

ข้อมูลนิกอาร์เรย์, สตริง, พอยเตอร์

พร้อมแล้วส่งเสียงหน่อยจ้า.....



ข้อมูลนิกอาร์เรย์

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



ตัวแปรอาร์เรย์เก็บจำนวนเต็ม 7 จำนวน

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

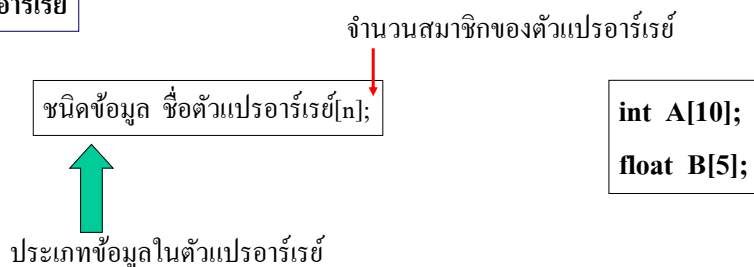


ตัวแปรอาร์เรย์เก็บทศนิยม 5 จำนวน

แถวลำดับ (Array)

ตัวแปรประเภทอาร์เรย์ เป็นตัวแปรที่สามารถเก็บข้อมูลหลายๆ ค่าไว้ในตัวแปรชื่อเดียวกันได้ โดยระบบจะใช้พื้นที่หน่วยความจำต่อเนื่องกัน เพื่อเก็บข้อมูลชนิดเดียวกันหลายจำนวน

การประกาศตัวแปรอาร์เรย์



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

ถ้าหากประกาศตัวแปรเป็นสตริง ตัวแปรนั้นก็คืออาร์เรย์ของ char นั่นเองครับ

การอ้างถึงสมาชิกในอาร์เรย์

ชื่อตัวแปรอาร์เรย์(ตรงชนิกำกับ)

เลขจำนวนเต็มตั้งแต่ 0 ถึง n-1

ตัวอย่าง

```
int n[5];
```

```
n[4] = 25;
```

สามารถอ้างสมาชิกแต่ละหน่วยของอาร์เรย์ n[] โดยใช้ n[0],n[1],n[2],n[3]

และ n[4]

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

ถ้าหากมีการอ้างถึงอาร์เรย์อาจเป็นดังต่อไปนี้

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


ขนาดหน่วยความจำของอาร์เรย์

ขึ้นอยู่กับประเภทของข้อมูลและจำนวนสมาชิกที่จองไว้

ตัวอย่าง

int x[20];

x[0] x[1] x[2] x[3] x[17] x[18] x[19]



จองเนื้อที่ช่องละ 2 ไบต์ จำนวน 20 ช่อง รวมเป็น 40 ไบต์

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);
}
```

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

```
#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");
    }
}
```

ตัวอย่าง

สมมติให้ i, j, k เป็นตัวแปรประเภท int และประกาศตัวแปร table เป็นอาร์เรย์เก็บจำนวนเต็ม

```
for (int k = 0; k < 9; k++)
    printf ("Value at %d = %d\n", k+1, table[k]);
table[i + j] = 0;
table[7 - table[j]] = j;
```

11



สิ่งที่ต้องระวัง

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

12

ตัวอย่าง 6.2

ให้อ่านค่าของจำนวนเต็ม 5 จำนวนจาก
คีย์บอร์ด 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



13

สมาชิกของอาร์เรย์อาจเป็นประเภทข้อมูลพื้นฐาน
ใด ๆ ก็ได้ หรืออาจเป็นข้อมูลประเภท Enumeration
เช่น

```
#define TSIZE      10
#define NAMESIZE   20
#define ADDRSIZE   30
enum month { JAN, FEB, MAR, APR, MAY,
             JUN, JUL, AUG, SEP, OCT,
             NOV, DEC }
```

14

```
typedef enum month Month;
int age[TSIZE];
float size[TSIZE+1];
Month date[8];
char name[NAMESIZE], address[ADDRSIZE];
```

15

แบบฝึกหัดเพิ่มเติม

1. จงเขียนฟังก์ชันตรวจสอบว่ากค y/Y หรือ n/N หรือไม่
2. เขียนฟังก์ชันรับข้อมูล 10 ค่าแล้วหาผลรวม
3. เขียนโปรแกรมหาค่า \sin , \cos , \tan โดยมีมุมเป็นองศา ตั้งแต่ 0 ถึง 90 โดยให้กระโดดครั้งละ 5
4. เขียนโปรแกรมหาค่า $f(x) = x^2 + 1$
5. เขียนโปรแกรมหาพื้นที่ใต้กราฟของ $y(x) = x^2 - 3x + 2$

แบบฝึกหัด

1. ถ้าหากมีข้อมูลอยู่ 10 ค่า จงเขียนโปรแกรมเรียงข้อมูลจากค่าน้อยที่สุดไปหาค่ามากที่สุด
2. จำนวนเฉพาะคือจำนวนเต็มบวก ซึ่งหารด้วยจำนวนอื่นไม่ลงตัว ยกเว้น 1 กับตัวมันเอง ตัวอย่างเช่น 7 เป็นจำนวนเฉพาะ แต่ 6 ไม่ใช่
จงสร้างตารางจำนวนเฉพาะ n จำนวนแรก
3. จงเขียนโปรแกรมแปลงเลขฐานสิบเป็นเลขฐานสอง

อาร์เรย์สองมิติ

การประกาศตัวแปรอาร์เรย์สองมิติจะใช้ดัชนี 2 ตัว เพื่อระบุจำนวนสมาชิกในแต่ละหลัก และแต่ละแถว ดังนี้

ชนิดข้อมูล ชื่อตัวแปรอาร์เรย์[Row][Column]

ตัวอย่างเช่น

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

จะมีสมาชิกทั้งหมด 6 ตัว (2 x 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} };
```

