231"; //**ตัวแปรนี้มีความยาว** = 6 ซึ่ง compiler ของภาษาซี จะนำ string ใส่ใน arrayให้อัตโนมัติ จะได้ดังนี้

c	'c'	's'	' '	'2'	'3'	'1'	'\0'

สายอักขระ (String)

char p[40];



ตัวอย่างการกำหนดข้อมูลตัวแปร(L16.c)

```c
#include <stdio.h>
//ให้สังเกตสิ่งที่เกิดขึ้นเมื่อพิมพ์ sentence3
int main() {
    char message[15]="Hello student!";
    char course[9]  ={'0','1','4','1','8','2','3','1','\0'};
    char vocab[3]   ={'a','n','t'};
    printf("sentence 1: %s\n",message);
    printf("sentence 2: %s\n",course);
    printf("sentence 3: %s\n",vocab);
    return 0;
}
```

ไม่มี \0 ปัญหาให้เอา
อย่างอื่นมาต่อ ในหน่วย
ความจำมาต่อจนเจอ \0

การรับค่าและแสดงผลของ string

การรับค่าของตัวแปร string จาก standard input

- ฟังก์ชัน scanf() **ห้ามมี space bar ระหว่างข้อมูลที่ป้อน

  scanf("%s",str);

  str คือ ตัวแปร string (array of character)

- ฟังก์ชัน gets() **จะหยุดรับเมื่อกดenter (มี space bar ได้)

  gets(char *s);

  s คือ ตัวแปร string

การรับค่าและแสดงผลของ string (ต่อ)

การแสดงผลค่าของตัวแปร string ออกที่standard output

- ฟังก์ชัน printf()

  printf(“%s”,str);

  str คือ ตัวแปร string ที่ต้องการแสดงผล

- ฟังก์ชัน puts()

  int puts(char *str);

  str คือ ตัวแปร string ที่ต้องการแสดงผล

ตัวอย่าง array of character (L19.c)

```c
#include <stdio.h>
int main() {
    char sentence[100];
    int count=1, c=0;
    printf ("Enter Sentence:");
    gets(sentence);
    while (sentence[c]!='\0'){
      if (sentence[c]==' ')
          count++;
      c++;
    }
    printf("number of words= %d\n",count);
    return 0;
}
```

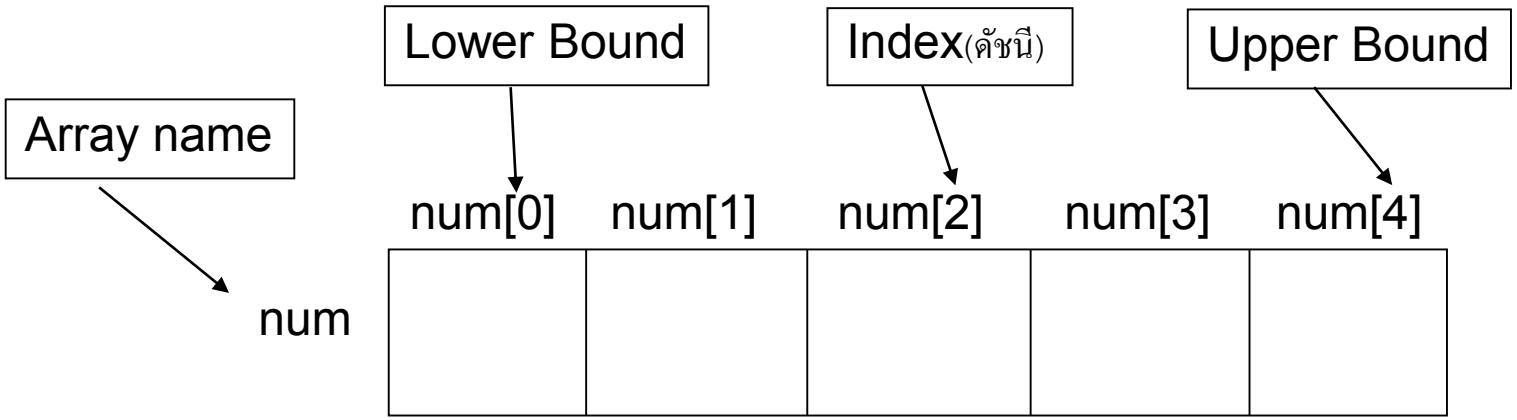
การอ้างถึงข้อมูลตัวแปร L18.c

```c
#include <stdio.h>
int main() {
    int   score[10];
    int   sum = 0;
    for(int i=0; i<10;i++) {
        printf("Enter score[%d]:",i);
        scanf("%d",&score[i]);
        sum += score[i];
    }
    printf("Average score =%d  \n",sum/10);
    return 0;
}
```

ขอบเขตของอาร์เรย์ 1 มิติ

- เลขดัชนีในอาร์เรย์ประกอบด้วยช่วงขอบเขตของค่าซึ่งประกอบด้วย ขอบล่างสุด (Lower Bound) และขอบเขตบนสุด (Upper Bound)

```
int    num[5];
```



L คือ ขอบเขตล่างสุด (Lower Bound)

U คือ ขอบเขตบนสุด (Upper Bound)

จำนวนสมาชิก = U-L+1

เช่น num มีสมาชิก เท่ากับ 4-0+1 = 5 ตัวเท่ากับ size (ขนาดของ array)

การคำนวณหาตำแหน่ง(Address)ในหน่วยความจำของอาร์เรย์หนึ่งมิติ

- คำนวณได้จากสูตร

$$Loc\ (A[i]) = Base + w * (i - L)$$

โดยที่

i : ตำแหน่งที่ต้องการ

Base : ตำแหน่งเริ่มต้นในหน่วยความจำ

L : ขอบเขตล่างสุด

w : ขนาดความกว้างของชนิดข้อมูล เนื้อที่หน่วยความจำที่ใช้

ตัวอย่าง การหาตำแหน่งในหน่วยความจำ ของอาร์เรย์หนึ่งมิติ

int Data[5];

data เก็บข้อมูลชนิดตัวเลข สมมติว่าใช้เนื้อที่ในการเก็บข้อมูล 2 bytes ต่อตัว โดยมีตำแหน่งเริ่มต้นในหน่วยความจำอยู่ที่ 1000

จงหาตำแหน่งที่ใช้เก็บข้อมูลของ Data[1]

แทนค่าดังนี้

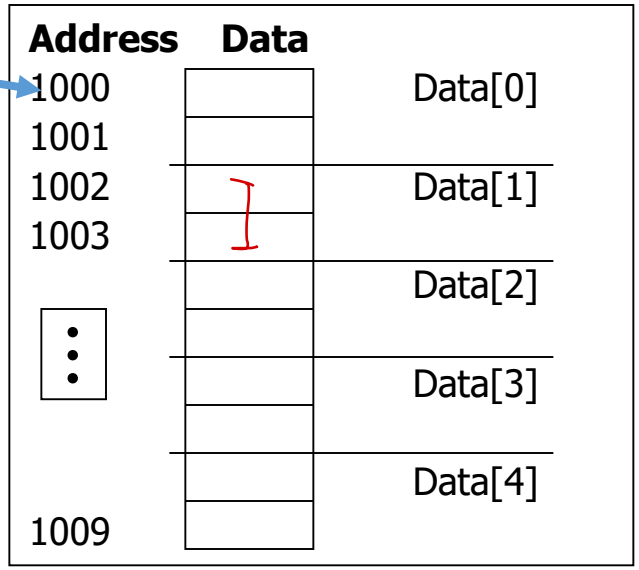
i = 2, B = 1000, L = 0, w = 2

จะได้ Loc(Data[1]) = 1000 + 2 * (1 – 0)

= 1000 + 2 * 1

= 1000 + 2

= 1002



- int D[10];
- data เก็บข้อมูลชนิดตัวเลข ภาษา C
- สมมติว่าใช้เนื้อที่ในการเก็บข้อมูล 4 bytes ต่อตัว โดยมีตำแหน่งเริ่มต้นในหน่วยความจำอยู่ที่ 1000
- จงหาตำแหน่งที่ใช้เก็บข้อมูลของ Data[6]

