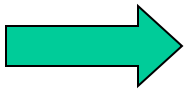
พอยเตอร์

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

```
int  age  =  23;
char ch  = 'a';
```



หน่วยความจำ

ตัวแปร age เก็บ 23

ตัวแปร ch เก็บ a

ประกาศตัวแปรพอยเตอร์

กำหนดค่าแอดเดรสของ age ให้ตัวแปรพอยเตอร์

ถ้าเป็นตัวแปรพอยเตอร์จะเก็บค่าแอดเดรสที่เก็บ 23 และ 'a'

```
#include <stdio.h>
void main()
{
        int  age;
        int  *pointer;
        age  =  25;
        pointer = &age;
}
```

การประกาศตัวแปรพอยน์เตอร์

ประเภทของข้อมูล *ชื่อตัวแปร

ตัวอย่าง

```
int  *p;
```

ประกาศตัวแปรชื่อ p ชี้ไปยังหน่วยความจำข้อมูลประเภท integer

```
int  a;
int  *p;
a  =  10;
p  =  &a;
```

ประกาศตัวแปร p เป็นพอยน์เตอร์

ใส่ค่า 10 ในตัวแปร a ซึ่งจะอยู่ในหน่วยความจำตำแหน่งหนึ่ง

นำค่าแอดเดรสที่เก็บ 10 ไปใส่ในตัวแปร p

การประกาศตัวแปรพอยน์เตอร์

int *pt_x;	สร้างตัวแปรพอร์ยเตอร์ชนิด int ทำให้ pt_x ใช้เก็บตำแหน่งที่อยู่ของตัวแปรชนิด int เท่านั้น
float *pt_num;	สร้างตัวแปรพอร์ยเตอร์ชนิด float ทำให้ pt_num ใช้เก็บตำแหน่งที่อยู่ของตัวแปรชนิด float เท่านั้น
char *pt_ch;	สร้างตัวแปรพอร์ยเตอร์ชนิด 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);
}
```

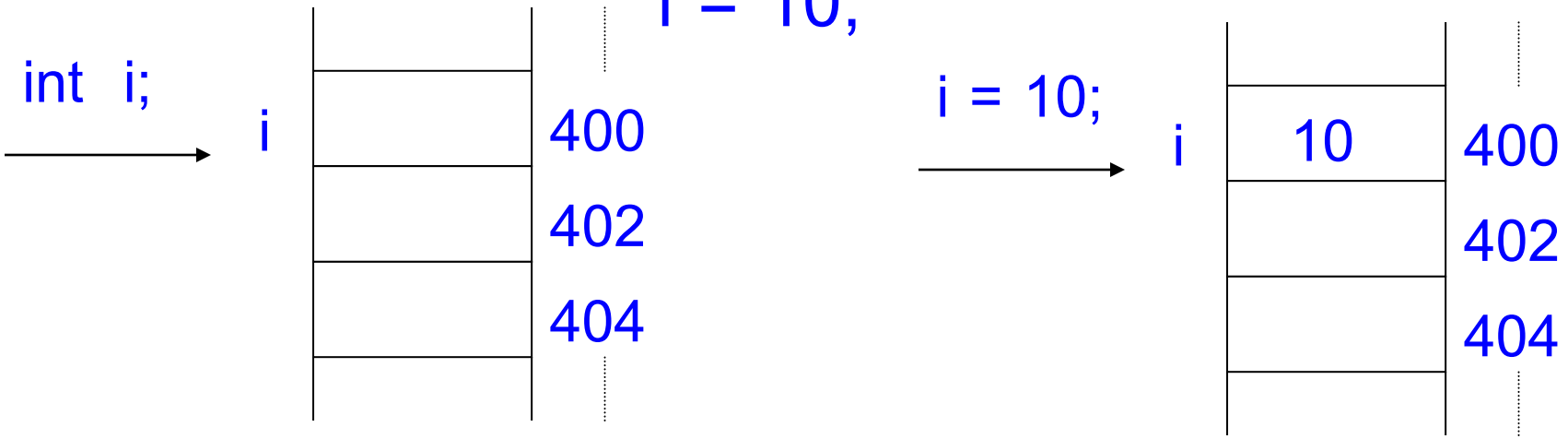
Address = 0065FDF4

```
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);
}
```

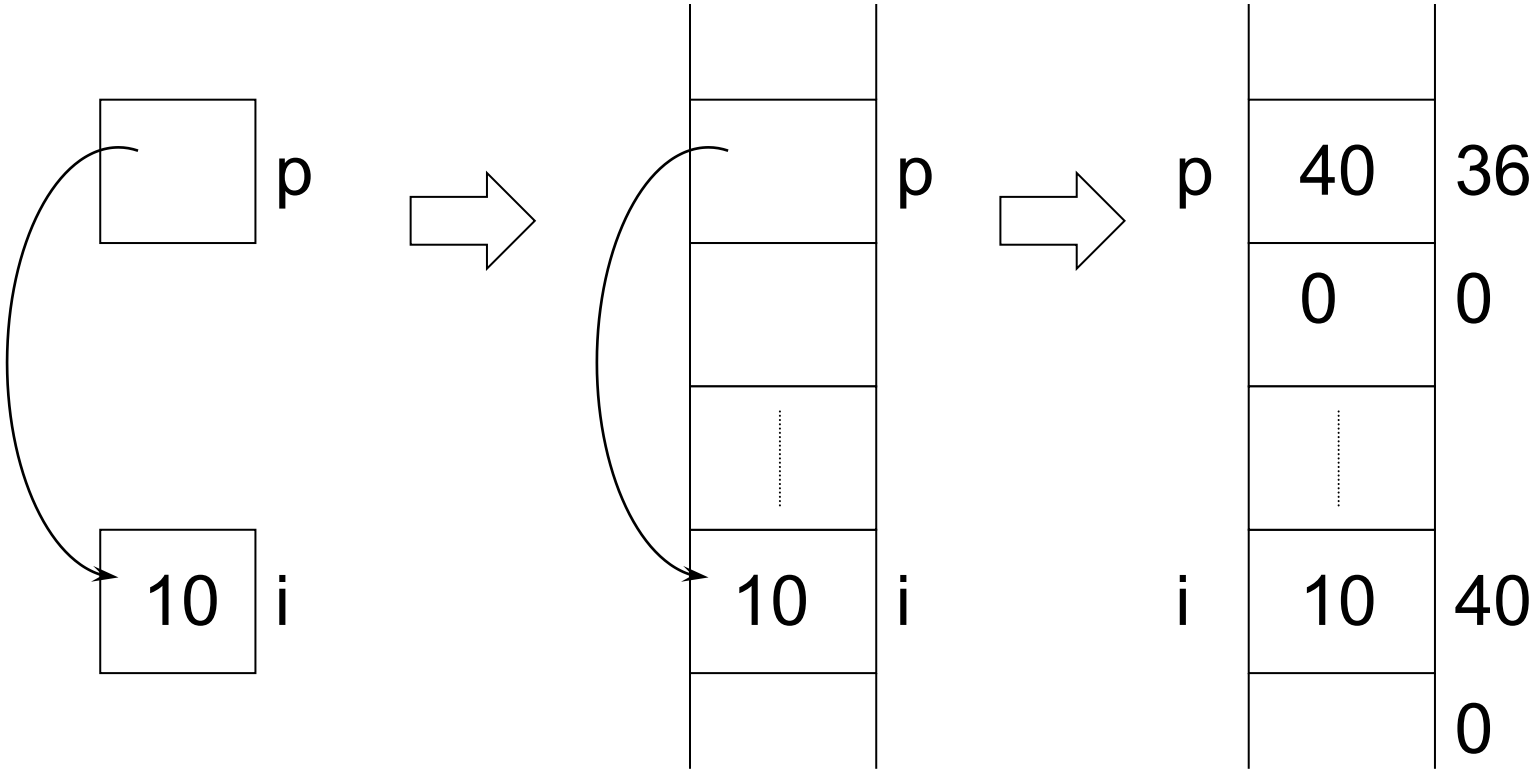
ตัวชี้กับแอดเดรส (Pointers and Address)

int i;

i = 10;



รูปการแทนข้อมูลในหน่วยความจำของตัวแปรประเภทพื้นฐาน



รูป การแทนข้อมูลในหน่วยความจำของตัวแปรประเภทตัวชี้

การประกาศตัวแปรประเภทตัวชี้

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

ตัวอย่าง

