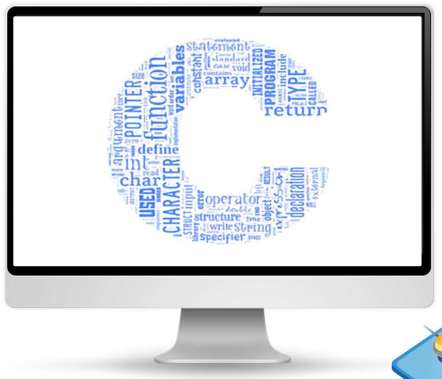
02204171 Structured Programming

Chapter 13 : Pointer (part 1)



Outline

- Pointer Basic
  - Address in C
  - What is a Pointer?
  - Pointer Declaration/Initialization
- Pointer Operators
- Pointer Arithmetic & Comparison
- Pointer and Array

1.1 Pointer Basic: Address in C

หน่วยความจำของคอมพิวเตอร์ สามารถมองได้ในลักษณะของช่องเก็บข้อมูล (memory cell) ขนาด 1 byte ที่ต่อเนื่องกัน โดยแต่ละช่องหน่วยความจำ จะมีหมายเลขเฉพาะสำหรับการอ้างอิง ที่เรียกว่า **แอดเดรส (Address)**

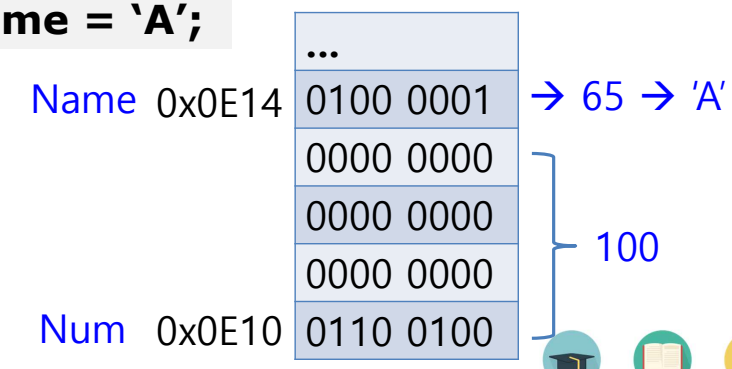
ตำแหน่งหน่วยความจำ เริ่มตั้งแต่ 0 จนถึง ค่าสูงสุดของขนาดหน่วยความจำ (byte) เช่น RAM ขนาด 64KB (ตำแหน่งหน่วยความจำจะมีค่าเป็น 0 (0x0000) – 65535 (0xFFFF))

1.1 Pointer Basic: Address in C

- **การเก็บข้อมูลในหน่วยความจำ (Address in C)**

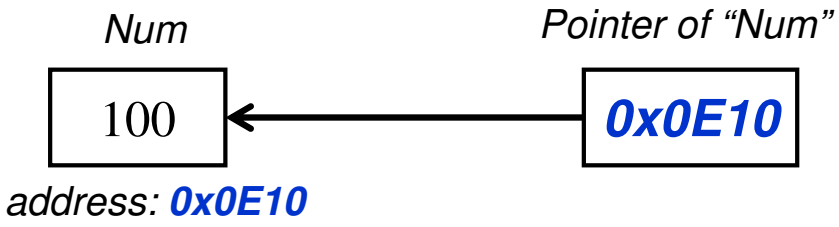
  เมื่อมีการประกาศตัวแปรใดๆ ตัวแปลภาษา C (compiler) จะจองพื้นที่ในหน่วยความจำเพื่อเก็บค่าตัวแปร

```
int Num= 100;
char Name = 'A';
```



1.2 Pointer Basic: What is a Pointer?

- **พอยน์เตอร์ (Pointer)** เป็นตัวแปรชนิดพิเศษในภาษาซี ทำหน้าที่ **เก็บตำแหน่งในหน่วยความจำ (Address)** ของตัวแปรชนิดอื่นๆ แทนการเก็บข้อมูลเหมือนกันตัวแปรพื้นฐานชนิดอื่นๆ



