

# แถวลำดับ ข้อมูลชนิดอาร์เรย์

1. อาร์เรย์
2. อาร์เรย์สองมิติ
3. สตริง
4. ฟังก์ชันสตริง

Reference : ไฟล์ประกอบการสอน ศูนย์โรงเรียนสามเสนวิทยาลัย โดย อ.ธีรวัฒน์ ประกอบผล  
ไฟล์เนื้อหารายวิชา Computer Programming I คณะวิทยาศาสตร์ ม.ศิลปากร  
โดย อ.ภิญโญ แท้ประสาทสิทธิ์

# ตัวแปรอาร์เรย์คืออะไร

การประกาศตัวแปรใช้งานในโปรแกรม เช่น

```
int x = 0 ;
```

```
float num;
```

ถ้าเราต้องการใช้ตัวแปรในโปรแกรม 10 ตัว นักเรียนจะต้องประกาศตัวแปรดังนี้

```
int a1 , a2 ,a3 ,a4,a5,a6, ..... a10 ;
```

ภาษา C จะมีวิธีการสร้างตัวแปรสำหรับเก็บข้อมูลชนิดเดียวกันหลาย ๆ ตัวได้ โดยการประกาศใช้เป็นตัวแปรชนิดแถวลำดับ (Array)

# ข้อมูลชนิดอาร์เรย์

-12	23	45	65	12	27	86
-----	----	----	----	----	----	----



ตัวแปรอาร์เรย์เก็บจำนวนเต็ม 7 จำนวน

12.8	85.21	32.1	23.9	43.5
------	-------	------	------	------



ตัวแปรอาร์เรย์เก็บทศนิยม 5 จำนวน

# แถวลำดับ (Array)

ตัวแปรประเภทอาร์เรย์ เป็นตัวแปรที่สามารถเก็บข้อมูลหลาย ๆ ค่าไว้ในตัวแปรชื่อเดียวกันได้ โดยระบบจะใช้พื้นที่หน่วยความจำต่อเนื่องกัน เพื่อเก็บข้อมูลชนิดเดียวกันหลายจำนวน

การประกาศตัวแปรอาร์เรย์

ชนิดข้อมูล ชื่อตัวแปรอาร์เรย์[n];

จำนวนสมาชิกของตัวแปรอาร์เรย์

```
int A[10];  
float B[5];
```

ประเภทข้อมูลในตัวแปรอาร์เรย์

# ตัวอย่าง

- `int n[10];`

ประกาศตัวแปรอาร์เรย์ชื่อ `n` มีขนาด 10 หน่วย แต่ละหน่วยเก็บเลขจำนวนเต็ม

- `char a[20];`

ประกาศตัวแปรอาร์เรย์ชื่อ `a` มีขนาด 20 หน่วย แต่ละหน่วยเก็บตัวอักษร

- `float g[5];`

ประกาศตัวแปรอาร์เรย์ชื่อ `g` มีขนาด 5 หน่วย แต่ละหน่วยเก็บเลขทศนิยม

การอ้างถึงสมาชิกในตัวแปรอาร์เรย์ จะอ้างถึงได้ตั้งแต่ `[0].....[n-1]`

ถ้าหากประกาศตัวแปรเป็นสตริง ตัวแปรนั้นก็คืออาร์เรย์ของ `char` นั่นเอง

# ตัวอย่าง

- `int ar[5];` //ประกาศตัวแปรอาร์เรย์ `ar` มีสมาชิก 5 ตัว
- สมาชิกตัวที่ 1 ของตัวแปรอาร์เรย์ `ar` คือ `ar[0]`
- ดังนั้นสมาชิกของ `ar[5]` ประกอบไปด้วย

`ar[0], ar[1], ar[2], ar[3], ar[4]`

เลขจำนวนเต็มตั้งแต่ 0 ถึง  $n-1$



การอ้างถึงสมาชิกในอาร์เรย์

ชื่อตัวแปรอาร์เรย์[ดรรชนีกำกับ]