```
int   *ip;
```

เป็นการประกาศตัวแปร ip ให้เป็นตัวแปรพอยน์เตอร์ที่ชี้ไปยังตัวแปรประเภท int

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

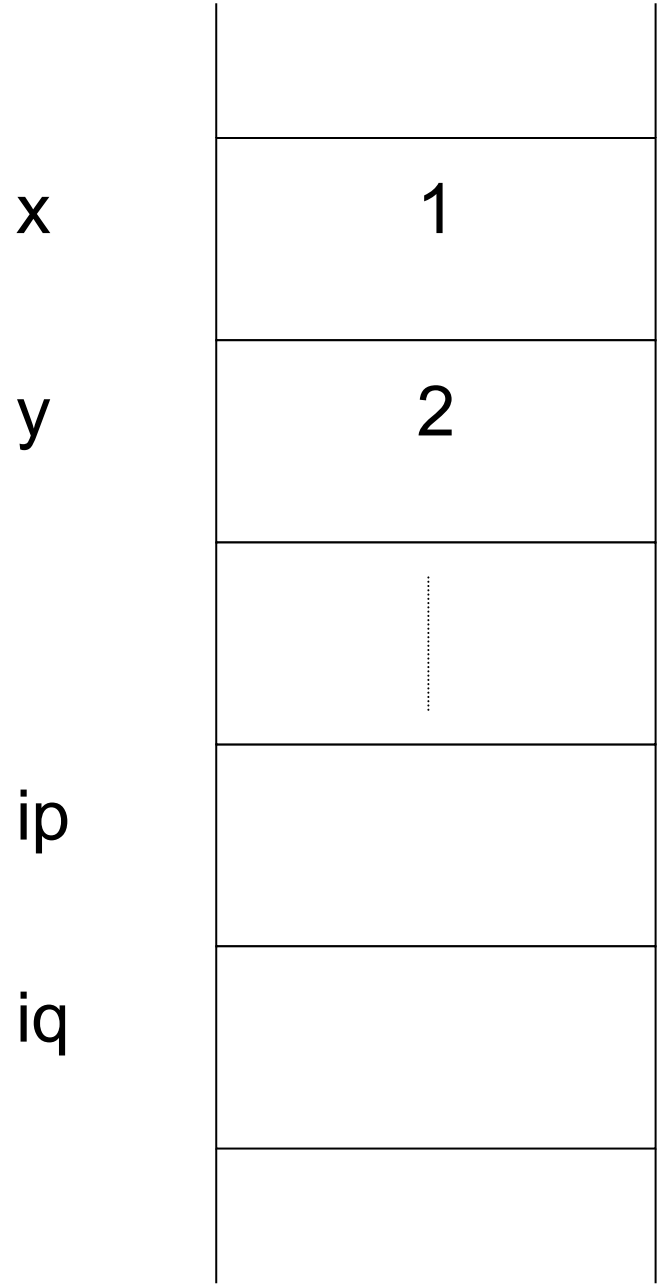
เป็นการประกาศตัวแปร dp เป็นตัวแปรพอยน์เตอร์ที่ชี้ไปยังตัวแปรประเภท double และประกาศฟังก์ชัน atof มีพารามิเตอร์เป็นตัวแปรพอยน์เตอร์ประเภท char

การกำหนดค่าและการอ่านค่าตัวแปรประเภทตัวชี้

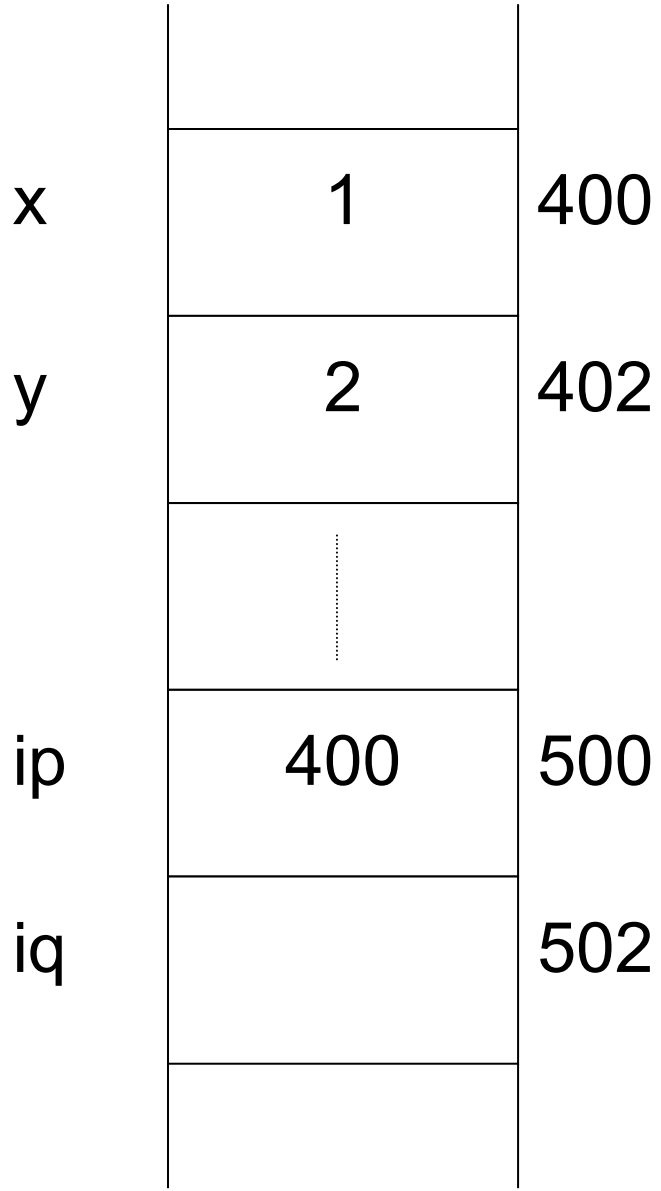
การกำหนดค่าให้กับตัวแปรพอยน์เตอร์จะเป็นการกำหนดแอดเดรสของตัวแปรที่มีประเภทสอดคล้องกับประเภทของตัวแปรพอยน์เตอร์เท่านั้น โดยการใช้ Unary Operator **&** **เป็นโอเปอเรเตอร์ที่อ้างถึงแอดเดรสของออปเจ็ค** (Object) **ใด ๆ**

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

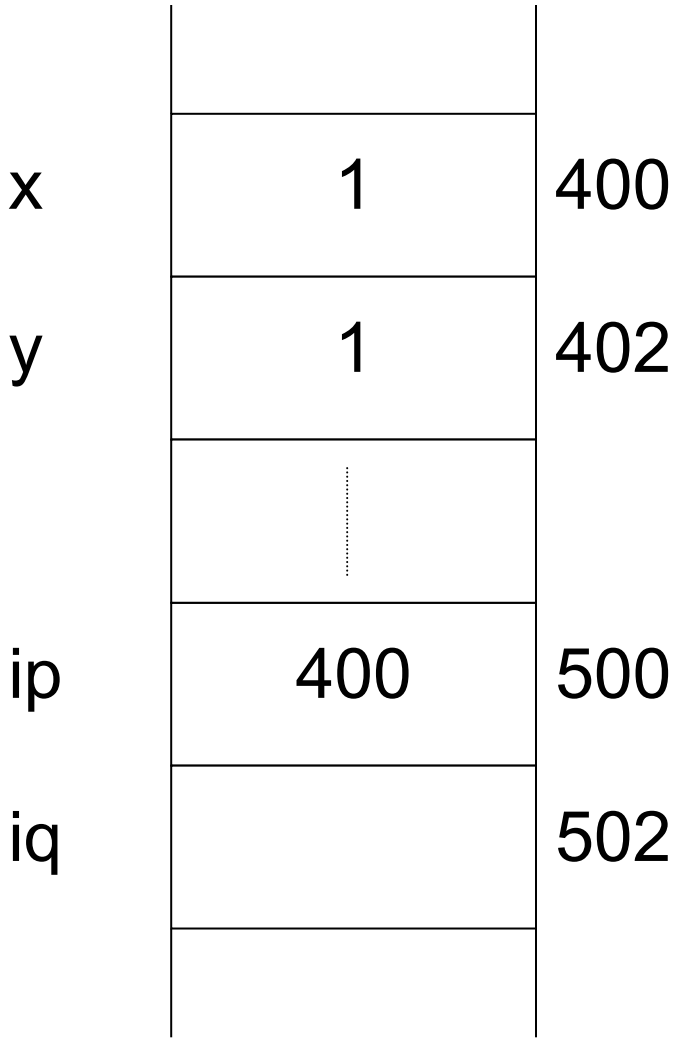
รูปการกำหนดค่าและการอ่านค่าตัวแปรตัวชี้



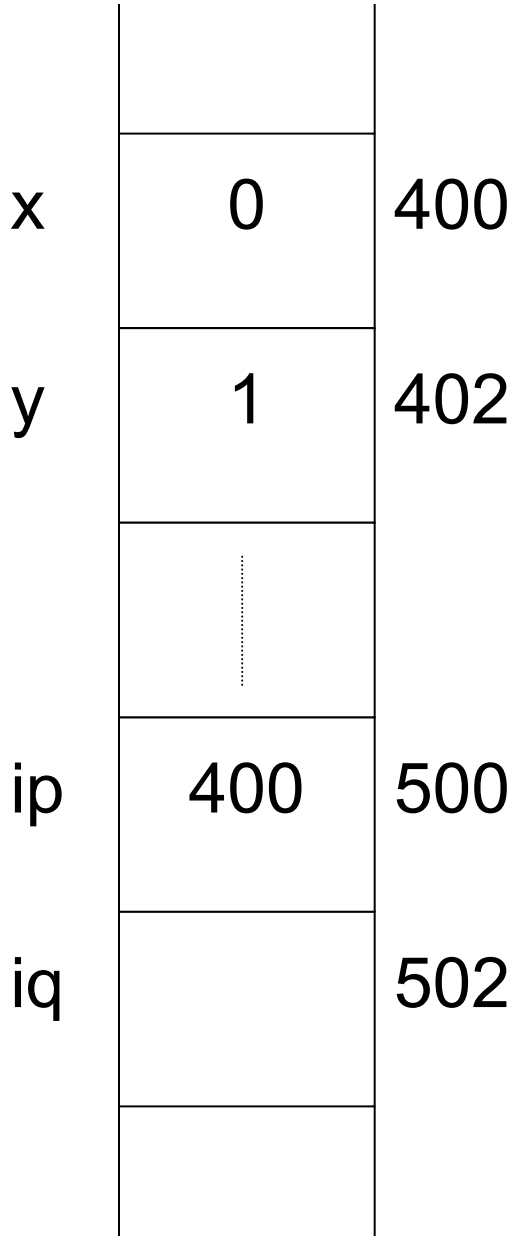