1.2 Pointer Basic: What is a Pointer?

การใช้พอยน์เตอร์เป็นจุดเด่นอย่างหนึ่งในการเขียนโปรแกรมภาษาซี

- การใช้ตัวแปรพอยน์เตอร์ชี้ไปที่ตำแหน่งหน่วยความจำนั้นทำให้เราสามารถเข้าถึงตัวแปรได้โดยตรง ส่วนใหญ่ใช้ในการเขียนโปรแกรมควบคุมการทำงานของอุปกรณ์ ประเภทไอซี (Integrated Circuit) หรือคอนโทรลเลอร์ (Controller)
- การใช้ตัวแปรพอยน์เตอร์ ทำให้การเขียน-อ่านข้อมูลจำนวนมาก เช่น อาเรย์ หรือการทำงานกับไฟล์มีประสิทธิภาพ
- การใช้ตัวแปรพอยน์เตอร์ ทำให้เราสามารถจัดสรรหน่วยความจำแบบไดนามิคได้ (dynamic memory allocation)

1.3 Pointer Basic: Pointer Declaration

- **การประกาศตัวแปรพอยน์เตอร์**

```
ชนิดข้อมูล  * ชื่อตัวแปร;
```

int char float double

การประกาศตัวแปรพอยน์เตอร์ทำได้เช่นเดียวกันกับการประกาศตัวแปรธรรมดา เพียงแต่เพิ่มเครื่องหมาย "*"

- **ตัวอย่าง 1**

```
int   *ptr_int;      // Pointer of int variable
float *ptr_float;    //Pointer of float variable
int a=5;
float b= 5.3;
```


1.3 Pointer Basic: Pointer Declaration

- **ตัวอย่าง การประกาศตัวแปรพอยน์เตอร์ (เพิ่มเติม)**

```
float* ptr_float, b;
float *ptr_float, b;
float  b,*ptr_float;
```

ผลของการประกาศตัวแปรทั้ง 3 แบบ หมายความว่า ptr_float เป็นพอยน์เตอร์สำหรับชี้ตัวแปรชนิด float และ b เป็นตัวแปรธรรมดาชนิด float

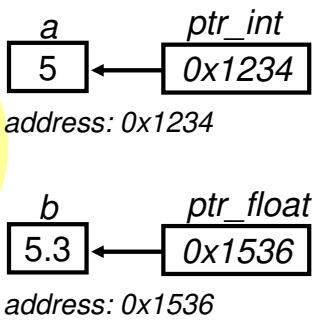
1.3 Pointer Basic: Pointer Declaration

หลังจากประกาศตัวแปรพอยน์เตอร์ ต้องทำการกำหนดให้พอยน์เตอร์ชี้ไปยังตัวแปรเป้าหมาย (Assignment of Pointer Variables) ซึ่งก็คือ**กำหนดค่าของตัวแปรพอยน์เตอร์** ให้มีค่าเป็น ตำแหน่งในหน่วยความจำของตัวแปรเป้าหมาย

- **ตัวอย่าง 2.1**

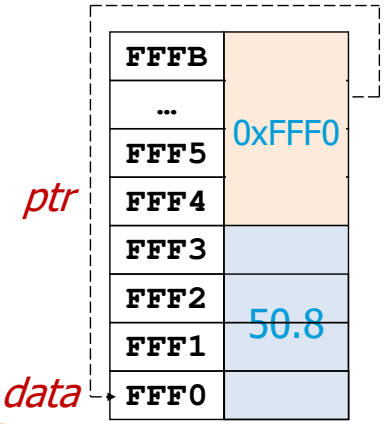
```
int   *ptr_int;
float *ptr_float;
int a=5;
float b= 5.3;
ptr_int=&a;
ptr_float=&b;
```

การกำหนดค่าตำแหน่งในหน่วยความจำของตัวแปรเป้าหมาย (ชนิดข้อมูลพื้นฐาน) ให้กับพอยน์เตอร์ ใช้ตัวดำเนินการ “&”