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] = 5;
```

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

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

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

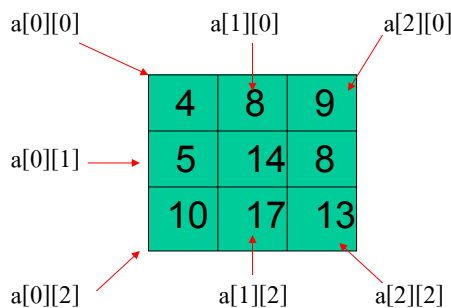
```
    a[1][2] = 17;
```

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

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

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

```
}
```



a[0][0]	4	a[1][0]	8	a[2][0]	9	
a[0][1]	5		14		8	
a[0][2]	10		a[1][2]	17	a[2][2]	13

การเก็บข้อมูลในอาร์เรย์สองมิติ

```
main()
```

```
{
```

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

```
    int x, y;
```

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

```
    {
```

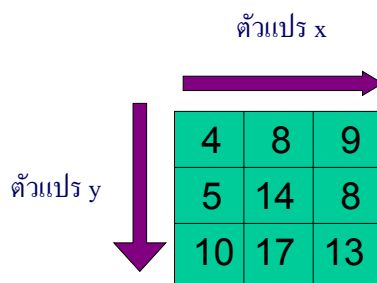
```
        for(x = 0; x <= 2; x++)
```

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

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

```
    }
```

```
}
```



	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

โปรแกรมอ่านข้อมูลจำนวนเต็มจากตัวแปรอาร์เรย์ 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++)
```

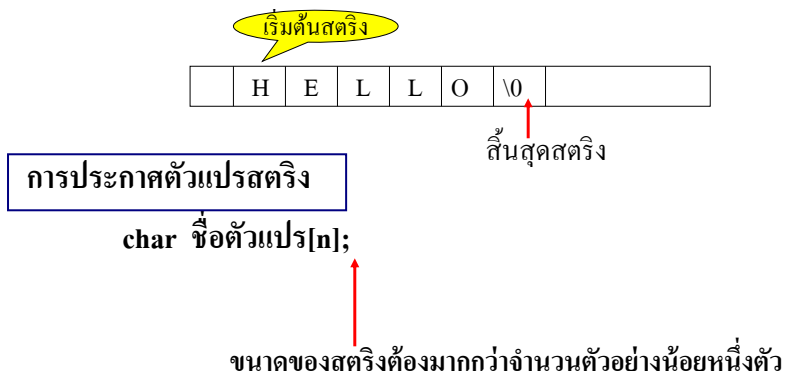
```
        s = s + a[i];
```

```
    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

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

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

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()

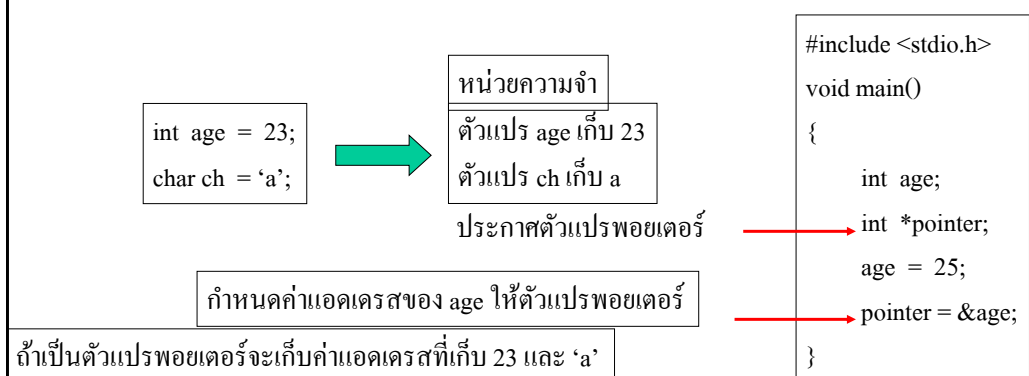
นับจำนวนอักขระในสตริง มีรูปแบบดังนี้