```
int  x = 1, y = 2;
int  *ip, *iq;
```



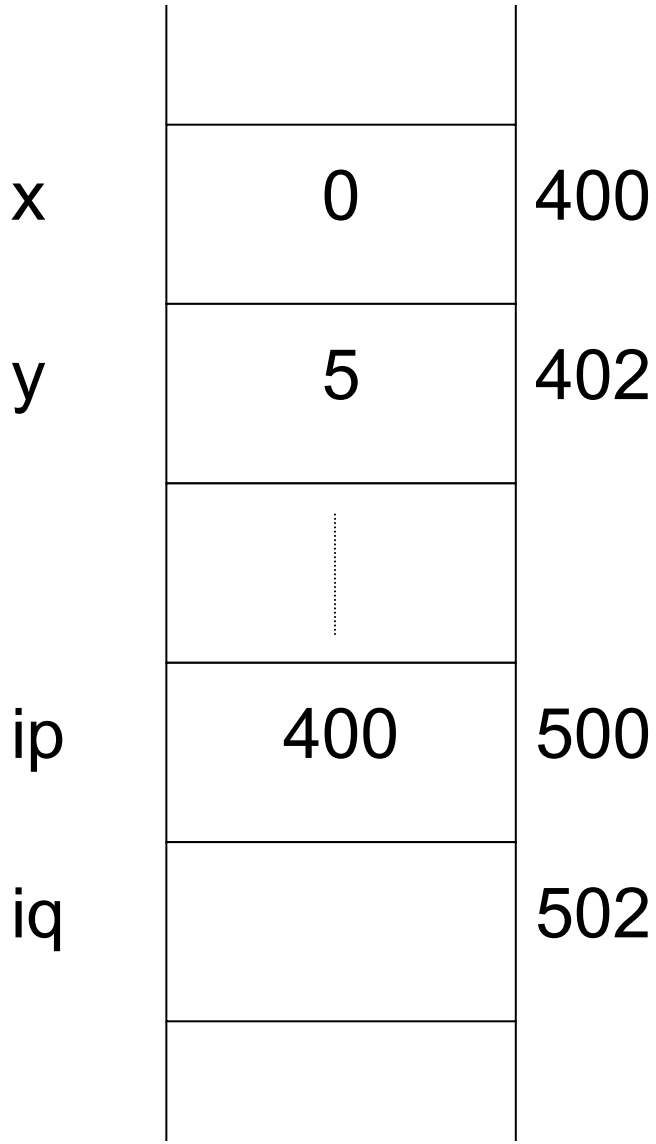