- `ar[0] = 15;` // กำหนดให้ `ar[0]` มีค่าเท่ากับ 15

- ถ้าหากเราต้องการเก็บเลขจำนวนเต็ม 10 ตัวไว้ด้วยกันภายใต้ชื่อ A เราเขียนว่า `int A[10];`
- ชนิดข้อมูลต้องนำหน้าชื่อเช่นเดียวกับการประกาศตัวแปรทั่วไป
- เราใช้วงเล็บเหลี่ยมหลังชื่อแวลวลำดับ และเราใส่ตัวเลขเข้าไปเพื่อบอกว่า **แวลวลำดับนี้จะเก็บข้อมูลได้สูงสุดกี่ตัว** ในที่นี้คือเก็บได้สูงสุด 10 ตัว
- สรุปความแตกต่างในการสร้างแวลวลำดับกับตัวแปรทั่วไปคือ แวลวลำดับจะมีวงเล็บเหลี่ยม (square bracket) และจำนวนข้อมูลที่จะรับได้ตามมา
  - แต่ตัวแปรทั่วไปจะมีแค่ชนิดข้อมูลและชื่อ
  - เปรียบเทียบ `int A;` กับ `int A[10];` แบบแรกเป็นตัวแปร `int` ทั่วไป แต่แบบที่สองคือแวลวลำดับที่เก็บ `int` ได้สูงสุด 10 ตัว

# ตัวอย่าง

ถ้าหากมีข้อมูลกลุ่มหนึ่งเป็นคะแนนของนักศึกษา 8 คน สามารถเก็บได้ดังนี้

หมายเลข	X[0]	X[1]	X[2]	X[3]	X[4]	X[5]	X[6]	X[7]
คะแนน	18	20	35	84	21	45	65	74

ถ้าหากมีการอ้างถึงอาเรย์อาจเป็นดังต่อไปนี้

X[2]	อ้างถึงเซลล์ที่ 2 มีค่าเท่ากับ 35
X[2] + X[3]	นำเซลล์ที่ 2 บวกกับเซลล์ที่ 3 จะได้ 35 + 84 เท่ากับ 119
X[1+3]	อ้างเซลล์ที่ 4 มีค่าเท่ากับ 21
X[5] + 1	นำเซลล์ที่ 5 มาบวกด้วย 1 จะได้เท่ากับ 46



# ตัวอย่างการอ้างอิงข้อมูลในแถวลำดับ

การประกาศ `int A[10];` หมายความว่า A คือแถวลำดับที่เก็บ `int`

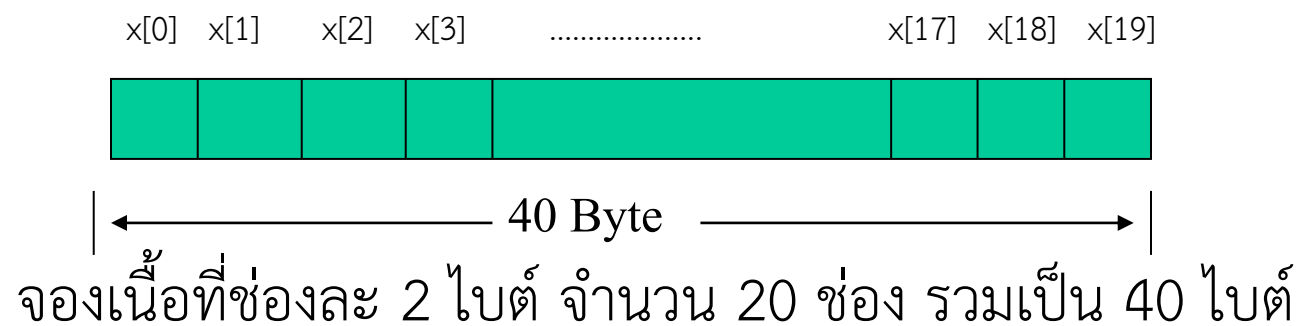
- ส่วน `A[ตัวเลขลำดับ]` คือข้อมูลในแถวลำดับ เช่น ถ้าตัวเลขลำดับคือ 1 หมายถึงข้อมูลตัวที่สอง ถ้าตัวเลขลำดับคือ 9 หมายถึงข้อมูลตัวที่ 10
- การเขียนว่า `A[ตัวเลขลำดับ]` จะให้ผลเหมือนตัวแปรทั่วไปแทบทุกอย่าง
- การรับค่าจาก `scanf` ทำได้เหมือนตัวแปรทั่วไป (ถ้ามีเลขลำดับประกอบ)
  - `scanf("%d", &A[0]);` เป็นการอ่านข้อมูลจากผู้ใช้งานเก็บไว้ที่ข้อมูลตัวแรก
  - `scanf("%d", &A[7]);` เป็นการอ่านข้อมูลจากผู้ใช้งานเก็บไว้ที่ข้อมูลตัวที่ 8
- การแสดงผลจาก `printf` ก็ทำได้เหมือนกับตัวแปรทั่วไปเช่นกัน
  - `printf("%d", A[0]);` เป็นการพิมพ์ค่าของแถวลำดับตัวแรก
  - `printf("%d", A[7]);` เป็นการพิมพ์ค่าของแถวลำดับตัวที่แปด