```
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);
}
```

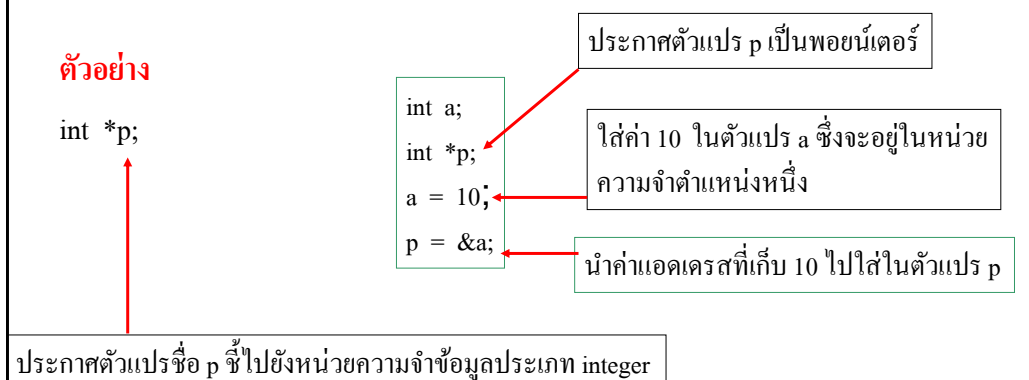
พอยเตอร์

พอยเตอร์(Pointer) หรือตัวชี้ ตัวแปรประเภทนี้จะไม่เก็บข้อมูล แต่จะเป็นค่าตำแหน่งหรือแอดเดรส(Address) ของข้อมูล จะพบมากในโปรแกรมประเภทโครงสร้างข้อมูลเช่น Stack, Queue, Linked list เป็นต้น



การประกาศตัวแปรพอยน์เตอร์

ประเภทของข้อมูล *ชื่อตัวแปร



การประกาศตัวแปรพอยน์เตอร์

int *pt_x;	สร้างตัวแปรพอยน์เตอร์ชนิด int ทำให้ pt_x ใช้เก็บตำแหน่งที่อยู่ของตัวแปรชนิด int เท่านั้น
float *pt_num;	สร้างตัวแปรพอยน์เตอร์ชนิด float ทำให้ pt_num ใช้เก็บตำแหน่งที่อยู่ของตัวแปรชนิด float เท่านั้น
char *pt_ch;	สร้างตัวแปรพอยน์เตอร์ชนิด char ทำให้ pt_ch ใช้เก็บตำแหน่งที่อยู่ของตัวแปรชนิด char เท่านั้น

ตัวอย่างแสดงค่าแอดเดรส

```
#include <stdio.h>
void main()
{
    int age;
    int *pointer;
    age = 25;
    pointer = &age;
    printf("Address = %p\n",pointer);
}
```



Address = 0065FDF4


```

main()
{
    int x; int *p;

    printf("Address = %p\n",p);

    p++;

    printf("Address = %p\n",p);

    x = 5;

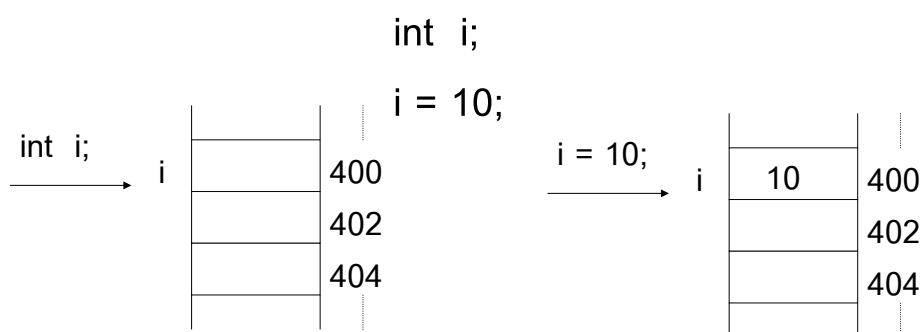
    p = &x;

    printf("Address = %p\n",p);

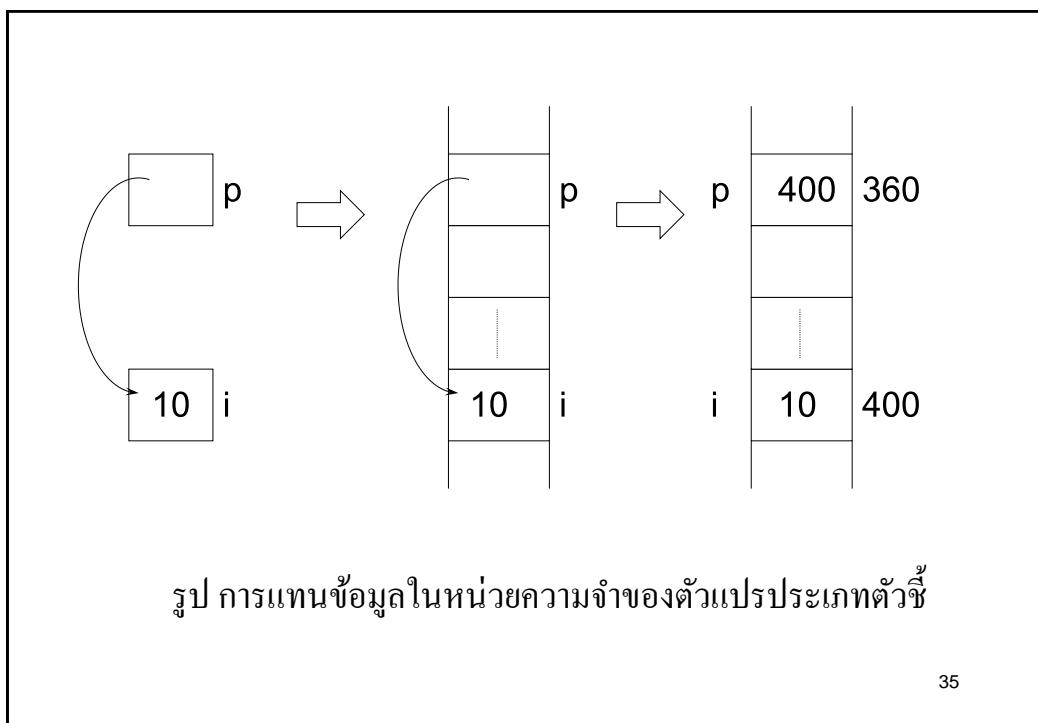
    printf("Data = %d\n",*p);
}

```

ตัวชี้กับแอดเดรส (Pointers and Address)



รูปการแทนข้อมูลในหน่วยความจำของตัวแปรประเภทพื้นฐาน



35

การประกาศตัวแปรประเภทตัวชี้

การประกาศตัวแปรประเภทพอยน์เตอร์จะใช้ Unary Operator * ซึ่งมีชื่อเรียกว่า Indirection หรือ Dereferencing Operator โดยจะต้องประกาศประเภทของตัวแปรพอยน์เตอร์ให้สอดคล้องกับประเภทของตัวแปรที่เราต้องการ (ยกเว้นตัวแปรพอยน์เตอร์ประเภท void ที่สามารถใช้ไปยังตัวแปรประเภทใดก็ได้)

36

ตัวอย่าง