ตัวอย่าง การหาตำแหน่งในหน่วยความจำ ของอาร์เรย์หนึ่งมิติ

ต้องการประกาศตัวแปรอาร์เรย์ income เพื่อจัดเก็บยอดรายได้ของ ปี พ.ศ. 2541-2550

ขนาดของ array จะคำนวณได้จาก 2550 – 2541 +1 = 10

ภาษา C เป็นไปตามรูปแบบ int income[10];

สมมติว่า ข้อมูลที่จัดเก็บกินเนื้อที่ 4 ไบต์ต่อสมาชิก1ตัว

สมมติว่า สมาชิกตัวแรกจัดเก็บในตำแหน่งที่ 7000

จงคำนวณหาดัชนีที่จัดเก็บยอดรายได้ของปี 2549 = 2549-2541+1 = 9

จงคำนวณหาแอดเดรสที่จัดเก็บยอดรายได้ของปี 2549

แทนค่า กรณีที่ lower bound **ไม่ใช่ 0 ดังนี้**

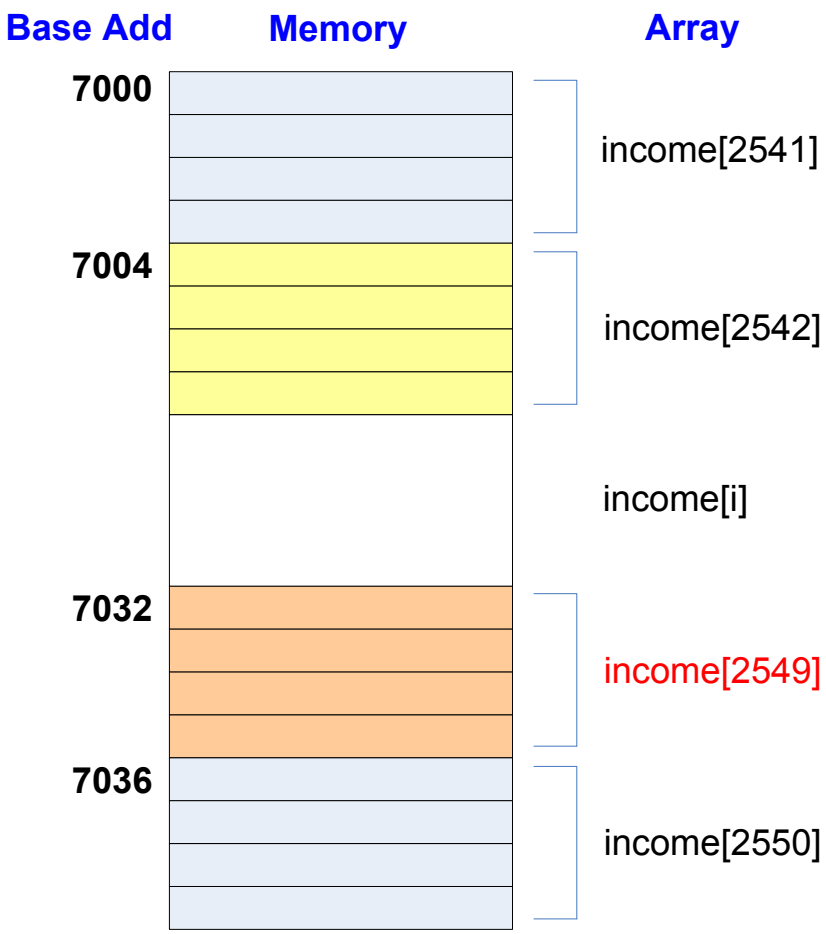
i = 2549, B= 7000, L = 2541, w = 4

จะได้ Loc(income[2549]) = 7000 + 4 * (2549 – 2541)

= 7000 + 4 * 8

= 7000 + 32

= 7032


Base Add
Memory
Array
7000
7004
7032
7036
income[2541]
income[2542]
income[i]
income[2549]
income[2550]

อาร์เรย์สองมิติ (Two Dimension Array)

- โครงสร้างข้อมูลที่มีการจัดเก็บข้อมูลแบบตารางสองทาง ข้อมูลมีการจัดเรียงกันตามแนวแถว (Row) และ แนวหลัก (Column) การอ้างถึงข้อมูลต้องระบุตำแหน่งแถวและตำแหน่งหลักที่ข้อมูลนั้นอยู่

	Col 1	Col 2	Col 3	Col 4
Row 1	Data	Data	Data	Data
Row 2	Data	Data	Data	Data
Row 3	Data	Data	Data	Data
Row 4	Data	Data	Data	Data

รูปแบบทั่วไปของโครงสร้างข้อมูลอาร์เรย์ 2 มิติ

ArrayName [L_1 : U_1 , L_2 : U_2]

เมื่อ ArrayName คือ ชื่อของโครงสร้างข้อมูลอาร์เรย์

L_1 คือ ค่าขอบเขตล่างสุด (Lower Bound) ของแถว

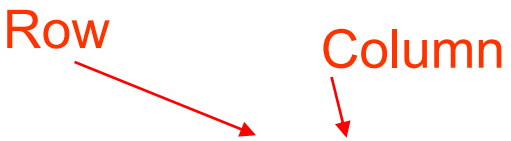
U_1 คือ ค่าขอบเขตสูงสุด (Upper Bound) ของแถว

L_2 คือ ค่าขอบเขตล่างสุด (Lower Bound) ของคอลัมน์

U_2 คือ ค่าขอบเขตสูงสุด (Upper Bound) ของคอลัมน์

อาร์เรย์สองมิติ

- เช่น int K[4,3];



- ในบางภาษาสามารถระบุขอบเขต array ได้ เช่น K[0:3,0:2]

Columns

Rows	0	1	2
0	10	5	3
1	2	1	9
2	11	6	7
3	0	4	3

การคำนวณหาจำนวนสมาชิกของอาร์เรย์สองมิติ

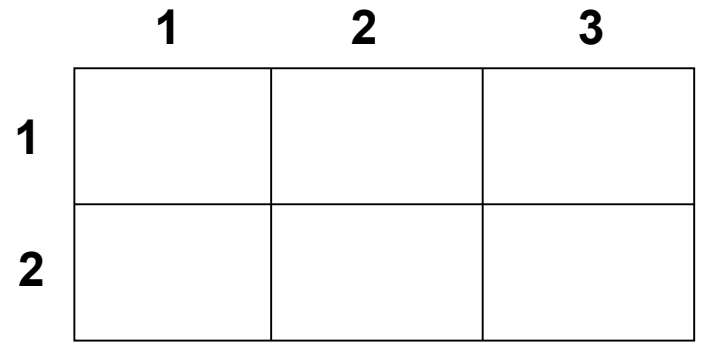
- จำนวนสมาชิก = $(U_1 - L_1 + 1) * (U_2 - L_2 + 1)$

  โดย U_1 = ขอบเขตบนสุด ของแถว

  L_1 = ขอบเขตล่างสุด ของแถว

  U_2 = ขอบเขตบนสุด ของคอลัมน์

  L_2 = ขอบเขตล่างสุด ของคอลัมน์



เช่น A[1:2,1:3] มีจำนวนสมาชิก = (2-1+1)*(3-1+1)

= 2*3

= 6

การประกาศอาร์เรย์ 2 มิติในภาษาคอมพิวเตอร์

- ภาษา c , C++ ประกาศโดยใช้รูปแบบดังนี้

  รูปแบบ data_type array_name[row_size][column_size]