# ขนาดหน่วยความจำของอาร์เรย์

ขึ้นอยู่กับประเภทของข้อมูลและจำนวนสมาชิกที่จองไว้

## ตัวอย่าง

```
int x[20];
```



```
char name[20];
```

↑ ใช้พื้นที่ 20 ไบต์

# ตัวอย่าง

โปรแกรมรับข้อมูล 10 ค่า แล้วหาผลรวมของข้อมูลเหล่านั้น

```
#include <stdio.h>
main()
{
    int num[10],sum,i;
    for(i = 0; i < 10; i++)
        { scanf("%d",&num[i]); }
    sum = 0;
    for(i = 0; i < 10; i++)
        { sum = sum + num[i]; }
    printf("sum is %d\n",sum);
}
```

**ตัวอย่าง** จงเขียนโปรแกรมที่รับค่าตัวเลขจำนวนเต็มจากผู้ใช้งาน 10  
ค่า จากนั้นให้พิมพ์ตัวเลขทั้งหมดออกมาเรียงลำดับจากหลังไปหน้า

# โค้ดตัวอย่าง

```
void main() {  
    int A[10];  
  
    int i;  
    for(i = 0; i < 10; i++) {  
        scanf("%d", &A[i]);  
    }  
  
    for(i = 9; i >= 0; i--) {  
        printf("%d ", A[i]);  
    }  
}
```

# การกำหนดค่าเริ่มต้นให้อาร์เรย์

ชนิดของข้อมูล ชื่อตัวแปรอาร์เรย์[ขนาด] = {value-list};


## ตัวอย่าง

```
int n[5] = {1,4,9,16,25};
```

```
char a[3] = {'A','B','C'};
```

```
int pw[ ] = {1,2,4,8,16,32,64,128};
```


```
char name[ ] = "COMPUTER";
```

ถ้าไม่ระบุขนาด โปรแกรมจะจองหน่วยความจำให้เอง

โปรแกรมอ่านค่าจากอาร์เรย์ และแสดงเป็นกราฟแท่ง

```
#include<stdio.h>
#define SIZE 10
main()
{
    int i,j;
    int n[SIZE] = {19,3,15,7,11,9,13,5,17,1};
    printf("%s%13s%17s\n","Element","Value","Histogram");
    for(i=0; i<=SIZE-1; i++)
    {
        printf("%7d%13d  ",i,n[i]);
        for(j = 1; j<= n[i]; j++)
            printf("%c",'*');
        printf("\n");
    }
}
```

# OUTPUT

 "C:\Users\Orange\Documents\C pro\mynumber\test.exe"

Element	Value	Histogram
0	19	*****
1	3	***
2	15	*****
3	7	*****
4	11	*****
5	9	*****
6	13	*****
7	5	*****
8	17	*****
9	1	*

Process returned 10 (0xA)    execution time : 0.031 s  
Press any key to continue.