```
ip = &x;
```



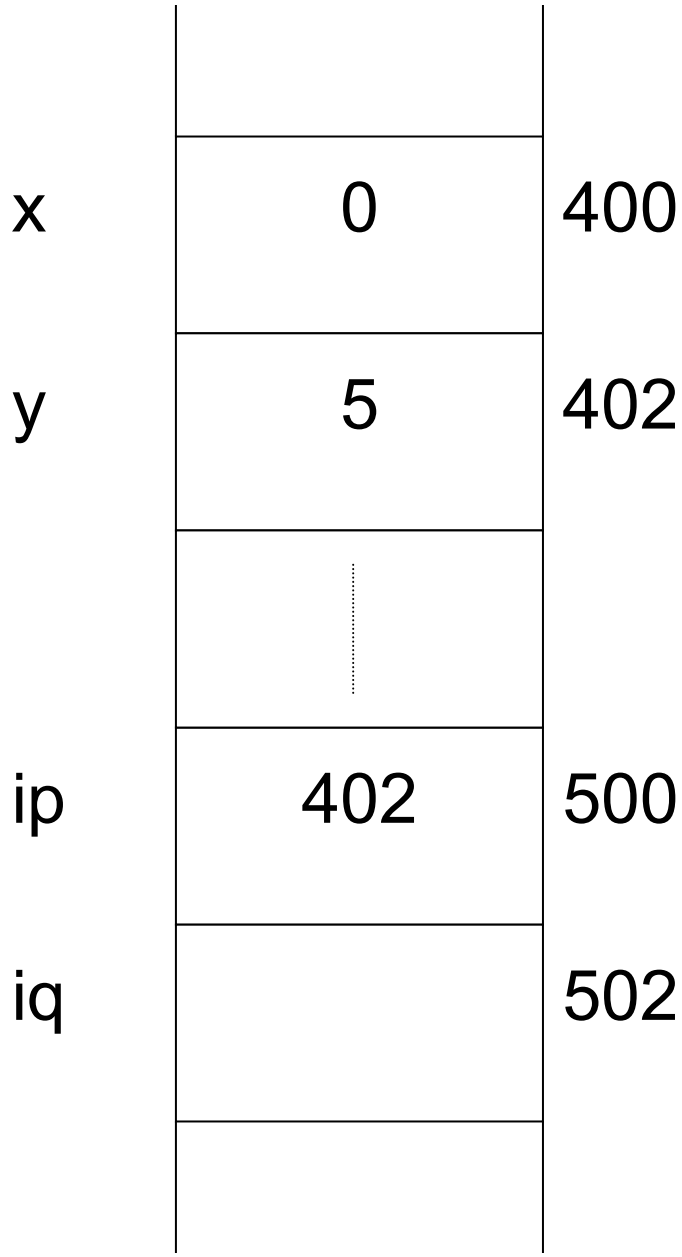

```
y = *ip;
```



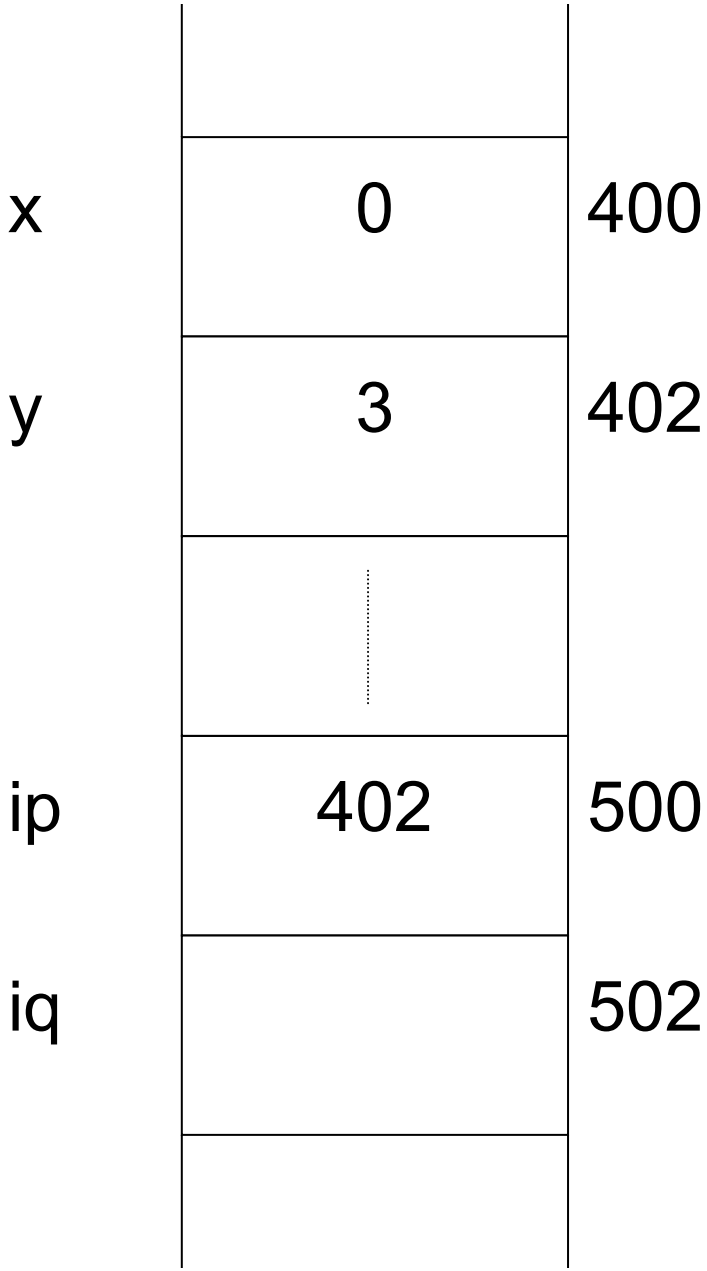

```
*ip = 0;
```



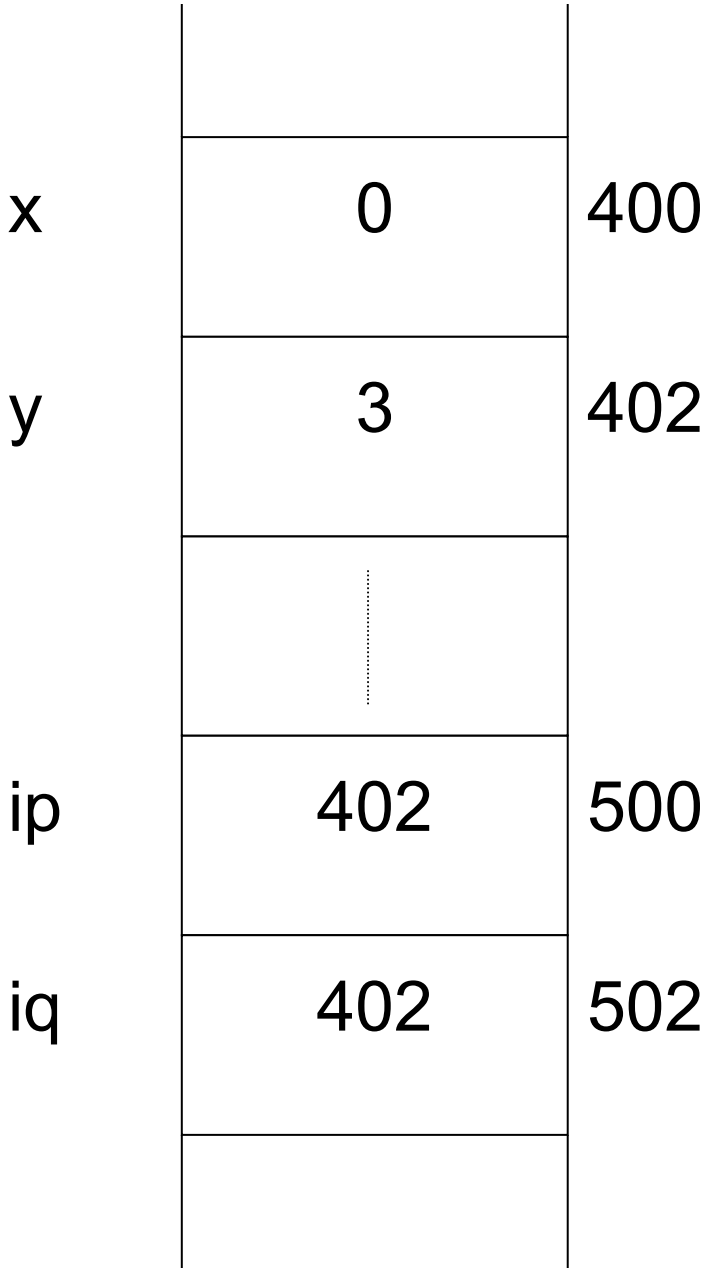
y = 5;



ip = &y;




```
*ip = 3;
```




```
iq =  ip;
```

ตัวชี้และอาร์กิวเมนท์ของฟังก์ชัน
(Pointer and Function Arguments)

เนื่องจากภาษาซีมีการส่งอากิวเมนต์ให้กับฟังก์ชันแบบ By Value และฟังก์ชันสามารถคืนค่า (return) ค่าได้เพียงหนึ่งค่า หากต้องการให้ฟังก์ชันมีการเปลี่ยนแปลงค่าและคืนค่ากลับมายังฟังก์ชันที่เรียกใช้มากกว่าหนึ่งค่าจะต้องนำพอยน์เตอร์เข้ามาช่วย

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

ตัวอย่าง

โปรแกรมตัวอย่างการสลับค่าตัวแปร 2 ตัว โดยผ่านฟังก์ชัน จะแสดงการส่งอาร์กิวเมนต์ในเป็นพอยน์เตอร์

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

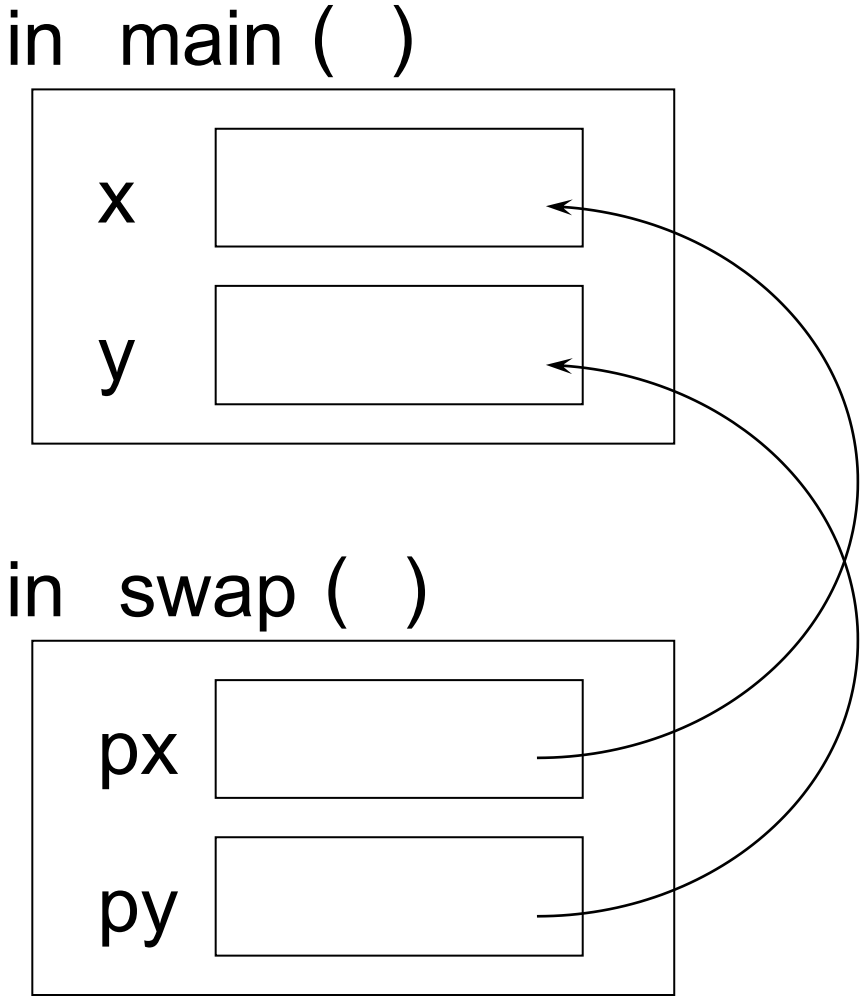
ตัวอย่าง (ต่อ)

