

Computer Programming II การเขียนโปรแกรมคอมพิวเตอร์2 LECTURE#8 พอยน์เตอร์ (Pointer)

อ.สฤติย์ ประสมพันธ์
ภาควิชาวิทยาการคอมพิวเตอร์และสารสนเทศ
KMUTNB

พอยน์เตอร์ หรือ ตัวชี้

- เป็นชนิดข้อมูลชนิดหนึ่งของภาษา C (int, float, char)
- ตัวแปรชนิดตัวชี้คืออะไร?
 - มีความเร็วในการทำงานสูง
 - ช่วยประหยัดเนื้อที่ในหน่วยความจำหลักขณะประมวลผลเมื่อเทียบกับ Array
 - ใช้ตัวชี้ร่วมกับฟังก์ชันเพื่อเพิ่มประสิทธิภาพการเขียนโปรแกรม

พอยน์เตอร์กับแอดเดรส (Pointers and Addresses)

- ตัวแปร
 - คือชื่อที่ใช้แทนข้อมูล
 - การประกาศตัวแปรเป็นการกำหนดชื่อเพื่อใช้แทนข้อมูล
 - เมื่อประกาศตัวแปร จะมีการจองเนื้อที่ในหน่วยความจำเพื่อเก็บข้อมูล
 - เราสามารถเข้าถึงข้อมูลได้โดยอ้างถึงตัวแปร
- การประกาศตัวแปร เช่น

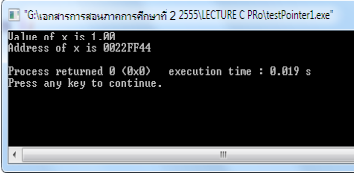
```
int i;
```

 - เป็นการประกาศ (Declaration) ตัวแปรชื่อ i
 - เป็นตัวแปรชนิด int (integer)

พอยน์เตอร์กับแอดเดรส (Pointers and Addresses)

```
#include <stdio.h>
#include <stdlib.h>
int main()
{
    float x = 1.0;
    printf("Value of x is %.2f\n", x);
    printf("Address of x is %p\n", &x);

    return 0;
}
```



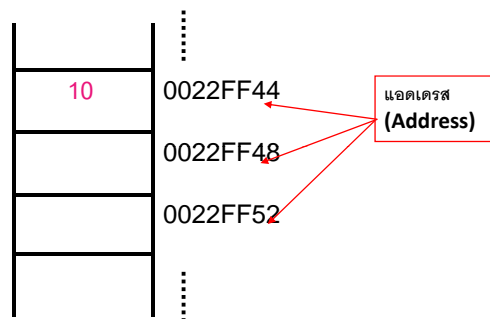
x 1.00
0022FF44

ภาพจำลองการแทนข้อมูลในหน่วยความจำแบบปกติ

```
int i;
```

i

```
i = 10;
```



แอดเดรส คือ ตำแหน่งที่เก็บข้อมูลในหน่วยความจำ

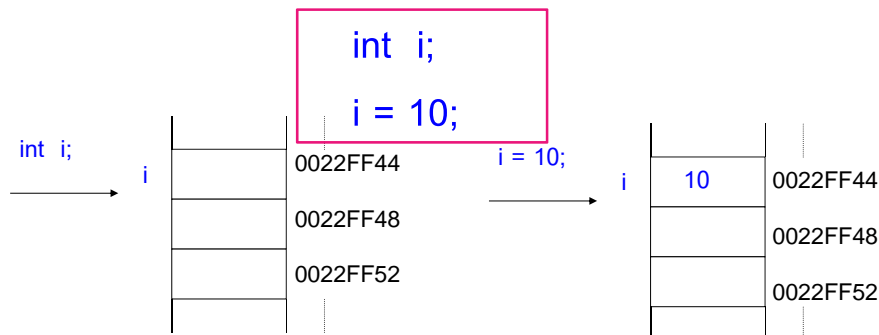
5

ตัวชี้กับแอดเดรส (Pointers and Address)