```
int *ip;
```

เป็นการประกาศตัวแปร ip ให้เป็นตัวแปรพอยน์เตอร์ที่ชี้ไปยังตัวแปรประเภท int

```
double *dp, atof(char *);
```

เป็นการประกาศตัวแปร dp เป็นตัวแปรพอยน์เตอร์ที่ชี้ไปยังตัวแปรประเภท double และประกาศฟังก์ชัน atof มีพารามิเตอร์เป็นตัวแปรพอยน์เตอร์ประเภท char

37

การกำหนดค่าและการอ่านค่าตัวแปรประเภทตัวชี้

การกำหนดค่าให้กับตัวแปรพอยน์เตอร์จะเป็นการกำหนดแอดเดรสของตัวแปรที่มีประเภทสอดคล้องกับประเภทของตัวแปรพอยน์เตอร์เท่านั้น โดยใช้ Unary Operator **&** เป็นโอเปอเรเตอร์ที่อ้างถึงแอดเดรสของออบเจกต์ (Object) ใด ๆ

38

```
int x = 1, y = 2;
int *ip, *iq;
ip = &x;
y = *ip;
*ip = 0;
y = 5;
ip = &y;
*ip = 3;
iq = ip;
```

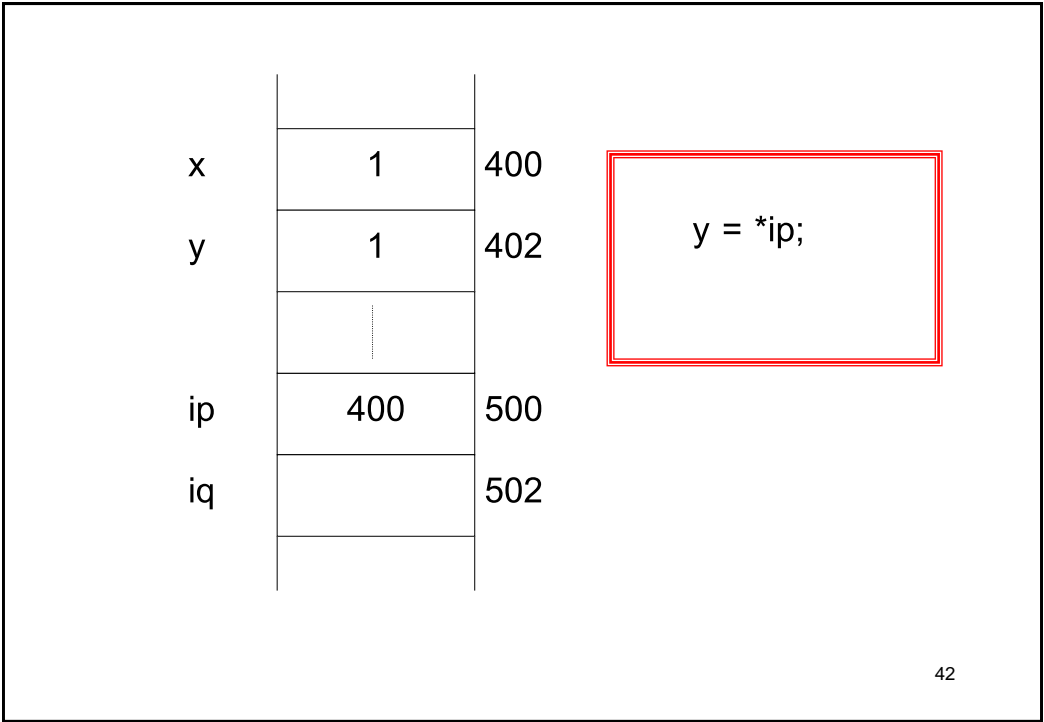
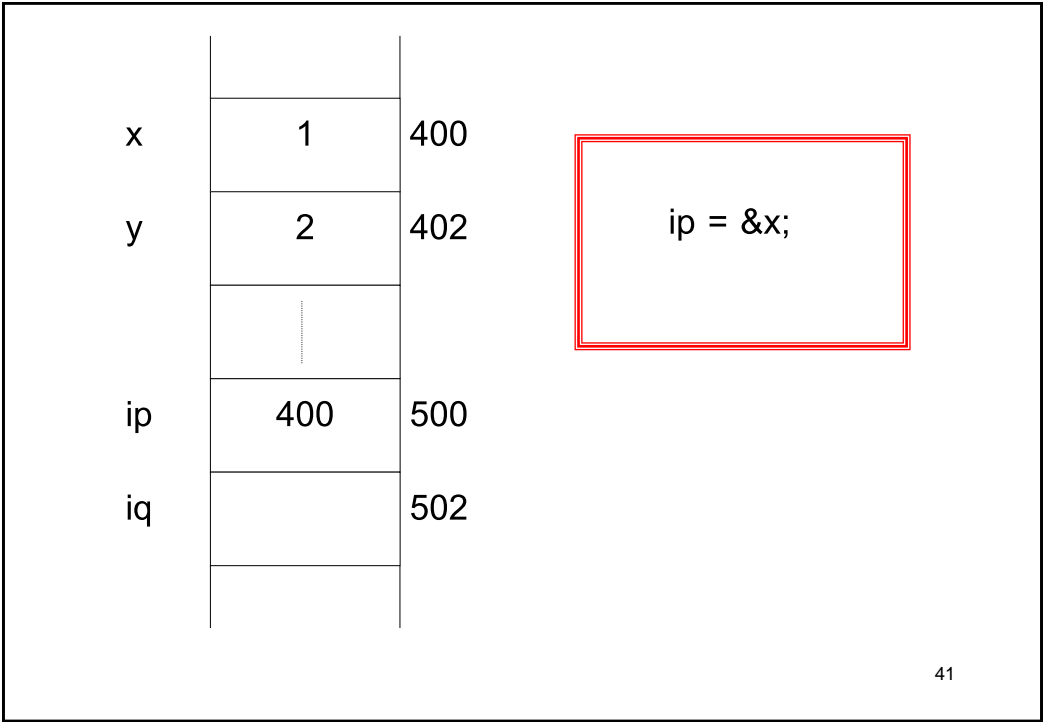
รูปการกำหนดค่าและการอ่านค่าตัวแปรตัวชี้

39

x	1	400
y	2	402
	⋮	
ip		500
iq		502