## สิ่งที่ต้องระวัง

ในภาษาซีจะไม่มีการกำหนดให้ตรวจสอบขอบเขตของอาร์เรย์ โปรแกรมเมอร์จะต้องพยายามเขียนโปรแกรมที่เกี่ยวข้องกับสมาชิกของอาร์เรย์ภายในขอบเขตที่ประกาศอาร์เรย์ไว้ หากมีการอ้างอิงถึงสมาชิกอาร์เรย์นอกขอบเขตที่ได้ระบุไว้ เช่น `table[12]` สิ่งที่ได้คือการไปอ่านข้อมูลในพื้นที่ของหน่วยความจำที่อาจจะเก็บค่าของตัวแปรตัวอื่น หรือค่าอื่นใดที่ไม่อาจคาดเดาได้

## ตัวอย่าง 6.2

ให้อ่านค่าของจำนวนเต็ม 5 จำนวนจากคีย์บอร์ด  
และแสดงผลในลำดับที่กลับกัน

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

```
#define SIZE 5
```

```
main ( ) {
```

```
    int k;
```

```
    int table[SIZE];
```

```
    for (k = 0; k < SIZE; k++)
```

```
        scanf ("%d", &table[k]);
```

```
    for (k = SIZE-1; k >= 0; k--)
```

```
        printf ("%d\n", table[k]);
```

```
}
```

1 2 3 4 5

5 4 3 2 1

# ชนิดตัวแปรอาร์เรย์

1. ตัวแปรอาร์เรย์ 1 มิติ คือ ตัวแปรอาร์เรย์ที่เก็บข้อมูลเพียงแถวเดียว เปรียบได้กับตารางที่มีหนึ่งแถว ดังนี้

12	25	4	48	16
----	----	---	----	----

a[0] a[1] a[2] a[3] a[4]

2. ตัวแปรอาร์เรย์หลายมิติ (2 มิติ ตารางที่มีทั้งแถวและคอลัมน์) เช่น

แถว 0 หลัก 0 a[0] [0]	→	2	3	←แถว 0 หลัก 1 a[0][1]
แถว 1 หลัก 0 a[1] [0]	→	6	9	←แถว 1 หลัก 1 a[1][1]

# อาร์เรย์สองมิติ

การประกาศตัวแปรอาร์เรย์สองมิติจะใช้ดัชนี 2 ตัว เพื่อระบุจำนวนสมาชิกในแต่ละหลัก และแต่ละแถว ดังนี้

ชนิดข้อมูล ชื่อตัวแปรอาร์เรย์[Row][Column]

ตัวอย่างเช่น

```
int AB[2][3];
```

จะมีสมาชิกทั้งหมด 6 ตัว ( $2 \times 3$ ) การอ้างสมาชิกแต่ละตัวทำได้ดังนี้

แถวที่ 0       $AB[0][0]$ ,  $AB[0][1]$ ,  $AB[0][2]$

แถวที่ 1       $AB[1][0]$ ,  $AB[1][1]$ ,  $AB[1][2]$

# การกำหนดค่าเริ่มต้นให้อาร์เรย์ 2 มิติ

ตัวอย่าง

```
int sqr[3][3] = {  
    1, 2, 3,  
    4, 5, 6,  
    7, 8, 9  
};  
  
int B[2][2] = { {1,2},{3,4} };
```

# ตัวอย่าง

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

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

โปรแกรมอ่านข้อมูลจำนวนเต็มจากตัวแปรอาร์เรย์สองมิติ แล้วนำข้อมูลทั้งหมดมาบวกกัน

```
#include <stdio.h>
main()
{
    int i, j, sum;
    int b[5][4];
    sum = 0;
    for(i = 0; i < 5; i++)
        for(j = 0; j < 4; j++)
        {
            scanf("%d", &b[i][j]);
            sum = sum + b[i][j];
        }
    printf("The sum is %d\n",sum);
}
```

# การเก็บข้อมูลในอาร์เรย์สองมิติ

```
main( )
```

```
{
```

```
    int a[3][3];
```

```
    a[0][0] = 4;
```

```
    a[0][1] = 8;
```

```
    a[0][2] = 9;
```

```
    a[1][0] = 5;
```

```
    a[1][1] = 14;
```