  เช่น

  ```
  int     num[2][3];
  char  name[5][20];
  ```

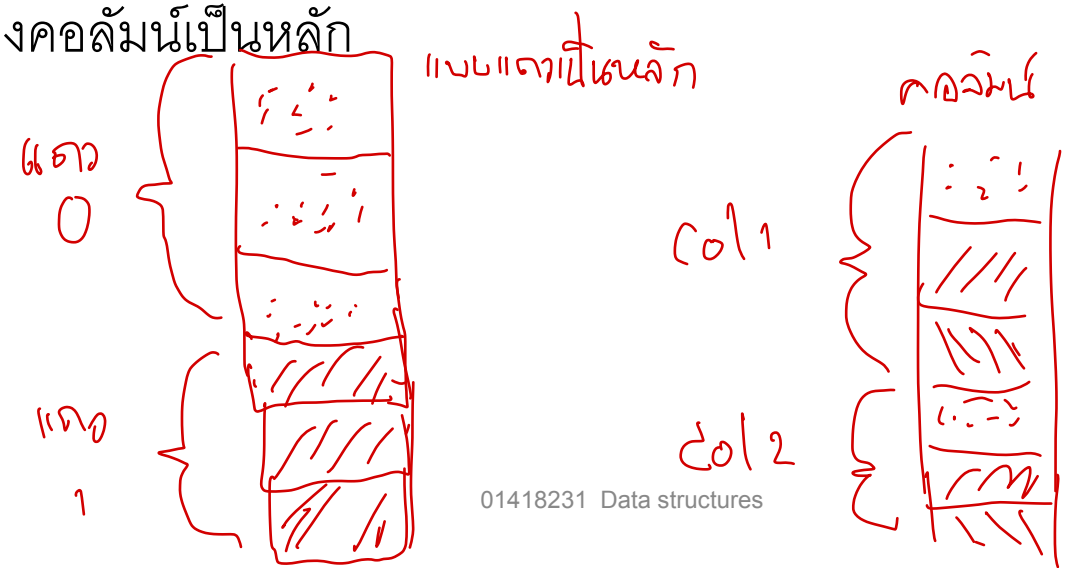
การจัดเก็บอาร์เรย์สองมิติในหน่วยความจำ

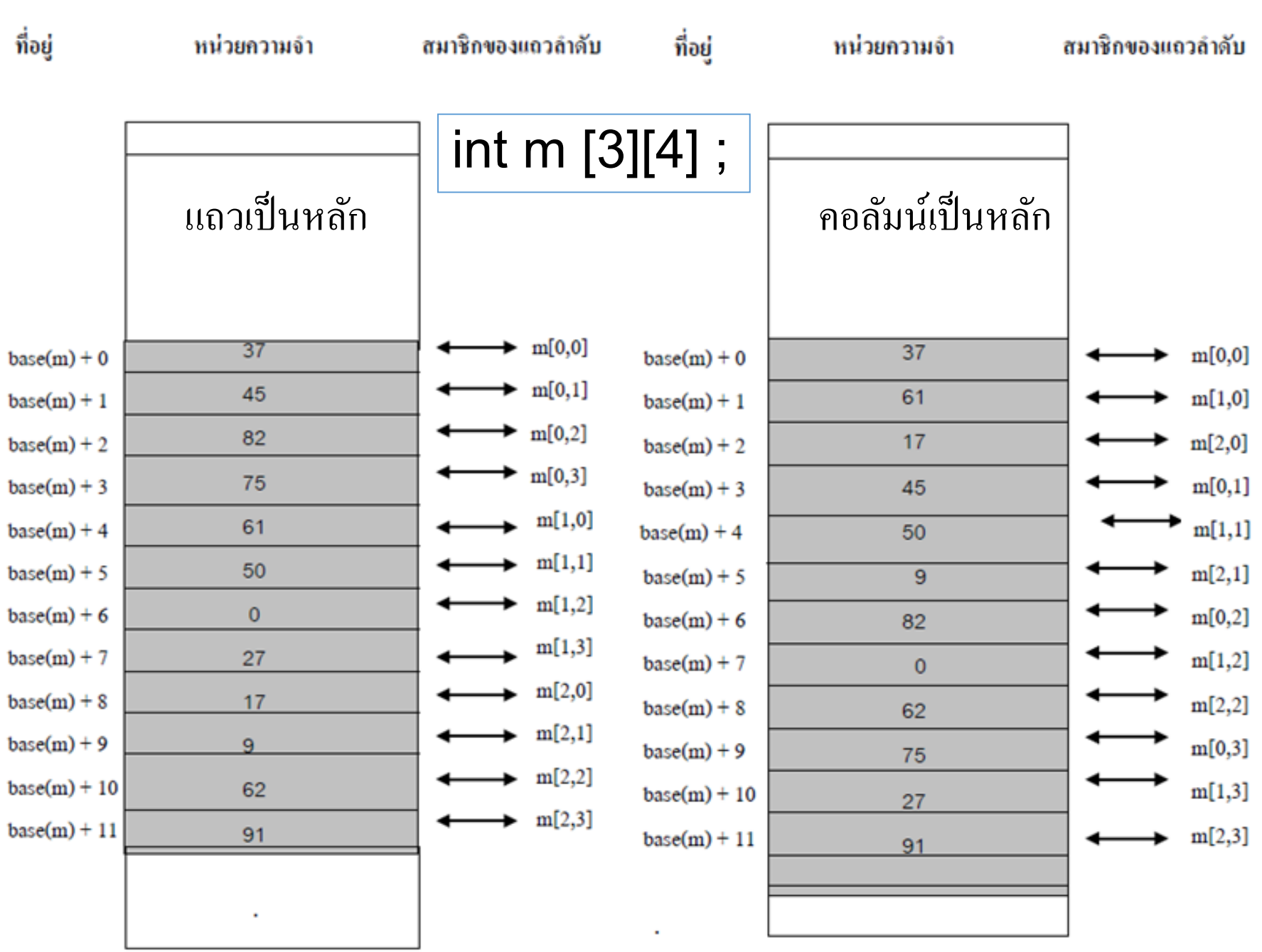
ทำได้ 2 แบบ

1. การจัดเก็บด้วยการเรียงแถวเป็นหลัก (Row Major Order)
2. การจัดเก็บด้วยการเรียงคอลัมน์เป็นหลัก (Column Major Order)

   การจัดเก็บแบบที่ 2 นี้ ใช้กับภาษาฟอร์ทราน

   เนื่องจากภาษาฟอร์ทรานเป็นภาษาที่ใช้กับคอมพิวเตอร์บางรุ่นในยุคต้นๆ เครื่องเหล่านั้นมีโครงสร้างแอดเดรสที่เหมาะกับการจัดเก็บข้อมูลแบบเรียงคอลัมน์เป็นหลัก




ที่อยู่
หน่วยความจำ
สมาชิกของแถวลำดับ
ที่อยู่
หน่วยความจำ
สมาชิกของแถวลำดับ
int m [3][4] ;
แถวเป็นหลัก
คอลัมน์เป็นหลัก
base(m) + 0
base(m) + 1
base(m) + 2
base(m) + 3
base(m) + 4
base(m) + 5
base(m) + 6
base(m) + 7
base(m) + 8
base(m) + 9
base(m) + 10
base(m) + 11
37
45
82
75
61
50
0
27
17
9
62
91
m[0,0]
m[0,1]
m[0,2]
m[0,3]
m[1,0]
m[1,1]
m[1,2]
m[1,3]
m[2,0]
m[2,1]
m[2,2]
m[2,3]
base(m) + 0
base(m) + 1
base(m) + 2
base(m) + 3
base(m) + 4
base(m) + 5
base(m) + 6
base(m) + 7
base(m) + 8
base(m) + 9
base(m) + 10
base(m) + 11
37
61
17
45
50
9
82
0
62
75
27
91
m[0,0]
m[1,0]
m[2,0]
m[0,1]
m[1,1]
m[2,1]
m[0,2]
m[1,2]
m[2,2]
m[0,3]
m[1,3]
m[2,3]