- การใช้ตัวชี้หรือพอยน์เตอร์เป็นอีกวิธีที่จะเข้าถึงตัวแปรปกติได้
- ตัวแปรชนิดพอยน์เตอร์จะเก็บค่าที่อยู่ของหน่วยความจำหลัก ซึ่งต่างกับตัวแปรปกติที่เก็บค่าที่แท้จริงของข้อมูล
- การใช้ตัวแปรชนิดพอยน์เตอร์จะเป็นการเข้าถึงข้อมูลหรือเป็นการอ้างถึงตำแหน่งที่เก็บข้อมูล

6

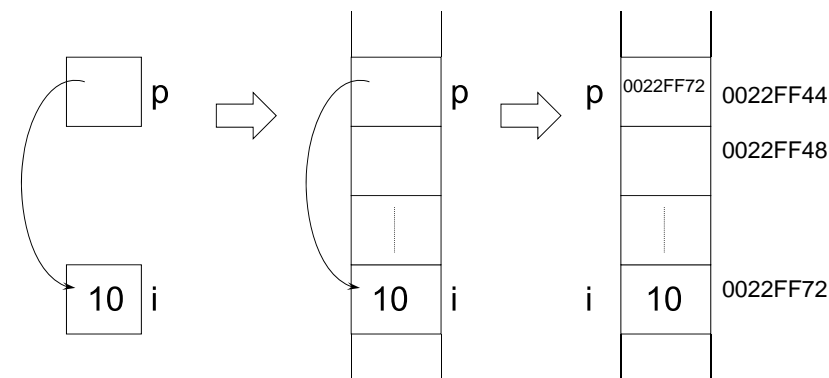
ตัวชี้กับแอดเดรส (Pointers and Address)



การแทนข้อมูลในหน่วยความจำของตัวแปรประเภทพื้นฐาน

7

ตัวชี้กับแอดเดรส (Pointers and Address)



```
int *p;
p = &i;
```

การแทนข้อมูลในหน่วยความจำของตัวแปรประเภทตัวชี้

8

การประกาศตัวแปรประเภทตัวชี้

- การประกาศตัวแปรประเภทพอยน์เตอร์จะใช้ * ซึ่งมีชื่อเรียกว่า Indirection หรือ Dereferencing Operator
- ต้องประกาศประเภทของตัวแปรพอยน์เตอร์ให้สอดคล้องกับประเภทของตัวแปรที่เราต้องการ
- ตัวแปรพอยน์เตอร์ประเภท void สามารถชี้ไปยังตัวแปรประเภทใดก็ได้

9

การประกาศตัวแปรประเภทตัวชี้

```
int *ip;
```

- ประกาศตัวแปร ip ให้เป็นตัวแปรพอยน์เตอร์ที่ชี้ไปยังตัวแปรประเภท int

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

- ประกาศตัวแปร dp เป็นตัวแปรพอยน์เตอร์ที่ชี้ไปยังตัวแปรประเภท double
- ประกาศฟังก์ชัน atof มีพารามิเตอร์เป็นตัวแปรพอยน์เตอร์ประเภท char

10

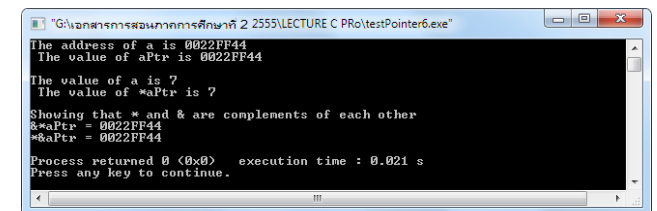
การกำหนดค่าและการอ่านค่าตัวแปรประเภทตัวชี้

- การกำหนดค่าให้กับตัวแปรพอยน์เตอร์จะเป็นการกำหนดแอดเดรสของตัวแปรที่มีประเภทสอดคล้องกับประเภทของตัวแปรพอยน์เตอร์
- ใช้ & เป็นโอเปอเรเตอร์ที่อ้างถึงแอดเดรสของตัวแปรใดๆ

11

ตัวชี้กับแอดเดรส (Pointers and Address)