```
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);
}
```

ตัวอย่าง (ต่อ)

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

อาร์กิวเมนท์ที่เป็นประเภทพอยน์เตอร์จะช่วยให้ฟังก์ชันสามารถเปลี่ยนค่าให้กับตัวแปรที่ส่งเข้ามาได้ เนื่องจากอาร์กิวเมนท์นั้นจะเก็บแอดเดรสของตัวแปรที่ส่งเข้ามา เมื่อมีการเปลี่ยนแปลงค่าของอาร์กิวเมนท์ผ่าน Dereferencing Operator (*) ค่าของตัวแปรที่ส่งเข้ามาจะถูกเปลี่ยนค่าพร้อมกันในทันที



รูปแสดงความสัมพันธ์ของการส่งอาร์กิวเมนท์แบบพอยน์เตอร์กับฟังก์ชัน

การใช้ตัวชี้กับอาร์เรย์

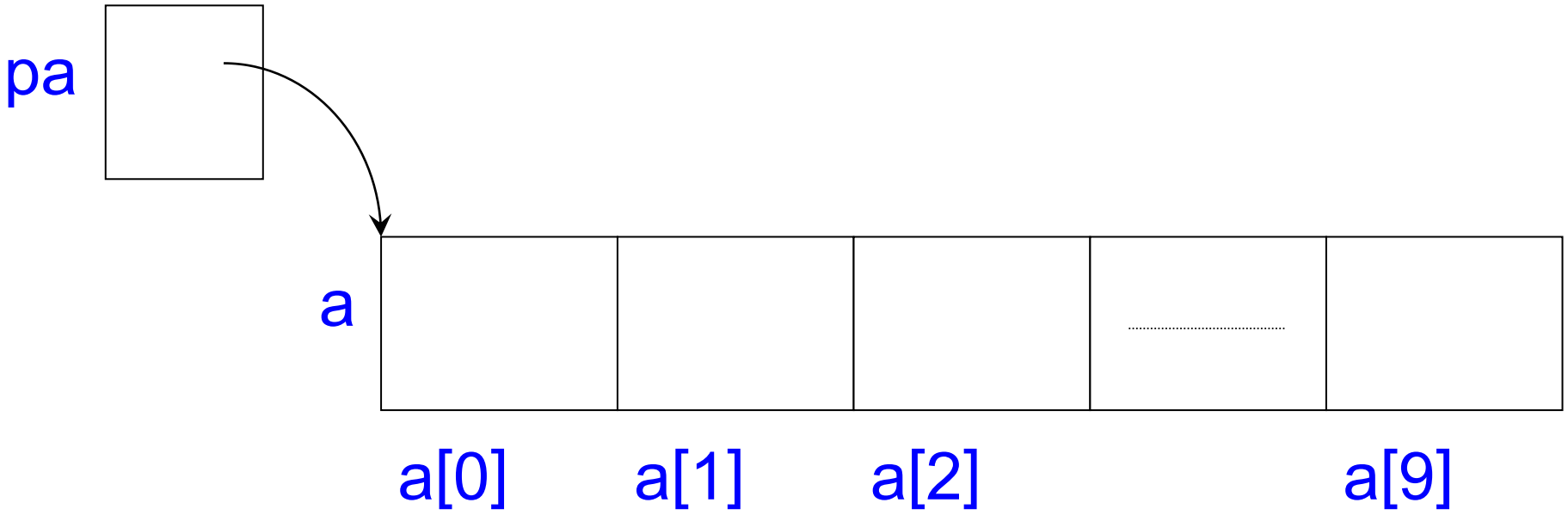
การทำงานใด ๆ ของอาร์เรย์สามารถใช้พอยน์เตอร์เข้ามาช่วย ซึ่งจะทำให้มีความเร็วในการทำงานสูงขึ้น สมมติว่ามีอาร์เรย์ a และพอยน์เตอร์ pa ดังนี้