สูตรการคำนวณหาตำแหน่งที่ใช้เก็บข้อมูลในอาร์เรย์สองมิติ

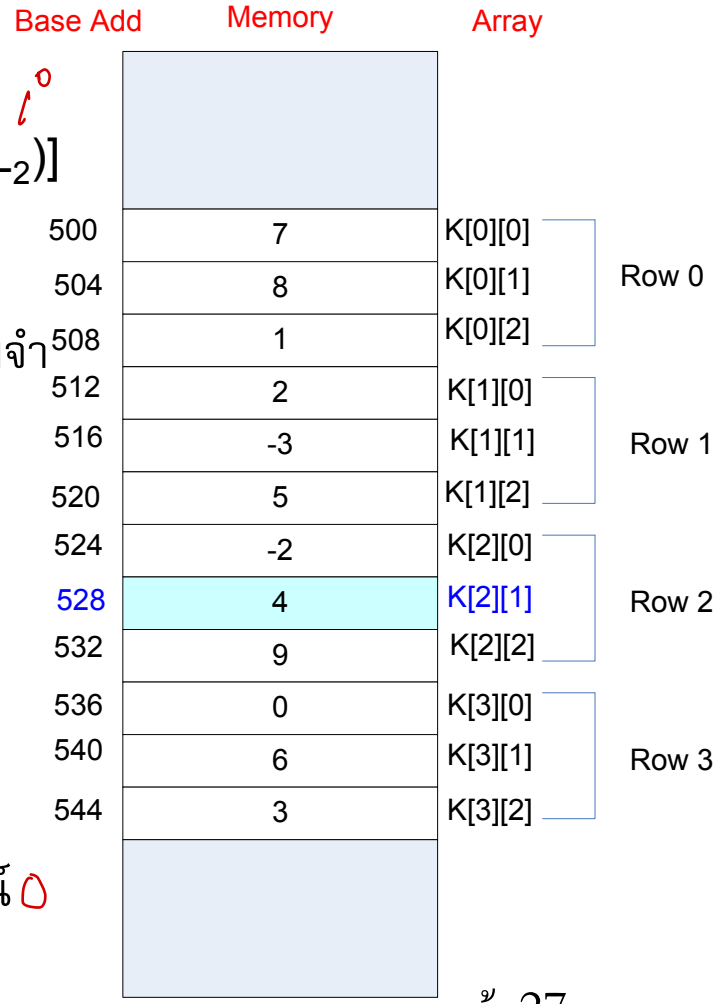
1. แบบการเรียงแถวเป็นหลัก

สูตร

$$LOC(K[i,j]) = B + w[C * (i-L_1) + (j-L_2)]$$

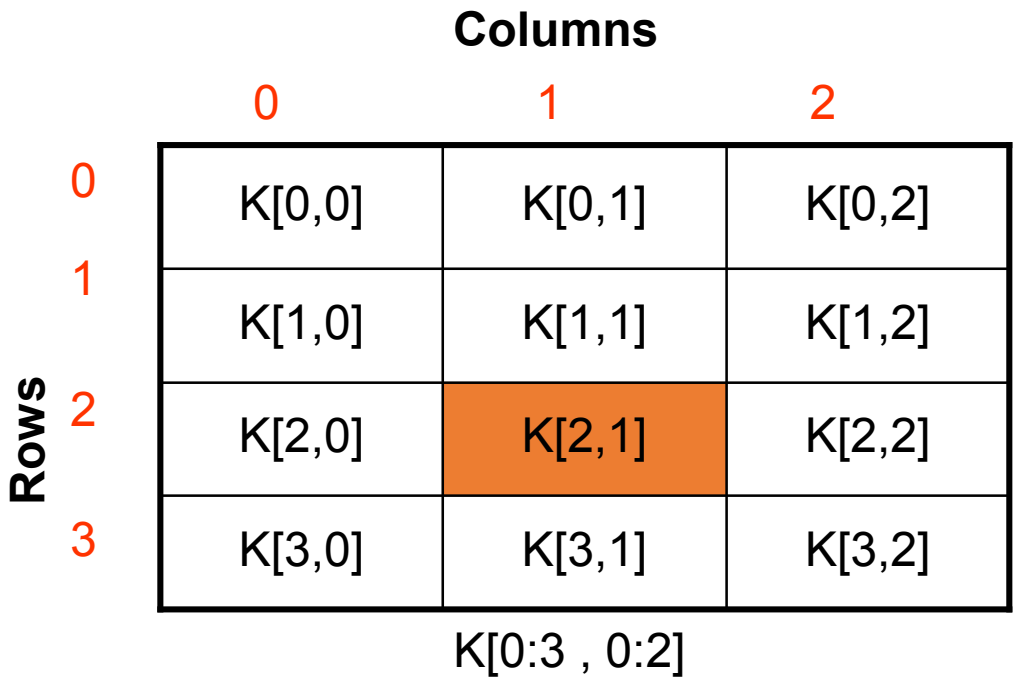
โดยที่

LOC(K[i,j])	= ตำแหน่งแอดเดรสที่เก็บ K[i,j] ในหน่วยความจำ
B	= แอดเดรสเริ่มต้น (Base Address)
w	= จำนวนช่องของหน่วยความจำที่จัดเก็บข้อมูลต่อหนึ่งสมาชิก
i	= ตำแหน่งของแถวในอาร์เรย์
j	= ตำแหน่งของคอลัมน์ในอาร์เรย์
L_1	= ค่าขอบเขตล่างสุด (lower Bound) ของแถว
L_2	= ค่าขอบเขตล่างสุด (lower Bound) ของคอลัมน์
C	= จำนวนคอลัมน์ของแถวลำดับ



ตัวอย่าง

ต้องการทราบตำแหน่งแอดเดรสที่เก็บข้อมูลอาร์เรย์ K มีขนาด 4x3 แถวที่ 2 คอลัมน์ 1 (K[2,1]) กำหนดให้แอดเดรสเริ่มต้นที่ 500 ข้อมูลที่จัดเก็บกินเนื้อที่ 4 ไบต์ต่อสมาชิก1ตัว



การคำนวณ

สูตร

LOC(K[i, j]) = B + w [C * (i-L1) + (j-L2)]

LOC(K[2, 1]) = 500+ 4 [3(2-0) + (1-0)]

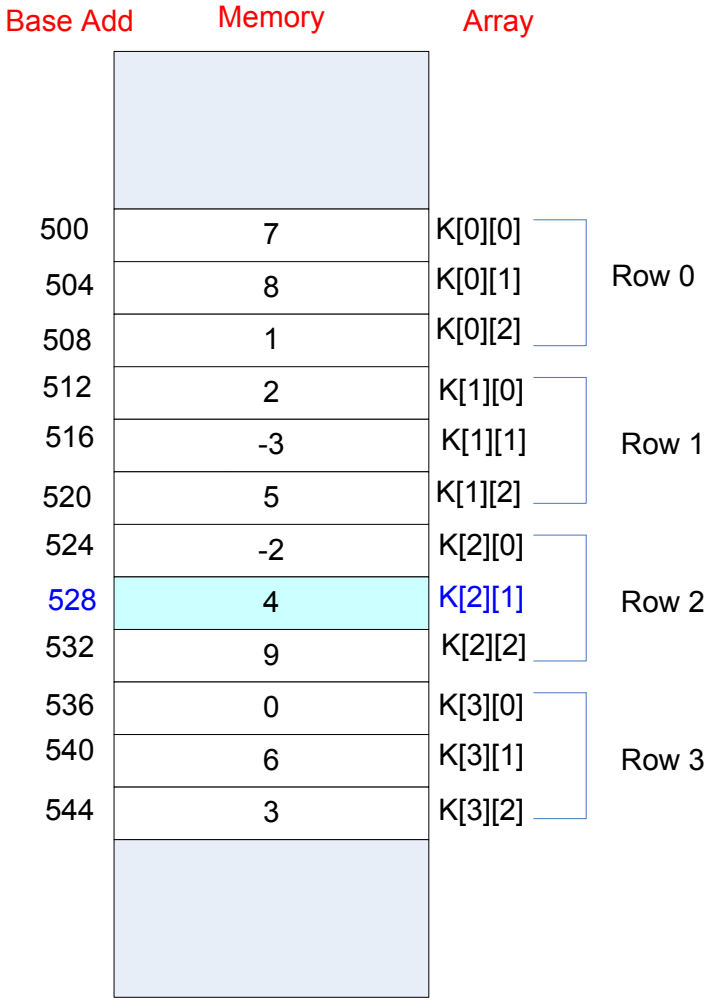
= 500+4[6+1]