```
int x = 1, y = 2;
int *ip, *iq;
```

40



x	0	400
y	1	402
	⋮	
ip	400	500
iq		502

*ip = 0;

43

x	0	400
y	5	402
	⋮	
ip	400	500
iq		502

y = 5;

44

x	0	400
y	5	402
ip	402	500
iq		502

ip = &y;

45

x	0	400
y	3	402
	⋮	
ip	402	500
iq		502

*ip = 3;

46

x	0	400	
y	3	402	
	⋮		
ip	402	500	
iq	402	502	

iq = ip;

47

ตัวชี้และอาร์กิวเมนต์ของฟังก์ชัน (Pointer and Function Arguments)

เนื่องจากภาษาซีมีการส่งอาร์กิวเมนต์ให้กับฟังก์ชันแบบ By Value และฟังก์ชันสามารถคืนค่า (return) ค่าได้เพียงหนึ่งค่า หากต้องการให้ฟังก์ชันมีการเปลี่ยนแปลงค่าและคืนค่ากลับมายังฟังก์ชันที่เรียกใช้มากกว่าหนึ่งค่าจะต้องนำพอยน์เตอร์เข้ามาช่วย

ตัวอย่างเช่น หากต้องการเขียนฟังก์ชันเพื่อสลับค่าของตัวแปร 2 ตัว ผลลัพธ์ที่ต้องการได้จากฟังก์ชันนี้จะมี 2 ค่าของตัวแปรที่ทำการสลับค่า หากอาร์กิวเมนต์เป็นตัวแปรธรรมดาจะไม่สามารถแก้ปัญหานี้ได้ จึงต้องใช้พอยน์เตอร์เข้ามาช่วย โดยการส่งค่าแอดเดรสของตัวแปรทั้ง 2 ให้กับฟังก์ชันที่จะสลับค่าของตัวแปรทั้ง 2 ผ่านทางตัวแปรพอยน์เตอร์ที่เป็นอาร์กิวเมนต์ของฟังก์ชัน

49

ตัวอย่าง

โปรแกรมตัวอย่างการสลับค่าตัวแปร 2 ตัว
โดยผ่านฟังก์ชัน จะแสดงการส่ง
อาร์กิวเมนต์ในเป็นพอยน์เตอร์

```
#include <stdio.h>
void swap (int *, int *);
```

50

ตัวอย่าง (ต่อ)

```

void main ( )
{
    int x = 5, y = 10;
    printf("Before swap : x = %d", x, ", y = %d\n", y);
    swap ( &x, &y);
    printf("After swap : x = %d", x, ", y = %d\n", y);
}

```

51

ตัวอย่าง (ต่อ)

```

void swap (int *px, int *py)
{
    int temp;
    temp = *px;
    *px = *py;
    *py = temp;
}

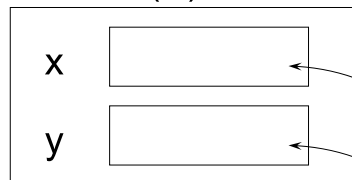
```

52

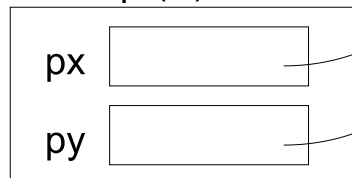
อาร์กิวเมนต์ที่เป็นประเภทพอยน์เตอร์จะช่วยให้ฟังก์ชันสามารถเปลี่ยนค่าให้กับตัวแปรที่ส่งเข้ามาได้ เนื่องจากอาร์กิวเมนต์นั้นจะเก็บแอดเดรสของตัวแปรที่ส่งเข้ามา เมื่อมีการเปลี่ยนแปลงค่าของอาร์กิวเมนต์ผ่าน Dereferencing Operator (*) ค่าของตัวแปรที่ส่งเข้ามาจะถูกเปลี่ยนค่าพร้อมกันในทันที

53

in main ()



in swap ()



รูปแสดงความสัมพันธ์ของการส่งอาร์กิวเมนต์แบบพอยน์เตอร์กับฟังก์ชัน

54

การใช้ตัวชี้กับอาร์เรย์

การทำงานใด ๆ ของอาร์เรย์สามารถใช้พอยน์เตอร์เข้ามาช่วย ซึ่งจะทำให้มีความเร็วในการทำงานสูงขึ้น สมมติว่ามีอาร์เรย์ a และพอยน์เตอร์ pa ดังนี้