```
int   a[10];
int   *pa;
```

กำหนดให้พอยน์เตอร์ pa ชี้ไปยังอาร์เรย์ a ด้วยคำสั่ง

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

pa จะเก็บค่าแอดเดรสเริ่มต้นของอาร์เรย์ a



รูป แสดงตัวชี้ชี้ไปยังแอดเดรสเริ่มต้นของอาร์เรย์

การนำไปใช้งานจะสามารถอ่านค่าอาร์เรย์ผ่านพอยน์เตอร์ได้ดังนี้

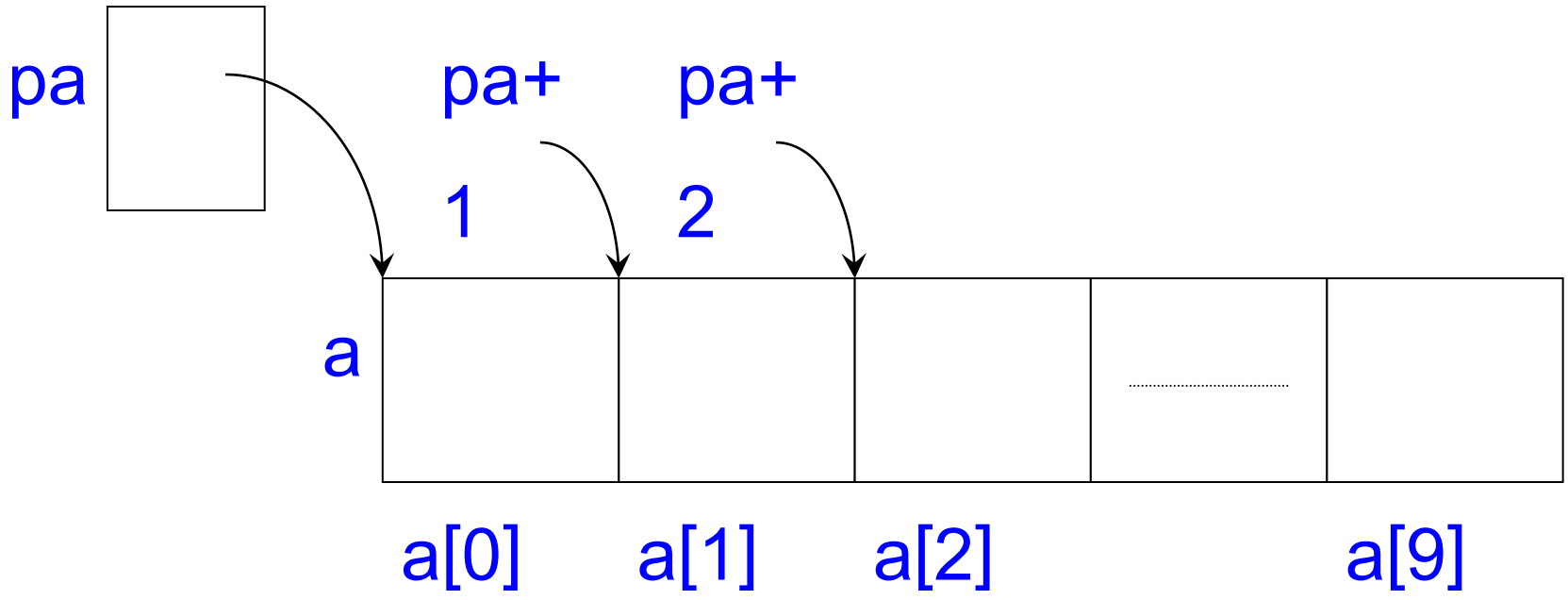
```
x  =  *pa;
```

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)



รูป แสดงการอ้างถึงตำแหน่งในอาร์เรย์ผ่านตัวชี้

การสั่งให้บวก 1 หรือบวก i หรือ ลบ i เป็นเหมือนการเลื่อนไปยังสมาชิกของอาร์เรย์ตำแหน่งที่ต้องการ เนื่องจากประเภทของข้อมูลแต่ละประเภทของอาร์เรย์ เช่น int, float, double และอื่น ๆ มีขนาดของข้อมูลที่ต่างกัน ทำให้ขนาดของสมาชิกภายในอาร์เรย์แต่ละประเภทมีขนาดแตกต่างกันด้วย การสั่งให้บวกหรือลบด้วยจำนวนที่ต้องการนั้นจะมีกลไกที่ทำหน้าที่คำนวณตำแหน่งที่ต้องการให้สอดคล้อง กับข้อมูลแต่ละประเภทโดยอัตโนมัติ

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

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

เมื่อมีการส่งชื่อของอาร์เรย์ให้แก่ฟังก์ชัน จะเป็นการส่งตำแหน่งแอดเดรสของสมาชิกตัวแรกของอาร์เรย์ให้แก่ฟังก์ชัน ดังนั้นพารามิเตอร์ในฟังก์ชันนั้นจะเป็นตัวแปรประเภทพอยน์เตอร์

ตัวอย่าง

ฟังก์ชันที่รับพารามิเตอร์เป็นพอยน์เตอร์ โดยอาร์กิวเมนท์ที่ส่งมาเป็นอาร์เรย์

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

จะเห็นว่า s เป็นพอยน์เตอร์ ในฟังก์ชันจะมีการตรวจสอบข้อมูลว่ามีค่าเท่ากับ '\0' หรือไม่ และมีการเลื่อนตำแหน่งทีละ 1 ค่า (นับว่าข้อมูลมีความยาวเพิ่มขึ้นทีละ1) โดยใช้ s++

การเรียกใช้ฟังก์ชัน strlen สามารถทำได้หลายลักษณะ

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

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

ตัวอย่าง

f (&a[2])

หรือ f (a+2)