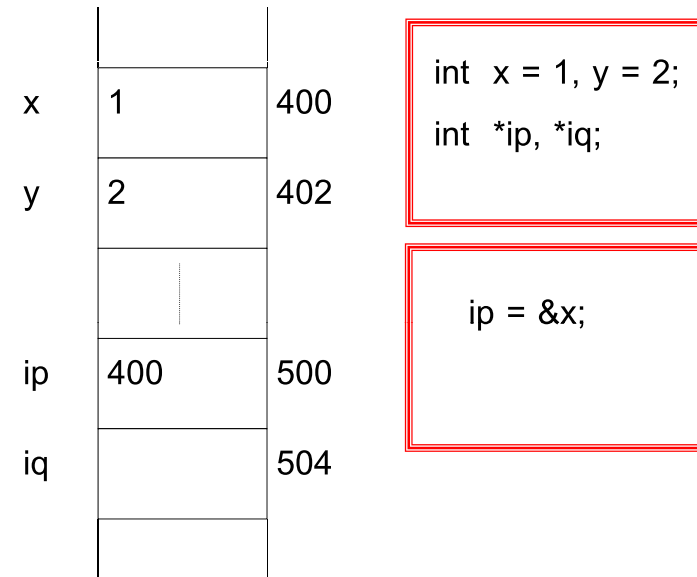
```
#include <stdio.h>
int main( void )
{
    int a; /* a is an integer */
    a = 7;
    int *aPtr; /* aPtr is a pointer to an integer */
    aPtr = &a; /* aPtr set to address of a */
    printf( "The address of a is %p\n The value of aPtr is %p", &a, aPtr );
    printf( "\n\nThe value of a is %d\n The value of *aPtr is %d", a, *aPtr );
    printf( "\n\nShowing that * and & are complements of each other\n&*aPtr = %p\n*&aPtr = %p\n", &*aPtr, *&aPtr );
    return 0; /* indicates successful termination */
} /* end main */
```



การกำหนดค่าและการอ่านค่าตัวแปรตัวชี้

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

13



x	1	400
y	1	402
ip	400	500
iq		504

y = *ip;

x	0	400
y	1	402
ip	400	500
iq		504

*ip = 0;

x	0	400
y	5	402
ip	400	500
iq		504

y = 5;

x	0	400
y	5	402
ip	402	500
iq		504

ip = &y;

x	0	400
y	3	402
ip	402	500
iq		504

*ip = 3;

x	0	400
y	3	402
ip	402	500
iq	402	504

iq = ip;

ตัวชี้และอาร์กิวเมนต์ของฟังก์ชัน(Pointer and Function Arguments)

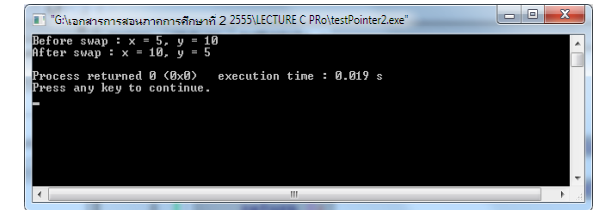
- เนื่องจากภาษาซีมีการส่งอาร์กิวเมนต์ให้กับฟังก์ชันแบบ By Value และฟังก์ชันสามารถคืนค่า (return) ค่าได้เพียงหนึ่งค่า
- หากต้องการให้ฟังก์ชันมีการเปลี่ยนแปลงค่าและคืนค่ากลับมายังฟังก์ชันที่เรียกใช้มากกว่าหนึ่งค่าจะต้องนำพอยน์เตอร์เข้ามาช่วย

ตัวอย่าง

โปรแกรมตัวอย่างการสลับค่าตัวแปร 2 ตัวโดยผ่านฟังก์ชัน จะแสดงการส่งอาร์กิวเมนต์ให้เป็นพอยน์เตอร์

```
#include<stdio.h>
void swap (int *, int *);
int main ( ) {
    int x = 5, y = 10;
    printf("Before swap : x = %d, y = %d\n",x, y);
    swap (&x, &y);
    printf("After swap : x = %d, y = %d\n",x, y);
    return 0;
}
void swap (int *px, int *py)
{
    int temp;
    temp = *px;
    *px = *py;
    *py = temp;
}
```