```
int a[10];
```

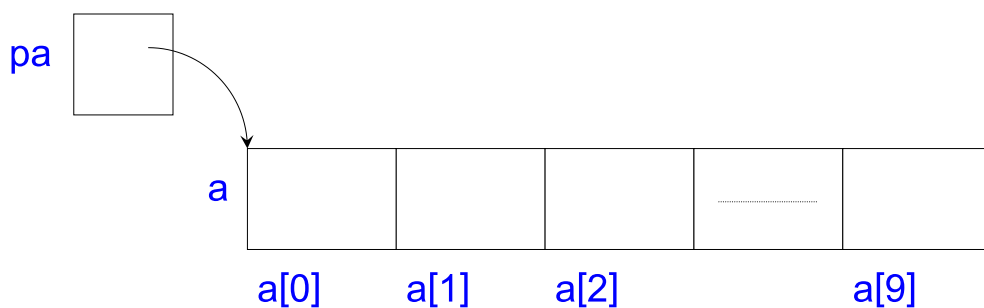
```
int *pa;
```

กำหนดให้พอยน์เตอร์ pa ชี้ไปยังอาร์เรย์ a ด้วยคำสั่ง

```
pa = &a[0]; /* หรือใช้คำสั่ง pa = a; */
```

pa จะเก็บค่าแอดเดรสเริ่มต้นของอาร์เรย์ a

55



รูป แสดงตัวชี้ชี้ไปยังแอดเดรสเริ่มต้นของอาร์เรย์

56

การนำไปใช้งานจะสามารถอ่านค่าอาร์เรย์ผ่าน
พอยน์เตอร์ได้ดังนี้

```
x = *pa;
```

จะเป็นการกำหนดค่าให้ x มีค่าเท่ากับ a[0] การเลื่อนไป
อ่านค่าสมาชิกตำแหน่งต่าง ๆ ของอาร์เรย์ผ่านทาง
พอยน์เตอร์สามารถทำได้โดยการเพิ่มค่าพอยน์เตอร์ขึ้น
1 เพื่อเลื่อนไปยังตำแหน่งถัดไป หรือเพิ่มค่าขึ้น N เพื่อ
เลื่อนไป N ตำแหน่ง หรืออาจจะลดค่าเพื่อเลื่อน
ตำแหน่งลง

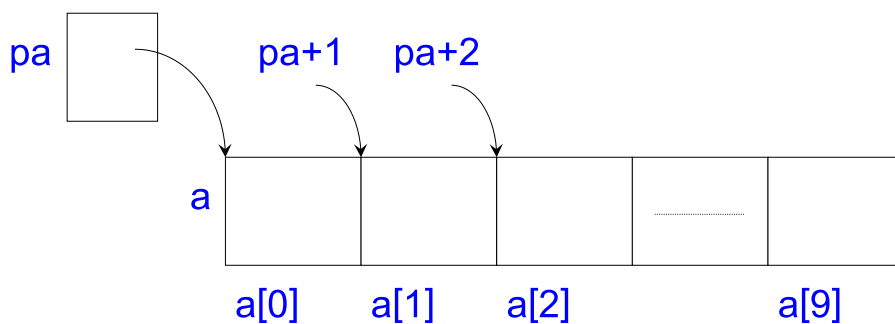
57

กรณีที่ pa ชี้อู่ที่ a[0] คำสั่ง