เป็นการส่งแอดเดรสของสมาชิก a[2] ให้กับฟังก์ชัน f การประกาศฟังก์ชัน f สามารถทำได้โดยการประกาศ

f (int arr[ ]) { ..... }

หรือ f (int *arr) { ..... }

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

p < q

จะเป็นจริงเมื่อ p ชี้ไปที่สมาชิกที่อยู่ก่อนหน้าสมาชิกที่ q ชี้อยู่ การเปรียบเทียบในลักษณะจะใช้ได้ต่อเมื่อ p และ q ชี้ไปที่อาร์เรย์เดียวกันเท่านั้น

นอกจากนี้ยังสามารถใช้การลบหรือการบวกกับพอยน์เตอร์ได้เช่นเดียวกัน แต่สิ่งที่ควรระวังคือการทำเช่นนั้นจะต้องอยู่ในขอบเขตขนาดของอาร์เรย์เท่านั้น

ตัวอย่าง

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

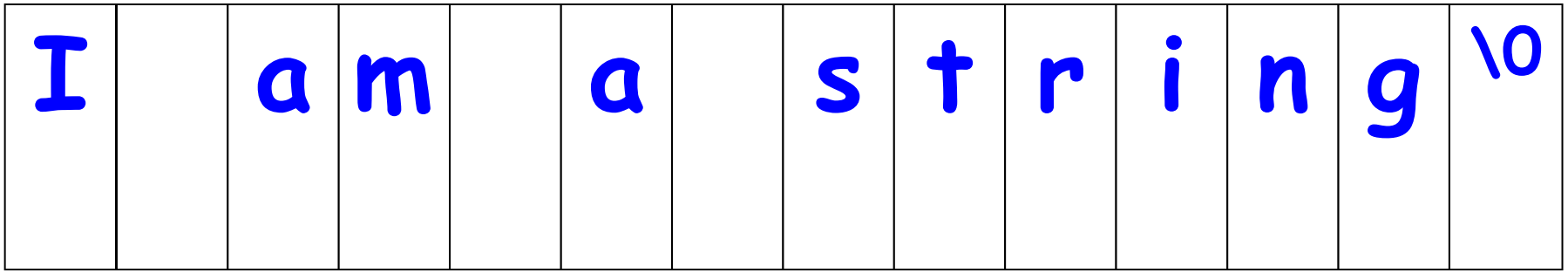
(Character Pointer and Function)

การทำงานกับข้อความหรือที่เรียกว่า สตริง (String) เป็นการใช้ข้อมูลตัวอักษรหลาย ๆ ตัว หรืออาร์เรย์ของข้อมูลประเภท char หรืออาจจะใช้พอยน์เตอร์ชี้ไปยังข้อมูลประเภท char การทำงานกับค่าคงที่สตริง (String Constant) สามารถเขียนภายในเครื่อง “ ”

ตัวอย่าง

"I am a string"

เมื่อมีการใช้ค่าคงที่สตริงจะมีการพื้นที่ในหน่วยความจำเท่ากับความยาวของค่าคงที่สตริงบวกด้วย 1 เนื่องจากลักษณะการเก็บข้อมูลประเภทข้อความในหน่วยความจำจะมีการปะตัวอักษร null หรือ '\0' ต่อท้ายเสมอเพื่อให้รู้ว่าเป็นจุดสิ้นสุดของข้อมูล การจองพื้นที่ดังกล่าวจะเหมือนการจองพื้นที่ของข้อมูลประเภทอาร์เรย์ เป็นอาร์เรย์ของ char



รูป แสดงแบบจำลองการเก็บข้อมูลประเภท สตริงใน หน่วยความจำ

ค่าคงที่สตริงที่พบเห็นได้เสมอได้แก่ข้อความที่ใช้ในฟังก์ชัน printf ( ) เช่น

printf (“Hello, world\n”);

ฟังก์ชัน printf ( ) จะรับพารามิเตอร์เป็นพอยน์เตอร์ชี้ไปยังแอดเดรสของข้อมูลที่ตำแหน่งเริ่มต้นของอาร์เรย์ และนำข้อความนั้นแสดงออกทางอุปกรณ์แสดงข้อมูลมาตรฐาน

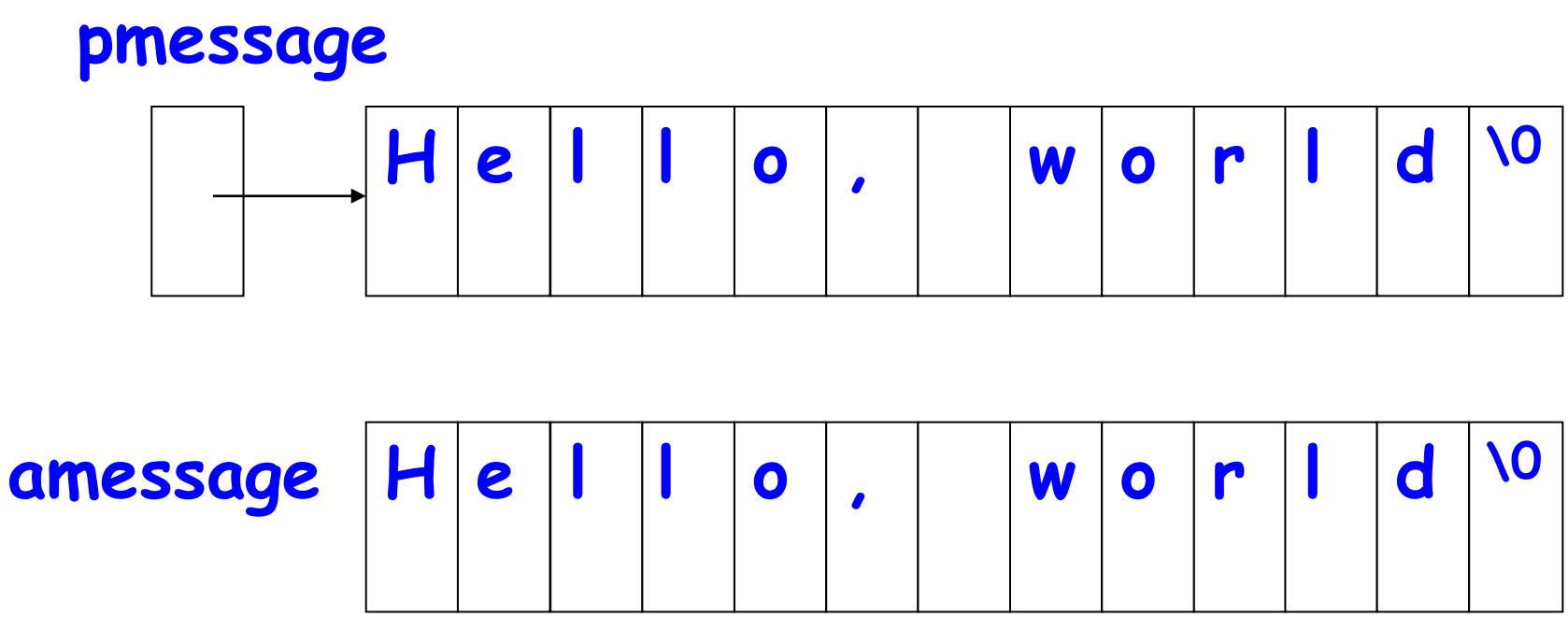
ในการเขียนโปรแกรมจะสามารถใช้พอยน์เตอร์ชี้ไปค่าคงที่สตริงใด ๆ ก็ได้ เช่น