หากอาร์กิวเมนต์เป็นตัวแปรธรรมดาจะไม่สามารถแก้ปัญหาได้ จึงต้องใช้พอยน์เตอร์เข้ามาช่วย โดยการส่งค่าแอดเดรสของตัวแปรทั้ง 2 ให้กับฟังก์ชันที่จะสลับค่าของตัวแปรทั้ง 2 ผ่านทางตัวแปรพอยน์เตอร์ที่เป็นอาร์กิวเมนต์ของฟังก์ชัน

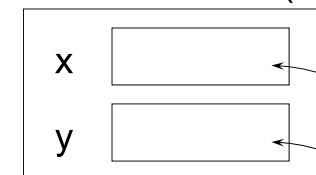


อาร์กิวเมนต์ที่เป็นประเภทพอยน์เตอร์

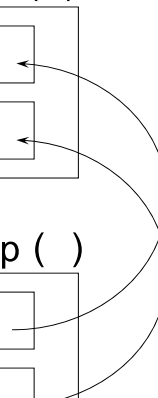
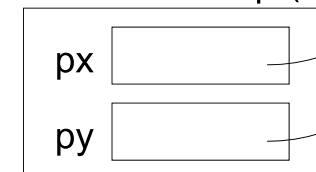
- อาร์กิวเมนต์ที่เป็นประเภทพอยน์เตอร์จะช่วยให้ฟังก์ชันสามารถเปลี่ยนค่าให้กับตัวแปรที่ส่งเข้ามาได้ เนื่องจากอาร์กิวเมนต์นั้นจะเก็บแอดเดรสของตัวแปรที่ส่งเข้ามา เมื่อมีการเปลี่ยนแปลงค่าของอาร์กิวเมนต์ผ่าน Dereferencing Operator (*) ค่าของตัวแปรที่ส่งเข้ามาจะถูกเปลี่ยนค่าพร้อมกันในทันที

ความสัมพันธ์ของการส่งอาร์กิวเมนต์แบบพอยน์เตอร์กับฟังก์ชัน

ตัวแปรใน main ()



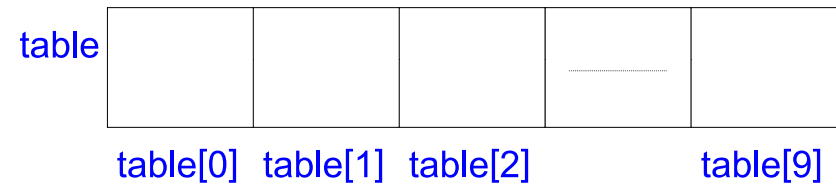
ตัวแปรใน swap ()



ตัวชี้กับอาร์เรย์ (Pointer and Arrays)

- อาร์เรย์เป็นประเภทข้อมูลที่เก็บชุดของข้อมูลประเภทเดียวกัน หรืออาร์เรย์เป็นโครงสร้างแบบ homogeneous ที่ประกอบด้วยอีลีเมนต์ (elements) ที่มีชนิด (type) เดียวกัน
- มักใช้กับการทำงานที่ต้องทำงานกับตัวแปรชนิดเดียวกันหลายตัวที่มีการทำงานเหมือนกัน เช่น คะแนนของนักศึกษาภายในห้อง 20 คน เป็นต้น อาร์เรย์ในภาษาซีจะนำหลักการของพอยน์เตอร์เข้ามาใช้
- การทำงานใด ๆ ของอาร์เรย์สามารถใช้พอยน์เตอร์เข้ามาแทนที่

แสดงภาพจำลองของอาร์เรย์ขนาดสมาชิก 10 ตัว

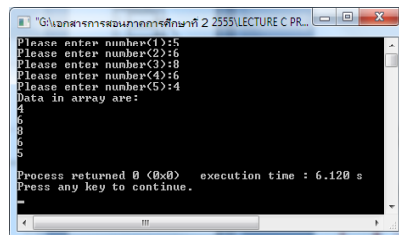


- สามารถอ้างถึงสมาชิกทุกตัวภายในอาร์เรย์ภายในขอบเขตของขนาดที่ได้ประกาศอาร์เรย์ไว้
- แต่การใช้อาร์เรย์มักจะเป็นการเข้าถึงสมาชิกในลักษณะทั่วไปโดยใช้ตัวแปรประเภท int มาช่วย