= 500+28

= 528

ดังนั้นอาร์เรย์ K แถวที่ 2 คอลัมน์ 1 จะจัดเก็บอยู่ในตำแหน่ง

แอดเดรสที่ 528



แถวเป็นหลัก

- ข้อมูลอาร์เรย์ A ขนาด 4x5
  ต้องการคำนวณตำแหน่งแอดเดรสที่เก็บข้อมูลแถวที่ 3 คอลัมน์ 4 (A[3,4])
- กำหนดให้แอดเดรสเริ่มต้นที่ 500
- ข้อมูลที่จัดเก็บกินเนื้อที่ 4 ไบต์ต่อสมาชิก1ตัว

สูตรการคำนวณหาตำแหน่งที่ใช้เก็บข้อมูลในอาร์เรย์สองมิติ

1. แบบการเรียงคอลัมน์เป็นหลัก

สูตร

$$LOC(K[i,j]) = B + w[R * (j - L_2) + (i - L_1)]$$

โดยที่

LOC(K[i,j]) = ตำแหน่งแอดเดรสที่เก็บ K[i,j] ในหน่วยความจำ

B = แอดเดรสเริ่มต้น (Base Address)

w = จำนวนช่องของหน่วยความจำที่จัดเก็บข้อมูลต่อหนึ่งสมาชิก

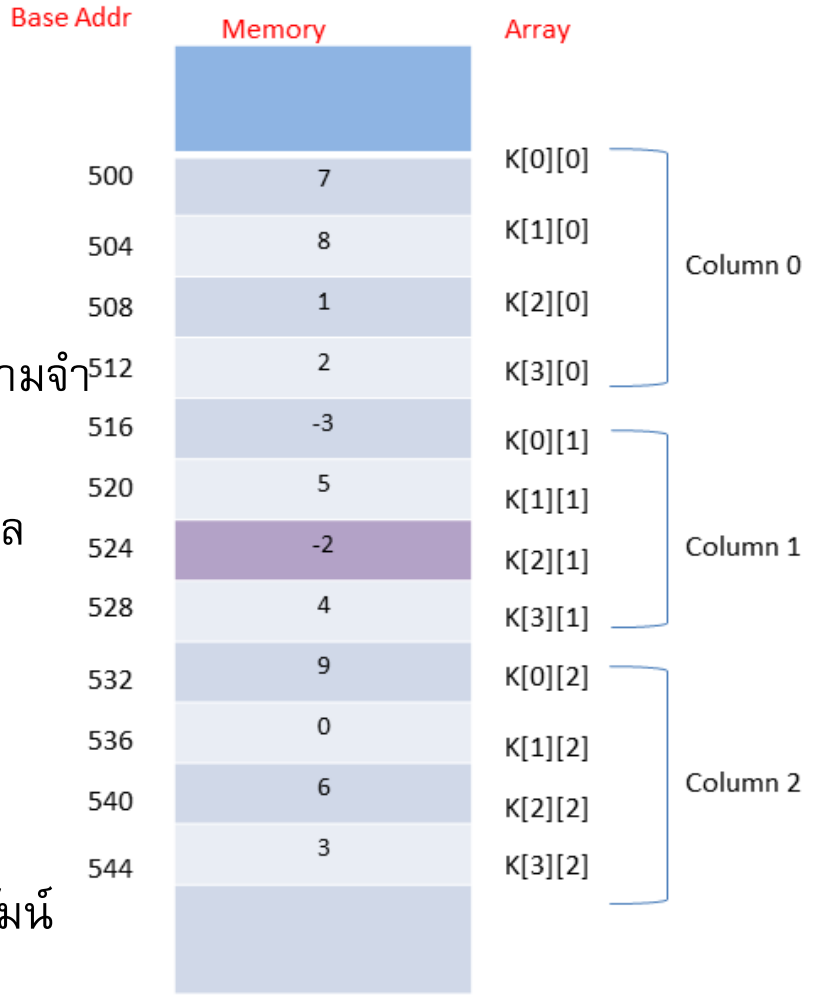
i = ตำแหน่งของแถวในอาร์เรย์

j = ตำแหน่งของคอลัมน์ในอาร์เรย์

L_1 = ค่าขอบเขตล่างสุด (lower Bound) ของแถว

L_2 = ค่าขอบเขตล่างสุด (lower Bound) ของคอลัมน์

R = จำนวนแถวของแถวลำดับ



การคำนวณ

สูตร

LOC(K[i, j]) = B + w [R * (j-L2) + (i-L1)]

LOC(K[2, 1]) = 500+4[4(1-0) + (2-0)]

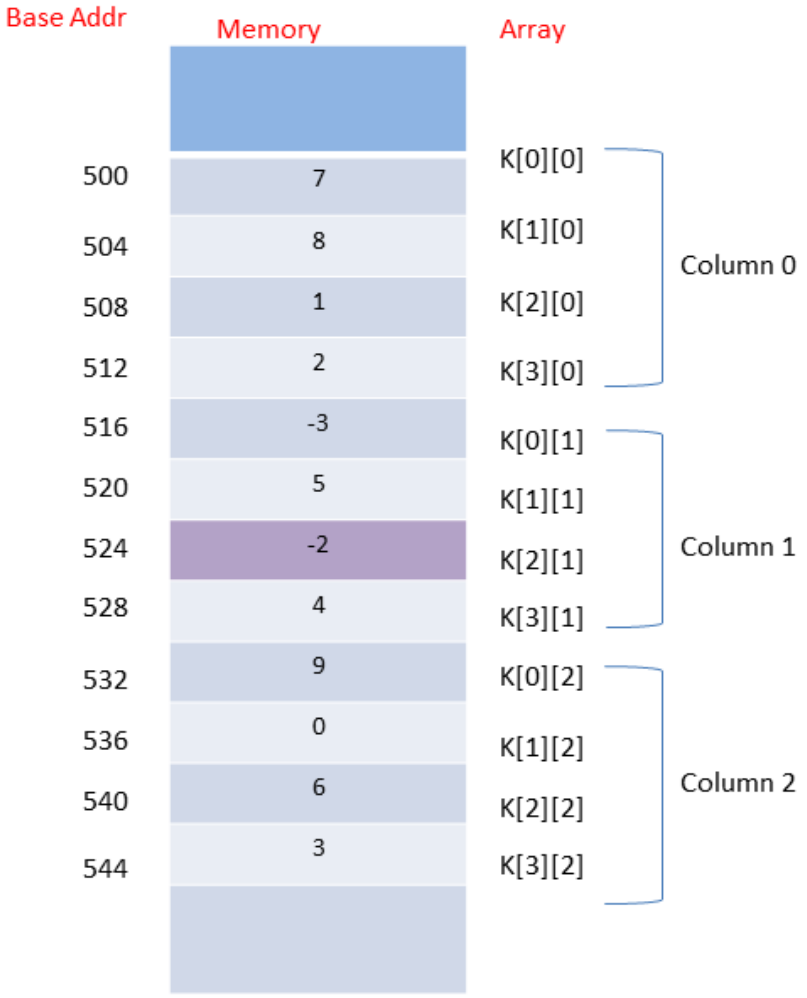
= 500+4[4+2]