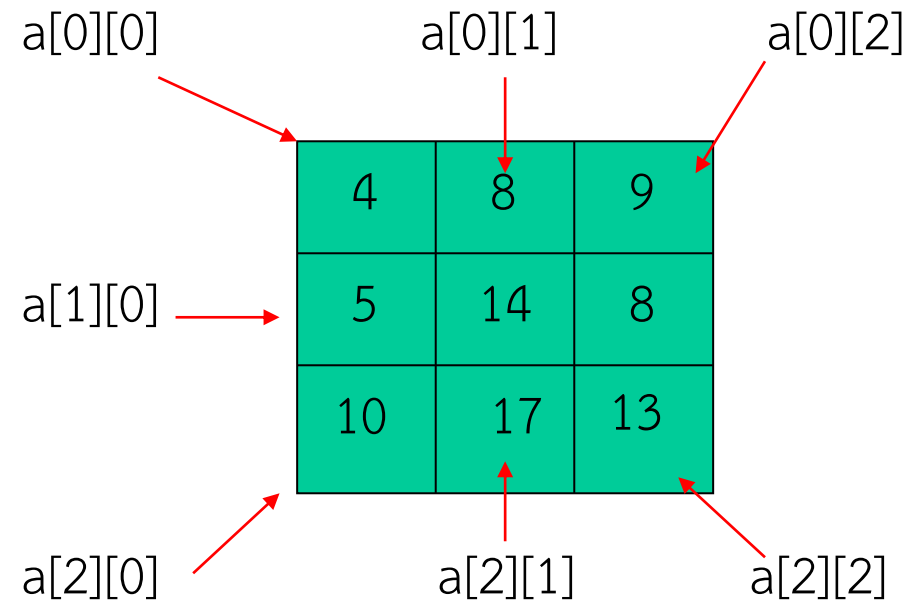
```
    a[1][2] = 8;
```

```
    a[2][0] = 10;
```

```
    a[2][1] = 17;
```

```
    a[2][2] = 13;
```

```
}
```





# การเก็บข้อมูลในอาร์เรย์สองมิติ

```
main( )
```

```
{
```

```
    int a[3][3]={4,8,9},{5,14,8},{10,17,13}};
```

```
    int x, y;
```

```
    for(y = 0; y <= 2; y++)
```

```
    {
```

```
        for(x = 0; x <= 2; x++)
```

```
            printf("%d\t",a[y][x]);
```

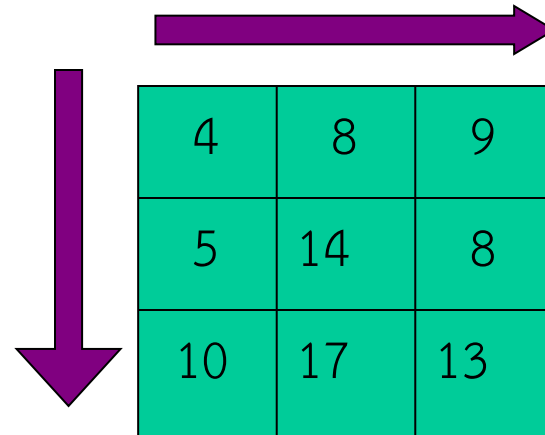
```
        printf("\n");
```

```
    }
```

```
}
```

ตัวแปร y

ตัวแปร x



A 3x3 grid representing a 2D array. The grid is filled with light blue cells. A purple arrow points downwards from the top-left cell to the bottom-left cell, labeled 'ตัวแปร y'. A purple arrow points from the top-left cell to the top-right cell, labeled 'ตัวแปร x'.