```
pa+1;
```

จะเป็นการอ้างถึงแอดเดรสของ a[1] หากเป็น pa+i
เป็นการอ้างถึงแอดเดรส a[i] หากต้องการอ้างถึง
ข้อมูลภายในของสมาชิกของอาร์เรย์ตำแหน่งที่ a[i]
จะใช้ *(pa+i)

58



รูป แสดงการอ้างถึงตำแหน่งในอาร์เรย์ผ่านตัวชี้

59

การสั่งให้บวก 1 หรือบวก i หรือ ลบ i เป็นเหมือนการ
 เลื่อนไปยังสมาชิกของอาร์เรย์ตำแหน่งที่ต้องการ เนื่องจาก
 ประเภทของข้อมูลแต่ละประเภทของอาร์เรย์ เช่น `int`, `float`,
`double` และอื่น ๆ มีขนาดของข้อมูลที่แตกต่างกัน
 ทำให้ขนาดของสมาชิกภายในอาร์เรย์แต่ละประเภทมีขนาด
 แตกต่างกันด้วย การสั่งให้บวกหรือลบด้วยจำนวนที่ต้องการนั้น
 จะมีกลไกที่ทำหน้าที่คำนวณตำแหน่งที่ต้องการให้สอดคล้อง กับ
 ข้อมูลแต่ละประเภทโดยอัตโนมัติ

60

นอกจากนี้ยังสามารถใช้พอยน์เตอร์แทนอาร์เรย์ การอ้างโดยใช้ `a[i]` สามารถใช้ `*(a+i)` เนื่องจากทุกครั้งที่อ้างถึง `a[i]` ภาษาซีจะทำหน้าที่แปลงเป็น `*(a+i)` เพราะฉะนั้นการเขียนในรูปแบบใดก็ให้ผลลัพธ์ในการทำงานเช่นเดียวกัน และการอ้างถึงแอดเดรส เช่น `&a[i]` จะมีผลเท่ากับการใช้ `a+i`

61

ในลักษณะเดียวกันการใช้งานพอยน์เตอร์ก็สามารถใช้คำสั่งในลักษณะอาร์เรย์ก็ได้ เช่น การอ้างถึง `*(pa+i)` สามารถเขียนด้วย `pa[i]` ก็ได้ผลเช่นเดียวกัน

สิ่งที่แตกต่างกันของอาร์เรย์และพอยน์เตอร์ คือ พอยน์เตอร์เป็นตัวแปร แต่อาร์เรย์ไม่ใช่ตัวแปร สมมติให้ `a` เป็นอาร์เรย์และ `pa` เป็นพอยน์เตอร์ การอ้างถึง `pa = a` หรือ `pa++` จะสามารถคอมไพล์ได้ แต่จะไม่สามารถใช้คำสั่ง `a = pa` หรือ `a++` ได้

62

เมื่อมีการส่งชื่อของอาร์เรย์ให้แก่ฟังก์ชัน
จะเป็นการส่งตำแหน่งแอดเดรสของสมาชิกตัวแรกของ
อาร์เรย์ให้แก่ฟังก์ชัน ดังนั้นพารามิเตอร์ในฟังก์ชันนั้น
จะเป็นตัวแปรประเภทพอยน์เตอร์

63

ตัวอย่าง

ฟังก์ชันที่รับพารามิเตอร์เป็นพอยน์เตอร์ โดย
อาร์กิวเมนต์ที่ส่งมาเป็นอาร์เรย์

```
int strlen (char *s)
{
    int n;
    for ( n = 0; *s != '\0'; s++ )
        n++;
    return n;
}
```

64

จะเห็นว่า s เป็นพอยน์เตอร์ ในฟังก์ชันจะมีการตรวจสอบข้อมูลว่ามีค่าเท่ากับ '\0' หรือไม่ และมีการเลื่อนตำแหน่งทีละ 1 ค่า (นับว่าข้อมูลมีความยาวเพิ่มขึ้นทีละ 1) โดยใช้ s++ การเรียกใช้ฟังก์ชัน strlen สามารถทำได้หลายลักษณะ

```
strlen ("hello world");    /* string constant */
strlen (array);            /* char array[10] */
strlen (ptr);              /* char *ptr;    */
```

65

นอกจากนี้ยังอาจจะประกาศพารามิเตอร์ภายในฟังก์ชัน strlen ได้ใน **2** ลักษณะ คือ char *s แบบในตัวอย่าง หรืออาจจะใช้ char s[] ก็ได้ โดยทั่วไปจะใช้ในลักษณะแรก เพราะช่วยในรู้ได้ทันทีว่า s เป็นตัวแปรพอยน์เตอร์ และยังสามารถส่งส่วนใดส่วนของอาร์เรย์ให้แก่ฟังก์ชันก็ได้ โดยไม่จำเป็นต้องส่งสมาชิกตัวแรกก็ได้เช่นกัน

66

ตัวอย่าง

`f (&a[2])`

หรือ `f (a+2)`

เป็นการส่งแอดเดรสของสมาชิก `a[2]` ให้กับฟังก์ชัน `f` การประกาศฟังก์ชัน `f` สามารถทำได้โดยการประกาศ

`f (int arr[]) { }`

หรือ `f (int *arr) { }`

67

การคำนวณกับแอดเดรส

ให้ `p` เป็นพอยน์เตอร์ชี้ไปยังอาร์เรย์ใด ๆ คำสั่ง `p++` เป็นการเลื่อน `p` ไปยังสมาชิกถัดไป และคำสั่ง `p += i` เป็นการเลื่อนพอยน์เตอร์ไป `i` ตำแหน่งจากตำแหน่งปัจจุบัน นอกจากนี้ยังสามารถใช้เครื่องหมายความสัมพันธ์ (Relational Operator) เช่น `==`, `!=`, `<`, `>=` และอื่น ๆ ทำงานร่วมกับพอยน์เตอร์ได้ สมมติให้ `p` และ `q` ชี้ไปยังสมาชิกของอาร์เรย์เดียวกัน

68

$$p < q$$

จะเป็นจริงเมื่อ p ชี้ไปที่สมาชิกที่อยู่ก่อนหน้าสมาชิกที่ q ชี้อยู่ การเปรียบเทียบในลักษณะจะใช้ได้ต่อเมื่อ p และ q ชี้ไปที่อาร์เรย์เดียวกันเท่านั้น

นอกจากนี้ยังสามารถใช้การลบหรือการบวกกับพอยน์เตอร์ได้เช่นเดียวกัน แต่สิ่งที่ควรระวังคือ การทำเช่นนั้นจะต้องอยู่ในขอบเขตขนาดของอาร์เรย์เท่านั้น

69

ตัวอย่าง

ฟังก์ชัน `strlen()` ปรับปรุงให้กระชับขึ้น