```
char  *pmessage = “Hello, world”;
```

pmessage จะเป็นพอยน์เตอร์ประเภท char ชี้ไปที่อาร์เรย์ของตัวอักษร จะแตกต่างจากการใช้อาร์เรย์ทั่วไปเช่น

```
char  amessage[ ] = “Hello, world”;
```

ลักษณะของอาร์เรย์เช่น amessage จะมีการจองพื้นที่ใช้กับอาร์เรย์ขนาด 13 ตัวอักษร รวมทั้ง null ส่วนลักษณะของพอยน์เตอร์ที่ชี้ไปยังค่าคงที่สตริง จะมีการจองพื้นที่ให้กับค่าคงที่สตริงขนาด 13 ตัวอักษรเช่นเดียวกัน แต่จะมีการจองพื้นที่ให้กับพอยน์เตอร์และทำการชี้พอยน์เตอร์นั้นไปยังพื้นที่ของค่าคงที่สตริงที่จองเอาไว้



รูป การจองพื้นที่ให้กับอาร์เรย์และตัวชี้ชี้ไปยังค่าคงที่สตริง

ตัวอย่าง

ฟังก์ชัน strcpy ( ) ทำหน้าที่สำเนาข้อความจากตัวแปรหนึ่งไปยังอีกตัวแปรหนึ่งเขียนในลักษณะอาร์เรย์

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

ตัวอย่าง

ฟังก์ชัน strcpy ( ) เขียนในลักษณะพอยน์เตอร์

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

ตัวอย่าง

ฟังก์ชัน strcpy ( ) เขียนในลักษณะพอยน์เตอร์แบบสั้น

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