4	8	9
5	14	8
10	17	13

OUTPUT

```
4      8      9
5      14     8
10     17     13
```

# การส่งตัวแปรอาร์เรย์เป็นอาร์กิวเมนต์

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

```
void funct(int [ ]);
```

```
void main( )
```

```
{
```

```
    int arrayp[20];
```

```
    .....
```

```
    funct(arrayp);
```

```
    .....
```

```
}
```


```
void funct(int arraya[ ])
```

```
{
```

```
    .....
```

```
}
```

เรียกใช้ฟังก์ชัน funct() โดยมี arrayp[ ] เป็นอาร์กิวเมนต์



โปรแกรมอ่านข้อมูลจำนวนเต็มจากตัวแปรรายๆ แล้วนำข้อมูลทั้งหมดมาบวกกัน

```
#include <stdio.h>

float sum(float [ ], int);

main()
{
    int i;

    float item[100];

    for(i = 0; i < 100; i++)
        scanf("%f", &item[i]);

    printf("Sum is
    %f\n",sum(item,100));
}
```

```
float sum(float a[ ], int n)
{
    int i;

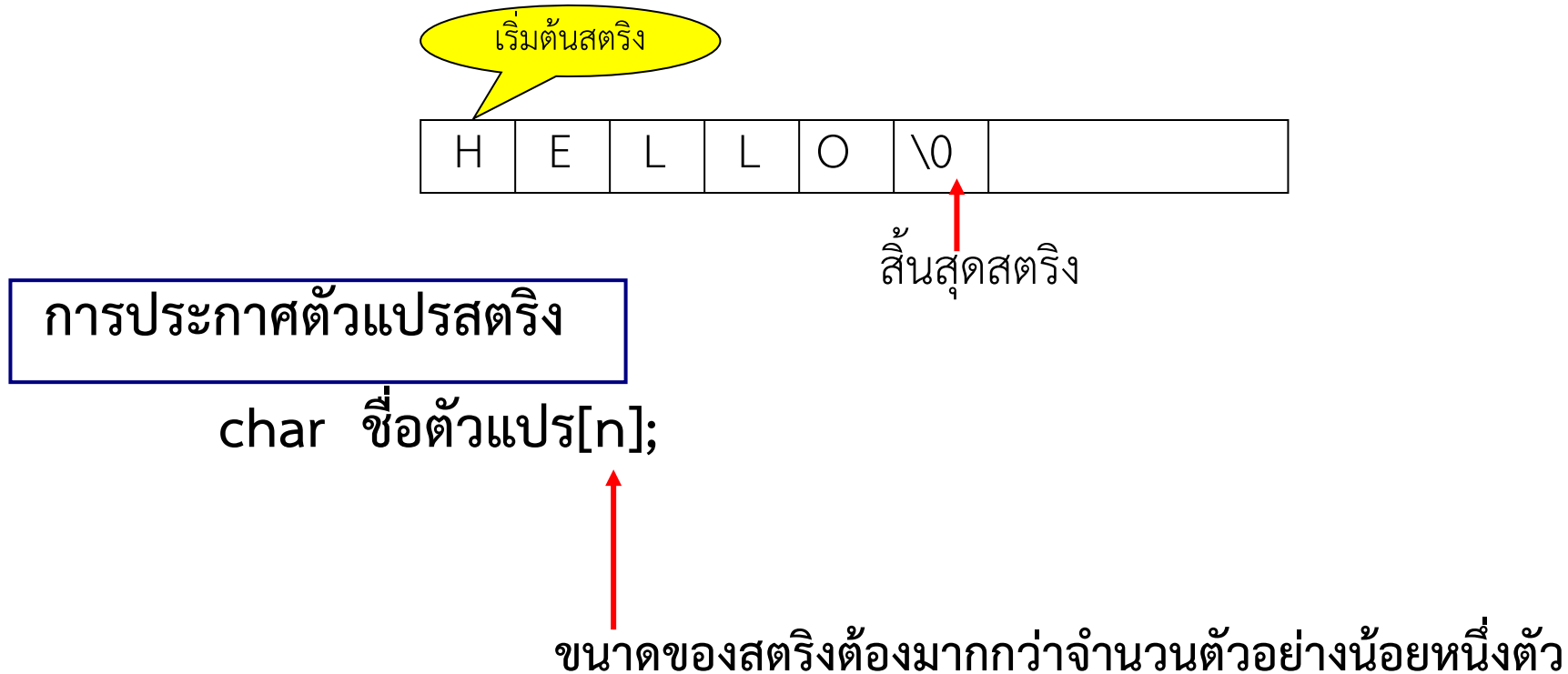
    float s = 0.0;

    for(i = 0; i < n; i++)
        s = s + a[i];

    return (s);
}
```