= 500+24

= 524

ดังนั้นอาร์เรย์ K แถวที่ 2 คอลัมน์ 1 จะจัดเก็บอยู่ในตำแหน

แอดเดรสที่ 524

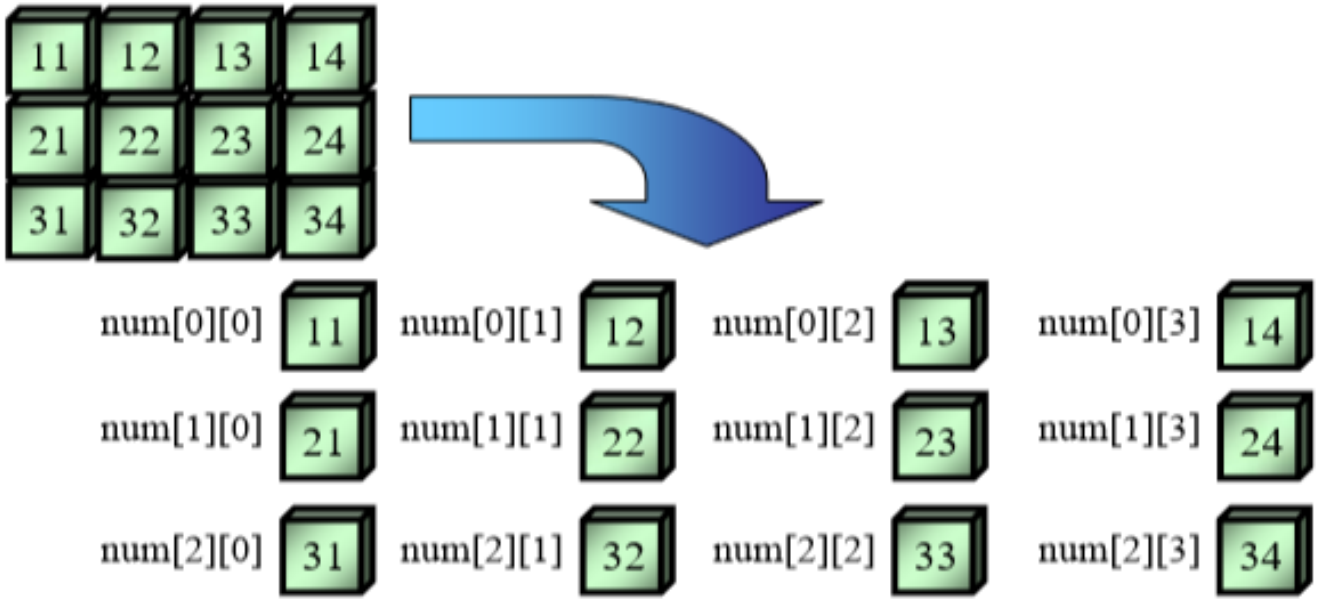


ให้คำนวณแบบ คอลัมน์เป็นหลัก

- ข้อมูลอาร์เรย์ A ขนาด 5x6
  ต้องการคำนวณตำแหน่งแอดเดรสที่เก็บข้อมูล (A[4,5])
- กำหนดให้แอดเดรสเริ่มต้นที่ 500
- ข้อมูลที่จัดเก็บกินเนื้อที่ 4 ไบต์ต่อสมาชิก1ตัว

ตัวอย่างการกำหนดข้อมูลตัวแปร

```
int     num[3][4] = { 11, 12, 13, 14,
                      21, 22, 23, 24,
                      31, 32, 33, 34 };
int     num[3][4] = { 11, 12, 13, 14, 21, 22, 23, 24, 31, 32, 33, 34 };
```

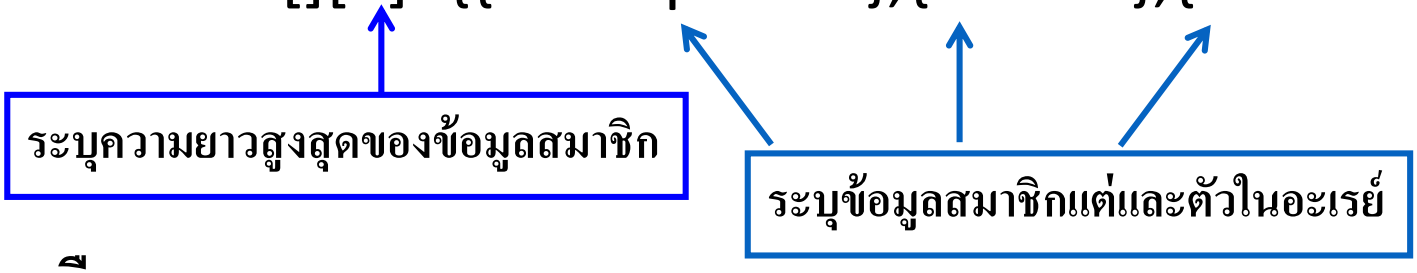


ที่มา [8] สไลด์ วิชา 90102003 Computer and Programming ภาควิชาวิศวกรรมคอมพิวเตอร์ คณะวิศวกรรมศาสตร์ สถาบันเทคโนโลยีพระจอมเกล้าเจ้าคุณทหารลาดกระบัง http://www.ce.kmitl.ac.th

การประกาศตัวแปร string

- สามารถประกาศและกำหนดค่าเริ่มต้นให้กับ string ได้ ดังนี้

```
char str2[][8]={{“Computer”},{”cs231”},{”Kaset”}};
```



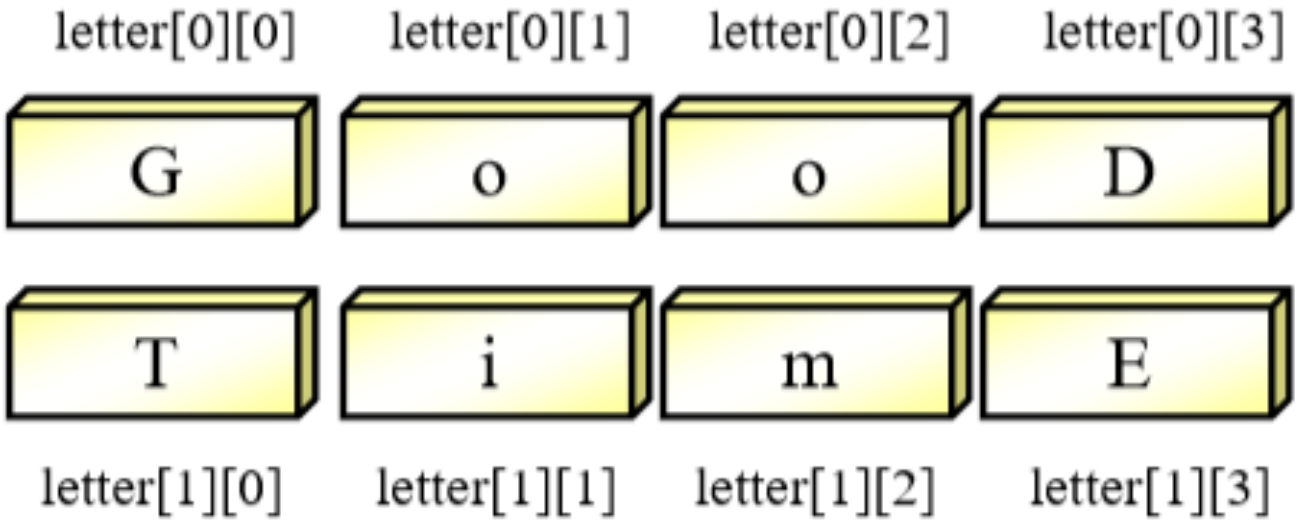
หรือ

```
char str3[][8]={“Computer”,”cs231”,”Kaset”};
```

ตัวอย่างการกำหนดข้อมูลตัวแปร

```
char   letter[2][4] = {     'G', 'o', 'o', 'D',
                            'T', 'i', 'm', 'E'};

char   letter[2][4] = {'G', 'o', 'o', 'D',    'T', 'i', 'm', 'E'};
```