```
int strlen (char *s)
{
    char *p = s;
    while (*p != '\0')
        p++;
    return p-s;
}
```

70

เนื่องจาก s ชี้อยู่ที่ตำแหน่งเริ่มต้น โดยมี p ชีไปที่ s เช่นเดียวกัน แต่จะมีการเลื่อน p ไปทีละหนึ่งตำแหน่ง จนกว่าค่าที่ตำแหน่งที่ p ชี้อยู่จะเท่ากับ '\0' เมื่อนำ p ค่าสุดท้ายมาลบกับ s ที่ตำแหน่งเริ่มต้น ก็จะได้ความยาวของข้อมูลที่ส่งเข้ามา

71

ตัวชี้ตัวอักษรและฟังก์ชัน

(Character Pointer and Function)

การทำงานกับข้อความหรือที่เรียกว่า สตริง (String) เป็นการใชข้อมูลตัวอักษรหลาย ๆ ตัว หรืออาร์เรย์ของข้อมูลประเภท char หรืออาจจะใช้พอยน์เตอร์ชี้ไปยังข้อมูลประเภท char การทำงานกับ ค่าคงที่สตริง (String Constant) สามารถเขียนภายในเครื่องหมาย “ ”

72

ตัวอย่าง

“I am a string”

เมื่อมีการใช้ค่าคงที่สตริงจะมีการพื้นที่ในหน่วยความจำ เท่ากับความยาวของค่าคงที่สตริงบวกด้วย 1 เนื่องจาก ลักษณะการเก็บข้อมูลประเภทข้อความในหน่วยความจำจะมีการปะตัวอักษร null หรือ '\0' ต่อท้ายเสมอเพื่อให้รู้ว่าเป็น จุดสิ้นสุดของข้อมูล การจองพื้นที่ดังกล่าวจะเหมือนการจอง พื้นที่ของข้อมูลประเภทอาร์เรย์ เป็นอาร์เรย์ของ char

73

I		a	m		a		s	t	r	i	n	g	\0
---	--	---	---	--	---	--	---	---	---	---	---	---	----

รูป แสดงแบบจำลองการเก็บข้อมูลประเภท สตริงใน
หน่วยความจำ

74

ค่าคงที่สตริงที่พบเห็นได้เสมอได้แก่ข้อความที่ใช้
ในฟังก์ชัน printf () เช่น

```
printf ( "Hello, world\n" );
```

ฟังก์ชัน printf () จะรับพารามิเตอร์เป็นพอยน์เตอร์ชี้ไปยังแอดเดรสของข้อมูลที่ตำแหน่งเริ่มต้นของอาร์เรย์ และนำข้อความนั้นแสดงออกทางอุปกรณ์แสดงข้อมูลมาตรฐาน

75

ในการเขียนโปรแกรมจะสามารถใช้พอยน์เตอร์ชี้ไปค่าคงที่สตริงใด ๆ ก็ได้ เช่น

```
char *pmessage = "Hello, world";
```

pmessage จะเป็นพอยน์เตอร์ประเภท char ชี้ไปที่อาร์เรย์ของตัวอักษร จะแตกต่างจากการใช้อาร์เรย์ทั่วไปเช่น

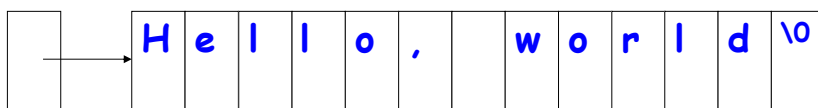
```
char amessage[ ] = "Hello, world";
```

76

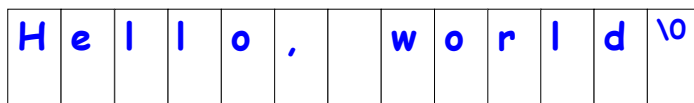
ลักษณะของอาร์เรย์เช่น amessage จะมีการจองพื้นที่ให้กับอาร์เรย์ขนาด 13 ตัวอักษรรวมทั้ง null ส่วนลักษณะของพอยน์เตอร์ที่ชี้ไปยังค่าคงที่สตริง จะมีการจองพื้นที่ให้กับค่าคงที่สตริงขนาด 13 ตัวอักษรเช่นเดียวกัน แต่จะมีการจองพื้นที่ให้กับพอยน์เตอร์ และทำการชี้พอยน์เตอร์นั้นไปยังพื้นที่ของค่าคงที่สตริงที่จองเอาไว้

77

pmessage



amessage



รูป การจองพื้นที่ให้กับอาร์เรย์และตัวชี้ชี้ไปยังค่าคงที่สตริง

78

ตัวอย่าง

ฟังก์ชัน strcpy () ทำหน้าที่สำเนา
ข้อความจากตัวแปรหนึ่งไปยังอีกตัวแปร
หนึ่งเขียนในลักษณะอาร์เรย์

```
void strcpy ( char *s, char *t )
{
    int i=0;
    while ( ( s[i] = t[i] ) != '\0' )
        i++;
}
```

79

ตัวอย่าง

ฟังก์ชัน strcpy () เขียนในลักษณะ
พอยน์เตอร์

```
void strcpy ( char *s, char *t )
{
    while ( ( *s = *t ) != '\0' ) {
        s++;
        t++;
    }
}
```

80

ตัวอย่าง

ฟังก์ชัน strcpy () เขียนในลักษณะ
พอยน์เตอร์แบบสั้น

```
void strcpy ( char *s, char *t )  
{  
    while ( ( *s++ = *t++ ) != '\0' ) ;  
}
```