# ข้อมูลแบบสตริง

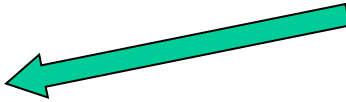
เป็นตัวแปรแบบอักขระมาต่อเรียงกัน โดยใช้ตัวอักขระ null หรือ “\0” เป็นตัวสิ้นสุดสตริง



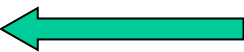
# การใส่ค่าในตัวแปรสตริง

ทำได้ดังนี้


1. ใช้ฟังก์ชันรับข้อมูล เช่น scanf( ), gets( ) เป็นต้น
2. กำหนดตอนประกาศตัวแปร
3. ใช้ฟังก์ชัน strcpy( ) ที่เก็บอยู่ใน string.h



```
gets(name);  
scanf("%s",&name);
```



```
char name[20] = "COMPUTER";
```



```
main()  
{  
    char name[20];  
    strcpy(name,"COMPUTER");  
    printf("%s",name);  
}
```

รูปแบบ

```
strcpy(ตัวแปรสตริง, "ข้อความสตริง");
```

# ฟังก์ชันของตัวแปรสตริง

## ฟังก์ชัน strcat()

นำสตริงสองตัวมาต่อกัน มีรูปแบบดังนี้

```
strcat(สตริง1 , สตริง2);
```

## ฟังก์ชัน strcmp()

นำสตริงสองตัวมาเปรียบเทียบกัน มีรูปแบบดังนี้

```
strcmp(สตริง1 , สตริง2);
```

ผลการเปรียบเทียบ	ค่าที่ส่งกลับ
สตริง1 < สตริง2	จำนวนลบ
สตริง1 = สตริง 2	ศูนย์
สตริง1 > สตริง 2	จำนวนบวก

# ฟังก์ชันของตัวแปรสตริง

ฟังก์ชัน strcpy()

คัดลอกสตริงต้นทางไปไว้ปลายทาง มีรูปแบบดังนี้

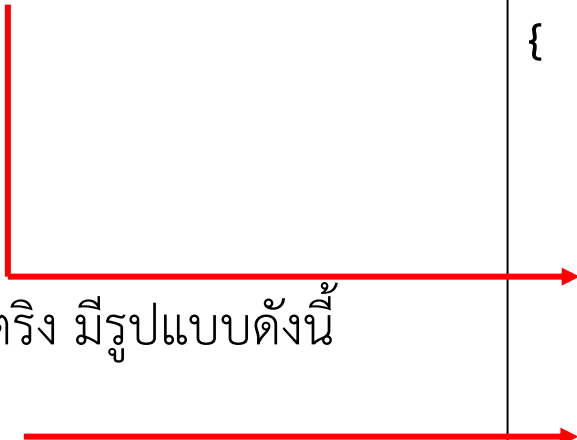
strcpy(สตริงปลายทาง , สตริงต้นทาง);

ฟังก์ชัน strlen()

นับจำนวนอักขระในสตริง มีรูปแบบดังนี้

strlen(สตริง);

```
main()
{
    int i;
    char str1[20];
    strcpy(str1,"Have a nice day");
    printf("%s\n",str1);
    i = strlen(str1);
    printf(" %d\n", i);
}
```



# การใช้แถวลำดับสองมิติแทนภาพ

- ภาพที่เราเห็นเป็นสองมิติ เราสามารถใช้แถวลำดับสองมิติแทนภาพได้
- เราสามารถเขียนภาพลงในแถวลำดับจนเสร็จแล้วพิมพ์ภาพออกมาทีเดียว
- ข้อดีของการใช้แถวลำดับก็คือ ตอนเราแก้ไขภาพเราไม่ต้องแก้ไขทีละแถวก็ได้ เราแก้ตำแหน่งไหนก่อนก็ได้

เช่น จงเขียนภาพกรอบขนาด  $5 \times 7$  (สูง  $\times$  กว้าง) โดยให้ขอบเป็นเครื่องหมาย \*

```
* * * * *
*           *
*           *
*           *
*           *
* * * * *
```