การอ้างถึงสมาชิกในอาร์เรย์

- ตัวอย่าง อ่านค่าของจำนวนเต็ม 5 จำนวนจากคีย์บอร์ด และแสดงผลในลำดับที่กลับกัน

```
#include <stdio.h>
#define SIZE 5
int main ( )
{
    int k;
    int table[SIZE];
    for (k = 0; k < SIZE; k++){
        printf("Please enter number(%d):", k+1);
        scanf ("%d", &table[k]);
    }
    printf("Data in array are:\n");
    for (k = SIZE-1; k >= 0; k--){
        printf ("%d\n", table[k]);
    }
    return 0;
}
```



การใช้ตัวชี้กับอาร์เรย์

- การทำงานใด ๆ ของอาร์เรย์สามารถใช้พอยน์เตอร์เข้ามาช่วย ซึ่งจะทำให้มีความเร็วในการทำงานสูงขึ้น
- สมมติว่ามีอาร์เรย์ a และพอยน์เตอร์ pa ดังนี้
- int a[10];
- int *pa;
- กำหนดให้พอยน์เตอร์ pa ชี้ไปยังอาร์เรย์ a ด้วยคำสั่ง



แสดงตัวชี้ชี้ไปยังแอดเดรสเริ่มต้นของอาร์เรย์

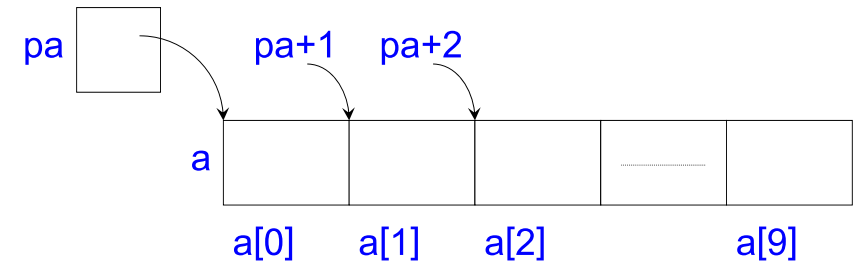
- pa = &a[0]; /* หรือใช้คำสั่ง pa = a; */
- pa จะเก็บค่าแอดเดรสเริ่มต้นของอาร์เรย์ a

การใช้ตัวชี้กับอาร์เรย์

- จะสามารถอ่านค่าอาร์เรย์ผ่านพอยน์เตอร์ได้ดังนี้
 - $x = *pa;$
- จะเป็นการกำหนดค่าให้ x มีค่าเท่ากับ $a[0]$
- การเลื่อนไปอ่านค่าสมาชิกตำแหน่งต่าง ๆ ของอาร์เรย์ผ่านทางพอยน์เตอร์สามารถทำได้โดยการเพิ่มค่าพอยน์เตอร์ขึ้น 1 เพื่อเลื่อนไปยังตำแหน่งถัดไป หรือเพิ่มค่าขึ้น N เพื่อเลื่อนไป N ตำแหน่ง หรืออาจจะลดค่าเพื่อเลื่อนตำแหน่งลง
- กรณีที่ pa ชี้อยู่ที่ $a[0]$ คำสั่ง $pa+1$; จะเป็นการอ้างถึงแอดเดรสของ $a[1]$
- หากเป็น $pa+i$ เป็นการอ้างถึงแอดเดรส $a[i]$
- หากต้องการอ้างถึงข้อมูลภายในของสมาชิกของอาร์เรย์ตำแหน่งที่ $a[i]$ จะใช้ $*(pa+i)$

การใช้ตัวชี้กับอาร์เรย์

การอ้างถึงตำแหน่งในอาร์เรย์ผ่านตัวชี้



- สามารถใช้พอยน์เตอร์แทนอาร์เรย์ การอ้างโดยใช้ $a[i]$ สามารถใช้ $*(a+i)$
- การอ้างถึงแอดเดรส เช่น $\&a[i]$ จะมีผลเท่ากับการใช้ $a+i$
- การอ้างถึง $*(pa+i)$ สามารถเขียนด้วย $pa[i]$