1.3 Pointer Basic: Assignment of Pointer Variables

```
float data = 50.8;
float *ptr;
ptr = &data;
ptr = 120;
```



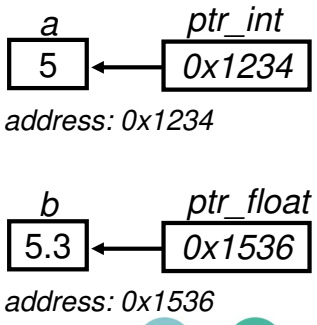
การกำหนดค่าที่ไม่ใช่ตำแหน่งในหน่วยความจำให้พอยน์เตอร์อาจทำให้โปรแกรมทำงานผิดพลาด

1.3 Pointer Basic: Pointer Declaration/Initialization

- **การประกาศพร้อมกับกำหนดค่าเริ่มต้นของตัวแปรพอยน์เตอร์** (Pointer Initialization)
  - เราสามารถประกาศตัวแปรพอยน์เตอร์ พร้อมทั้งกำหนดค่าของตัวแปรพอยน์เตอร์ ให้มีค่าเป็น ตำแหน่งในหน่วยความจำของตัวแปรเป้าหมาย

- **ตัวอย่าง 2.2**

```
int a=5;
float b= 5.3;
int   *ptr_int = &a;
float *ptr_float= &b;
```



1.3 Pointer Basic: Pointer Declaration/Initialization

- **ตัวอย่าง 3**

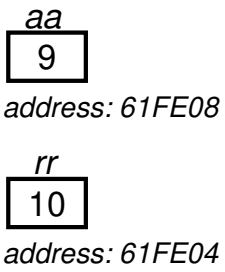
```
8 8
The values of ip and fp are: 0000000000000000 0000000000000000
int *ip = NULL;
float *fp = 0;
printf("%d %d\n",sizeof(ip), sizeof(fp));
printf("The values of ip and fp are: %p %p\n", ip, fp);

int a=5;
ip = &a;
printf("The value of ip is : %p\n", ip);
The value of ip is :  000000000061FE0C
```

%p ใช้แสดงค่า address ของตัวแปรพอยน์เตอร์

Quick check1

1. จงระบุชื่อตัวแปรพอยน์เตอร์ในโปรแกรมนี้
2. จงแก้ไขข้อผิดพลาดในโปรแกรมนี้ แล้วแสดงผลลัพธ์ทางจอภาพ

```
int *p1 ;
int* p2,p3;
int aa=9,rr=10;
p1 = &aa ;
p2 = &rr ;
p3 = &aa;
printf("p1 address of %d is %p \n", aa , p1);
printf("p2 address of %d is %p \n",rr , p2);
printf("Address of aa is %p \n",&aa);
```



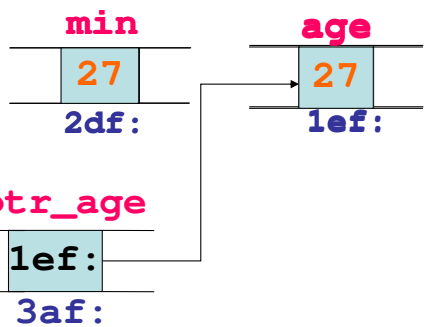
2. Pointer Operators

- **ตัวดำเนินการที่ใช้กับตัวแปรพอยเตอร์ (pointer operators)**
  - **เครื่องหมาย & (Ampersand)** หมายถึง Address operator เป็นตัวดำเนินการที่ใช้เพื่อบอก**ตำแหน่งในหน่วยความจำของตัวแปร**อื่นๆ
  - **เครื่องหมาย** * (Asterisk) หมายถึง Dereferencing operator เป็นตัวดำเนินการ ที่ใช้เมื่อต้องการเข้าถึง/ดำเนินการกับ**ค่าที่เก็บในตำแหน่งที่ตัวแปรพอยน์เตอร์นั้นชี้อยู่**
    - เป็นวิธีทางอ้อมในการเข้าถึงค่าของตัวแปร ดังนั้น จึงเรียกอีกชื่อหนึ่งว่า “indirect operator”

2. Pointer Operators

- **ตัวอย่าง 4**

```
int age=13,min;
int *ptr_age = &age;
*ptr_age = 27;
min= *ptr_age ;
```