# แก่นของการใส่ดอกจัน

```
void main() {  
    char A[5][7];
```

```
..... (1) ส่วนเตรียมค่า
```

```
.....  
    for(col = 0; col < 7; ++col) {
```

```
        A[0][col] = '*';
```

```
        A[4][col] = '*';
```

ใส่ดอกจันในแถวแรกและแถวสุดท้าย

```
    }
```

```
    for(row = 0; row < 5; ++row) {
```

```
        A[row][0] = '*';
```

```
        A[row][6] = '*';
```

ใส่ดอกจันในคอลัมน์แรกและสุดท้าย

```
    }
```

```
..... (2) ส่วนพิมพ์ผลลัพธ์
```

```
.....
```

```
}
```

ในตอนที่เรายังไม่พิมพ์ เราจะใส่ดอกจันตามแถวหรือคอลัมน์ก็ได้ทั้งนั้น

## ตอนพิมพ์ก็วิ่งลูปพิมพ์ออกมาทีละแถว

```
char A[5][7];
```

(2) ส่วนพิมพ์ผลลัพธ์

```
for(row = 0; row < 5; ++row) {  
    for(col = 0; col < 7; ++col) {  
        printf("%c", A[row][col]);  
    }  
    printf("\n");  
}
```

ดึงตัวอักษรที่เก็บไว้ใน A ออกมาพิมพ์  
ทีละตัวไปเรื่อย ๆ จนหมด

สังเกตด้วยว่าส่วนของการคิดตำแหน่งดอกรันกับการพิมพ์จะแยกออกจากกัน

→ ช่วยให้เราแบ่งงานเขียนโปรแกรมออกเป็นหลายส่วนที่ไม่ซับซ้อนได้

# อย่าลืมเตรียมค่าเริ่มต้นในภาพ

ค่าในแถวลำดับตอนแรกจะมีอะไรก็ไม่อาจทราบได้ ดังนั้นต้องเตรียมค่าไว้ก่อน

```
void main() {  
    char A[5][7];  
  
    int row, col;  
    for(row = 0; row < 5; ++row) {  
        for(col = 0; col < 7; ++col) {  
            A[row][col] = ' '  
        }  
    }  
    .....  
}
```

เริ่มแรกกำหนดให้ตัวอักษรเป็นช่องว่าง  
ให้หมดจากนั้นค่อยใส่ดอกจันทับลงไป

การเตรียมค่าเริ่มต้นแล้วเขียนทับค่าเดิมเป็นเทคนิคที่พบบ่อยในการจัดการภาพ

# สรุปแนวคิดเรื่อง Array

- แถวลำดับเป็นสิ่งที่สำคัญมาก คาดว่าน่าจะได้เจอในโจทย์ข้อสอบคัดเลือกค่าย 1 แน่แน่นอน
- เวลาทำงานกับแถวลำดับสองมิติ ไม่ควรใช้ตัวแปรชื่อ  $i, j$  ในการอ้างถึงข้อมูล แต่ควรใช้คำว่า row และ col หรืออื่น ๆ ที่สื่อความหมายที่ดี
- ถ้าใช้  $i$  กับ  $j$  คนจำนวนมากจะหลงและมักนำไปสู่โปรแกรมที่ผิด (ซำร้ายยังหาเจอยากเพราะ  $i$  กับ  $j$  มันดูคล้ายกัน)
- การเขียนหรืออ่านค่าในแถวลำดับไม่จำเป็นต้องเรียงในทิศใดทิศหนึ่ง เราอยากอ้างถึงข้อมูลตรงไหนก็ได้ตามใจชอบทันที
  - การอ้างถึงข้อมูลจุดใดก็ได้ทันทีแบบนี้เรียกว่า การเข้าถึงแบบสุ่ม (random access)
  - ส่วนการอ้างถึงข้อมูลแบบที่ต้องผ่านตัวแรกก่อนที่จะค่อย ๆ ไล่ไปตัวที่สอง สาม สี่เรื่อย ๆ จะเรียกว่า การเข้าถึงโดยลำดับ (sequential access)