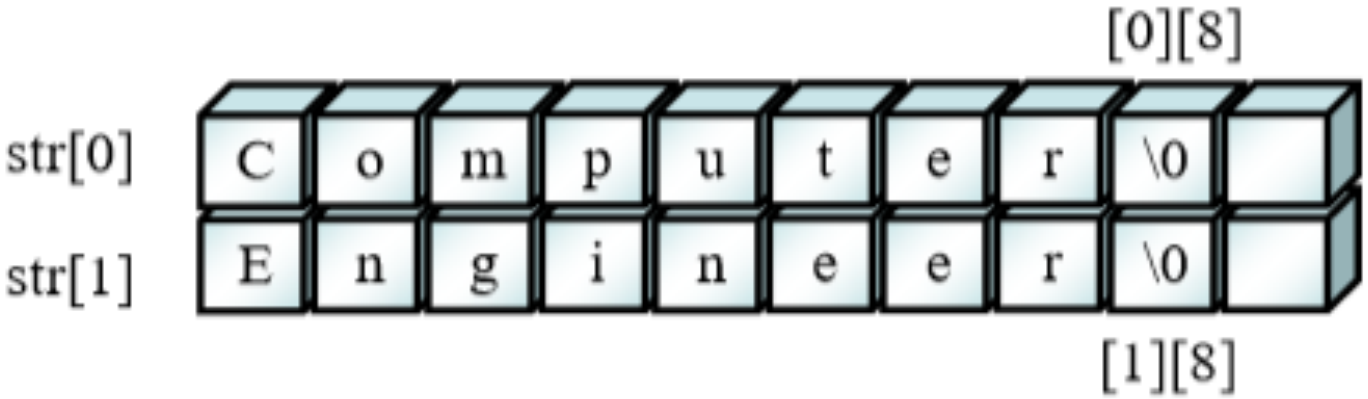


ที่มา [8] สไลด์ วิชา 90102003 Computer and Programming ภาควิชาวิศวกรรมคอมพิวเตอร์ คณะวิศวกรรมศาสตร์ สถาบันเทคโนโลยีพระจอมเกล้าเจ้าคุณทหารลาดกระบัง http://www.ce.kmitl.ac.th

ตัวอย่างการกำหนดข้อมูลตัวแปร

```
char str[2][10] = {"Computer",
                   "Engineer" };
```

```
char str[2][10] = {"Computer", "Engineer" };
```



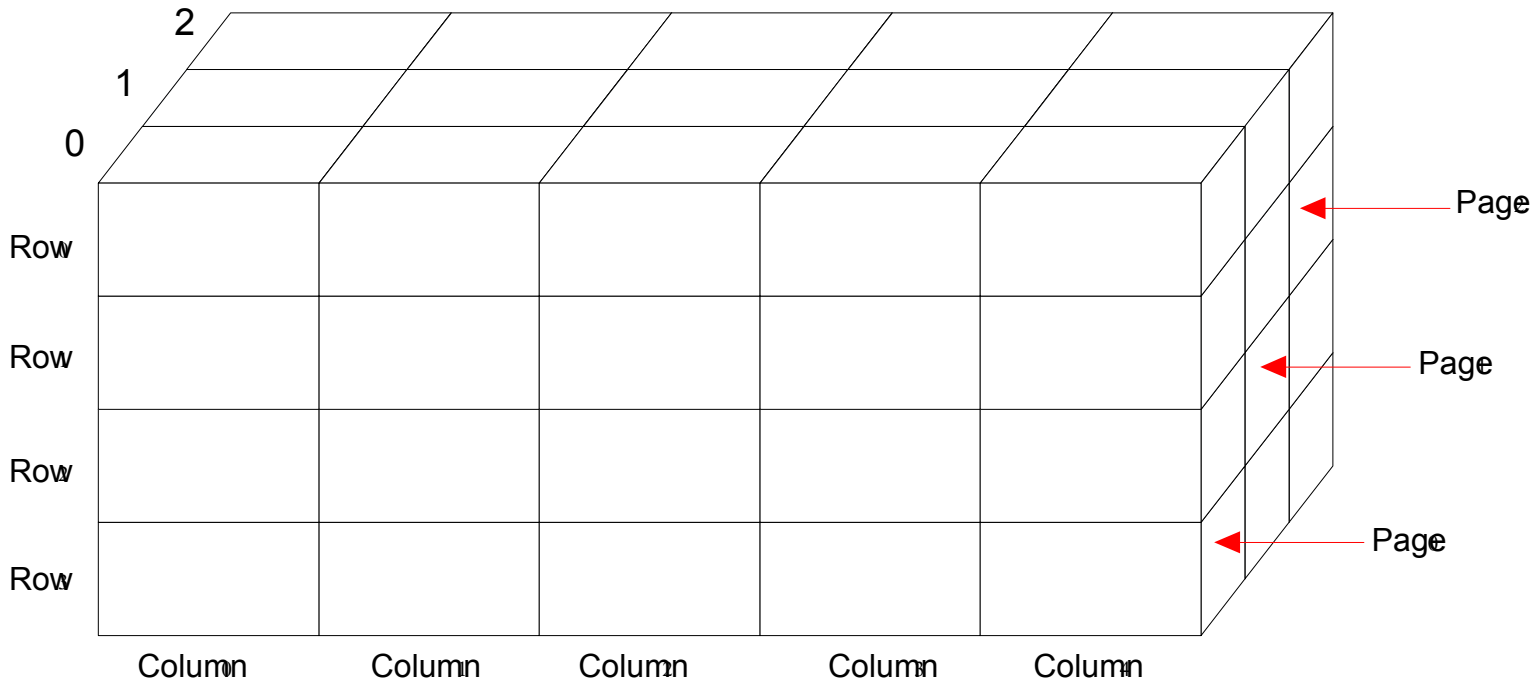
ที่มา [8] สไลด์ วิชา 90102003 Computer and Programming ภาควิชาวิศวกรรมคอมพิวเตอร์ คณะวิศวกรรมศาสตร์ สถาบันเทคโนโลยีพระจอมเกล้าเจ้าคุณทหารลาดกระบัง http://www.ce.kmitl.ac.th

ตัวอย่างโปรแกรม (L20.c)

```c
#include<stdio.h>
int main() {
    int matrix[3][3],r,c;
    for (r=0; r<3; r++) {
        for(c=0;c<3; c++){
            printf("Enter numbers[%d][%d]:",r,c);
            scanf ("%d",&matrix[r][c]);
        }
    }
    printf("Matrix\n");
    for(r=0;r<3;r++){
        for(c=0;c<3;c++){
            printf("%5d",matrix[r][c]);
        }
        printf("\n");
    }
    return 0;
}
```

อาร์เรย์สามมิติ (Three Dimension Array)

- อาร์เรย์สามมิติคือการนำเอาอาร์เรย์สองมิติมาเรียงซ้อนกันหลายๆชั้น (page) ดังนั้นจึงทำให้อาร์เรย์สามมิติ จะมีลักษณะเป็นแถวและคอลัมน์แล้วก็จะมีความลึกเพิ่มขึ้นมา



รูปแบบทั่วไปของโครงสร้างข้อมูลอาร์เรย์ 3 มิติ

ArrayName[L_1 : U_1 , L_2 : U_2 , L_3 : U_3]

เมื่อ ArrayName คือ ชื่อของโครงสร้างข้อมูลอาร์เรย์

L_1 คือ ค่าขอบเขตล่างสุด (Lower Bound) ของแถว

U_1 คือ ค่าขอบเขตสูงสุด (Upper Bound) ของแถว

L_2 คือ ค่าขอบเขตล่างสุด (Lower Bound) ของคอลัมน์

U_2 คือ ค่าขอบเขตสูงสุด (Upper Bound) ของคอลัมน์

L_3 คือ ค่าขอบเขตล่างสุด (Lower Bound) ของความลึก

U_3 คือ ค่าขอบเขตสูงสุด (Upper Bound) ของความลึก

การหาจำนวนสมาชิกของอาร์เรย์ 3 มิติ

- หาจากสูตร

  จำนวนสมาชิก = $(U_1-L_1+1) * (U_2-L_2+1) * (U_3-L_3+1)$

- เช่น จำนวนสมาชิกของอาร์เรย์ A(2,3,4) หรือ A(0:1,0:2,0:3)