การอ้างถึงตำแหน่งในอาร์เรย์ผ่านตัวชี้

- สิ่งที่แตกต่างกันของอาร์เรย์และพอยน์เตอร์ คือ พอยน์เตอร์เป็นตัวแปร แต่อาร์เรย์ไม่ใช่ตัวแปร
- สมมติให้ a เป็นอาร์เรย์ และ pa เป็นพอยน์เตอร์
 - การอ้างถึง $pa = a$ หรือ $pa++$ จะสามารถคอมไพล์ได้
 - แต่จะไม่สามารถใช้คำสั่ง $a = pa$ หรือ $a++$ ได้

การอ้างถึงตำแหน่งในอาร์เรย์ผ่านตัวชี้

- เมื่อมีการส่งชื่อของอาร์เรย์ให้แก่ฟังก์ชัน จะเป็นการส่งตำแหน่งแอดเดรสของสมาชิกตัวแรกของอาร์เรย์ให้แก่ฟังก์ชัน ดังนั้นพารามิเตอร์ในฟังก์ชันนั้นจะเป็นตัวแปรประเภทพอยน์เตอร์

```
#include <stdio.h>
int main ( )
{
    char str1[10];
    scanf("%s",str1);
    printf("String length is
%d\n",strlen(str1));

    return 0;
}

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

อาจจะประกาศพารามิเตอร์ภายในฟังก์ชัน strlen ได้ใน 2 ลักษณะ คือ `char *s` หรือ `char s[]` ก็ได้

โดยทั่วไปจะใช้ในลักษณะแรก เพราะช่วยในรู้ได้ทันทีว่า s เป็นตัวแปรพอยน์เตอร์

สามารถส่งส่วนใดส่วนหนึ่งของอาร์เรย์ให้แก่ฟังก์ชันก็ได้ โดยไม่จำเป็นต้องส่งสมาชิกตัวแรกก็ได้เช่นกัน `strlen (&str1[2])`

เลขคณิตของ Pointer กับ Array

- ให้ p เป็นพอยน์เตอร์ชี้ไปยังอาร์เรย์ใด ๆ
 - คำสั่ง p++ เป็นการเลื่อน p ไปยังสมาชิกถัดไป และ
 - คำสั่ง p += i เป็นการเลื่อนพอยน์เตอร์ไป i ตำแหน่งจากตำแหน่งปัจจุบัน
 - สามารถใช้เครื่องหมายสัมพันธ์(Relational Operator) เช่น ==, !=, <, >= และอื่น ๆ ทำงานร่วมกับพอยน์เตอร์ได้
-

เครื่องหมายความสัมพันธ์ (Relational Operator)

- สมมติให้ p และ q ชี้ไปยังสมาชิกของอาร์เรย์เดียวกัน
 $p < q$
 - จะเป็นจริงเมื่อ p ชี้ไปที่สมาชิกที่อยู่ก่อนหน้าสมาชิกที่ q ชี้อยู่
 - การเปรียบเทียบในลักษณะจะใช้ได้ต่อเมื่อ p และ q ชี้ไปที่อาร์เรย์เดียวกันเท่านั้น
 - นอกจากนี้ยังสามารถใช้การลบหรือการบวกกับพอยน์เตอร์ได้ เช่นเดียวกัน แต่สิ่งที่ควรระวังคือ การทำเช่นนั้นจะต้องอยู่ในขอบเขตขนาดของอาร์เรย์เท่านั้น
-

ฟังก์ชัน strlen() ปรับปรุงให้กระชับขึ้น

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

- เนื่องจาก s ชี้อยู่ที่ตำแหน่งเริ่มต้น โดยมี p ชี้ไปที่ s เช่นเดียวกัน แต่จะมีการเลื่อน p ไปทีละหนึ่งตำแหน่ง จนกว่าค่าที่ตำแหน่งที่ p ชี้อยู่จะเท่ากับ '\0' เมื่อนำ p ค่าสุดท้ายมาลบกับ s ที่ตำแหน่งเริ่มต้นก็จะได้ความยาวของข้อมูลที่ส่งเข้ามา