  จำนวนสมาชิก = (1-0+1)*(2-0+1)*(3-0+1)

  = 2*3*4

  = 24

การประมวลผลสายอักขระ (string manipulation)

- สามารถใช้ความรู้เกี่ยวกับเรื่อง control มาจัดการกับ string ได้
- นำอักขระพิเศษ '\0' ที่ปิดท้าย string มาใช้ประโยชน์ในการตรวจสอบ

เช่น

- นับความยาวของ string
- เปรียบเทียบ string 2 ชุด
- ต่อ string 2 ชุด ให้เป็น 1 ชุด

นับความยาวของ string(L24.C)

```c
#include <stdio.h>
int main() {
    char word[] = "CS231 Data Structure";
    int   count_str(char s[]);
    printf("Length is %d  \n",count_str(word));
    return 0;
}

int count_str(char s[]) {
    int count=0;
    int i=0;
    while(s[i]!= '\0'){
        count++;
        i++;
    }
    return count;
}
```

ต่อ string 2 ชุด เข้าด้วยกัน (L25_2.C)

```c
#include <stdio.h>
int main() {
    char s1[100]="Hello_";
    char s2[]   ="Student!";
    int  i=0,j=0;
    printf("s1=%s s2=%s\n",s1,s2);

    while(s1[i] != '\0') {
        i++;
    }
    while(s2[j] != '\0') {
        s1[i]=s2[j];
        i++;
        j++;
    }
    s1[i]= '\0';

    printf("After concatenate\n");
    printf("s1=%s s2=%s\n",s1,s2);
    return 0;
}
```

ฟังก์ชันในการจัดการกับstring

- ในภาษาซี มีการสร้างฟังก์ชันเพื่อนำมาจัดการกับ string โดยผู้ใช้จะต้อง เรียกใช้ไลบารี string.h

- ฟังก์ชันต่างๆ เกี่ยวกับ string ดังนี้
- strcmp(), strcpy(), strlen(), strcat(), strncpy() เป็นต้น

ฟังก์ชัน strcmp()

ทำหน้าที่เปรียบเทียบ **string 2** ชุด

Syntax:

```
int strcmp(char *str1, char *str2);
```

str1, str2 คือ ตัวแปร **string** ที่นำมาเปรียบเทียบ

การเปรียบเทียบ ถ้า **str1< str2** ได้ผลลัพธ์น้อยกว่าศูนย์

ถ้า **str1 = str2** ได้ผลลัพธ์เท่ากับศูนย์

ถ้า **str1 > str2** ได้ผลลัพธ์มากกว่าศูนย์

ผลลัพธ์ที่ได้จะส่งกลับมาในรูปแบบจำนวนเต็ม

```
#include <string.h>
#include <stdio.h>

int main(void)
 {
   char *buf1 = "aaa", *buf2 = "bbb", *buf3 = "ccc";
   int ptr;

   ptr = strcmp(buf2, buf1);
   if (ptr > 0)
     printf("buffer 2 is greater than buffer 1\n");
   else
     printf("buffer 2 is less than buffer 1\n");

   ptr = strcmp(buf2, buf3);
   if (ptr > 0)
     printf("buffer 2 is greater than buffer 3\n");
   else
     printf("buffer 2 is less than buffer 3\n");

   return 0;
 }
```

ผลลัพธ์ buffer2 is greater than buffer1

buffer2 is less than buffer3

ฟังก์ชัน strcpy()

- **คัดลอกข้อมูลจาก string หนึ่ง ไปยังอีก string หนึ่ง โดย string ที่เป็นตัวหลักจะต้องมีขนาดที่มากกว่าหรือเท่ากับ string ที่จะนำมาคัดลอก ซึ่งการคัดลอกจะคัดลอกทีละอักขระจนกระทั่งพบอักขระ ‘\0’ จึงจะหยุดคัดลอก และจะไม่คัดลอก ‘\0’ ไปด้วย**

Syntax:

```
char * strcpy(char *dest, char *src);
```

dest คือ ตัวแปร string หรือ พอยน์เตอร์ชนิดอักขระ ปลายทาง

src คือ ตัวแปร string หรือ พอยน์เตอร์ชนิดอักขระ ต้นทาง

ฟังก์ชัน strcat()

- นำ string 2 ชุดมาเชื่อมต่อกัน ผลลัพธ์ที่ได้จะได้ string ที่มีความยาวเท่ากับ string ทั้ง 2 ชุดมารวมกัน

Syntax:

```
char * strcat(char *dest, char *src);
```

dest คือ ตัวแปร string หรือ พอยน์เตอร์ชนิดอักขระ ปลายทางที่เก็บผลลัพธ์จากการเชื่อมต่อ string

src คือ ตัวแปร string หรือ พอยน์เตอร์ชนิดอักขระ ต้นทางที่นำ string ไปเชื่อมต่อ

ตัวอย่าง

```c
#include <stdio.h>
#include <string.h>
int main() {
    char result[25];
    char *s1 = "Happy", *s2= " ", *s3 = "Holidays";

    strcpy(result,s1);
    strcat(result,s2);
    strcat(result,s3);

    printf("%s\n",result);
    return 0;
}
```

ฟังก์ชัน strlen()

- นับความยาวของ **string** ที่ต้องการ

Syntax:

```
int strlen(str);
```

str คือ ตัวแปร **string** ที่ต้องการนำมานับความยาวหรือนับจำนวนอักขระของ **string** โดยการนับนี้จะไม่นับอักขระพิเศษ **'\0'** และการนับจะสิ้นสุดเมื่อเจอ '\0'
ผลลัพธ์ที่ได้จะส่งกลับมาเป็นเลขจำนวนเต็ม

ตัวอย่าง

```
#include <stdio.h>
#include <string.h>

int main(void)
{
  char *string = "Borland International";

  printf("%d\n", strlen(string));
  return 0;
}
```

ผลลัพธ์

21

ฟังก์ชัน strncpy()

- คัดลอก **string** หนึ่ง ไปยังอีก **string** หนึ่ง โดยสามารถระบุความยาวมากที่สุดของ **string** ที่ต้องการคัดลอกไปยังปลายทางได้

  Syntax:

  ```
  char * strncpy(char *dest, char *src, size_t maxlen);
  ```

  dest คือ **string** ปลายทางที่จะนำ **string**ที่ต้องการมาคัดลอกลงไป

  src คือ **string** ต้นทางที่ต้องการคัดลอก

  maxlen คือ จำนวนความยาวมากที่สุดของ **string** ที่ต้องการคัดลอก

ตัวอย่าง

```
#include <stdio.h>
#include <string.h>

int main(void)
{
  char string[10];
  char *str1 = "abcdefghi";

  strncpy(string, str1, 3);
  string[3] = '\0';
  printf("%s\n", string);
  return 0;
}
```

ผลลัพธ์

abc

แบบฝึกหัด

- จงเขียนโปรแกรมเรียกใช้ฟังก์ชัน catn_str( ) ซึ่งทำหน้าที่ต่อ string 2 ชุดเข้าด้วยกันระบุจำนวนอักขระที่นำไปต่อได้ โดยห้ามใช้ฟังก์ชันสำเร็จรูป
- จงเขียนโปรแกรมเรียกใช้ฟังก์ชัน cmp_str( ) ซึ่งทำหน้าที่เปรียบเทียบ string 2 โดยห้ามใช้ฟังก์ชันสำเร็จรูป
- จงเขียนโปรแกรมเรียกใช้ฟังก์ชัน count_word( ) ซึ่งทำหน้าที่นับคำในประโยค(string)

แหล่งอ้างอิง

- สไลด์ ของ อ.วิวัฒน์ ชินนาทศิริกุล
- http://www.ce.kmitl.ac.th
- http://www.cs.science.cmu.ac.th
- https://www.cs.kku.ac.th