-

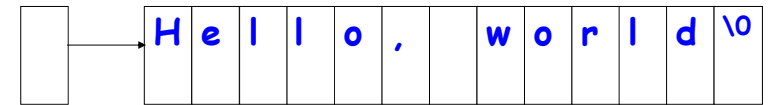
ตัวชี้ตัวอักษรและฟังก์ชัน (Character Pointer and Function)

- การทำงานกับข้อความหรือที่เรียกว่า สตริง (String) หรืออาร์เรย์ของข้อมูลประเภท char อาจจะใช้พอยน์เตอร์ชี้ไปยังข้อมูลประเภท char
 - การทำงานกับค่าคงที่สตริง (String Constant) สามารถเขียนภายในเครื่องหมาย “ ” เช่น “I am a string”
 - เมื่อมีการใช้ค่าคงที่สตริงจะมีการพื้นที่ในหน่วยความจำเท่ากับความยาวของค่าคงที่สตริงบวกด้วย 1 เนื่องจากลักษณะการเก็บข้อมูลประเภทข้อความในหน่วยความจำจะมีการปะตัวอักษร null หรือ '\0' ต่อท้ายเสมอเพื่อให้รู้ว่าเป็นจุดสิ้นสุดของข้อมูล
 - การจองพื้นที่ดังกล่าวจะเหมือนการจองพื้นที่ของข้อมูลประเภทอาร์เรย์ เป็นอาร์เรย์ของ char
-

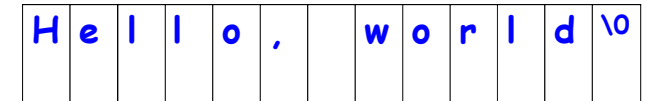
ตัวชี้ตัวอักษรและฟังก์ชัน

- ค่าคงที่สตริงที่พบเห็นได้เสมอได้แก่ข้อความที่ใช้ในฟังก์ชัน printf () เช่น
printf ("Hello, world\n");
- ฟังก์ชัน printf () จะรับพารามิเตอร์เป็นพอยน์เตอร์ชี้ไปยังแอดเดรสของข้อมูลตำแหน่งเริ่มต้นของอาร์เรย์ และนำข้อความนั้นแสดงออกทางอุปกรณ์แสดงข้อมูลมาตรฐาน
- ในการเขียนโปรแกรมจะสามารถใช้พอยน์เตอร์ชี้ไปค่าคงที่สตริงใด ๆ ก็ได้ เช่น
char *pmessage = "Hello, world";
- pmessage จะเป็นพอยน์เตอร์ประเภท char ชี้ไปที่อาร์เรย์ของตัวอักษร จะแตกต่างจากการใช้อาร์เรย์ทั่วไปเช่น
char amessage[] = "Hello, world";

pmessage



amessage



การจองพื้นที่ให้กับอาร์เรย์และตัวชี้ชี้ไปยังค่าคงที่สตริง

- ฟังก์ชัน strcpy () ทำหน้าที่สำเนาข้อความจากตัวแปรหนึ่งไปยังอีกตัวแปรหนึ่งเขียนในลักษณะอาร์เรย์

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

- ฟังก์ชัน strcpy () เขียนในลักษณะพอยน์เตอร์

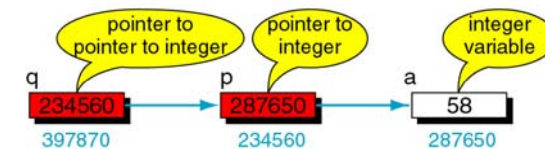
```
void strcpy( char *s, char *t )
{
    while ( ( *s = *t ) != '\0' ) {
        s++;
        t++;
    }
}
```

- ฟังก์ชัน strcpy () เขียนในลักษณะพอยน์เตอร์แบบสั้น

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

พอยน์เตอร์ซ้อนพอยน์เตอร์

```
/* Local Declarations */
int    a;
int    *p;
int    **q;
```



```
/* Statements */
a = 58;
p = &a;
q = &p;
printf(" %3d", a);
printf(" %3d", *p);
printf(" %3d", **q);
```