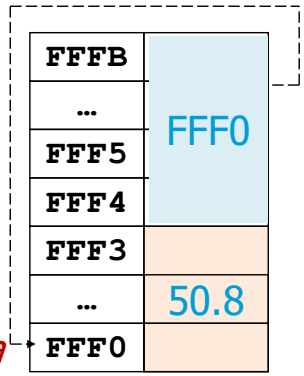
min = ค่าที่เก็บใน**ตำแหน่งที่** ptr_age ชี้อยู่

เครื่องหมาย & : การอ้างถึง**ตำแหน่ง**ของตัวแปร

เครื่องหมาย *: การอ้างถึง**ค่า**ในตัวแปรที่พอยน์เตอร์ชี้

2. Pointer Operators

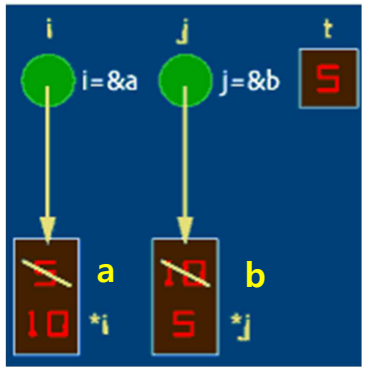
```
float data = 50.8;
float *ptr;
ptr = &data;
printf("%p %.1f\n",ptr,*ptr);
*ptr = 27.4;
printf("*ptr=%.1f\n",*ptr);
printf("data=%.1f", data);
```



2. Pointer Operators

ตัวอย่าง 5: Using Pointers for Function Parameter

```
#include <stdio.h>
 void swap(int *i, int *j){
   int t = *i;
   * i = *j;
   *j = t;
 }
 void main() {
  int a=5; b=10;
  printf("%d %d\n",a,b);
 swap(&a,&b);
  printf("%d %d\n",a,b);
 }
```



https://computer.howstuffworks.com/c26.htm

Quick check2

1. แสดงผลลัพธ์ทางจอภาพของโปรแกรมต่อไปนี้

```
int a,b,*c;
a=b=10;
printf("a=%d b=%d\n",a,b);

c=&a;
*c=21;

printf("a=%d b=%d\n",a,b);
```


Quick check2

2. ข้อใดเป็นการดำเนินการกับตัวแปรพอยน์เตอร์ที่ถูกต้อง

```
int a,b,*c;
float d;
a=b=10;
---- ✡ ----
```

✡
c = 20;
c = &b;
*c = &b;
*c = b;
int *d=c;
c = &d;

3. Pointer Arithmetic & Comparison

ตัวแปรพอยน์เตอร์กับการกระทำทางคณิตศาสตร์

ตำแหน่งในหน่วยความจำก็คือ เลขจำนวนเต็ม ดังนั้นเราสามารถดำเนินการทางคณิตศาสตร์กับค่าตำแหน่งในหน่วยความจำได้

ตัวดำเนินการคณิตศาสตร์กับตำแหน่งในหน่วยความจำในภาษา C ได้แก่

ไม่มีการ * , / พอยน์เตอร์

++	หมายถึง	การเพิ่มค่าครั้งละ 1
--	หมายถึง	การลดค่าครั้งละ 1
+, +=	หมายถึง	การบวก
- , -=	หมายถึง	การลบ

Increment and decrement

Addition and Subtraction

3. Pointer Arithmetic & Comparison

- ตัวแปรพอยน์เตอร์กับการกระทำทางคณิตศาสตร์

(ตัวดำเนินการเพิ่มค่า และ ลดค่า)

การเพิ่มค่า (Increment) ตัวแปรพอยน์ตอร์ : หมายถึง การเลื่อนตัวชี้จากตำแหน่งปัจจุบันขึ้น (เลื่อนตำแหน่งในหน่วยความจำ) ไป N ไบต์

การลดค่า (decrement) ตัวแปรพอยน์ตอร์ : หมายถึง การเลื่อนตัวชี้จากตำแหน่งปัจจุบันลง (เลื่อนตำแหน่งในหน่วยความจำ) ไป N ไบต์

N คือ ขนาดของแต่ละชนิดตัวแปรที่พอยน์เตอร์นั้นชี้

เช่น char มีขนาด 1 byte

int มีขนาด 2 bytes/4 bytes

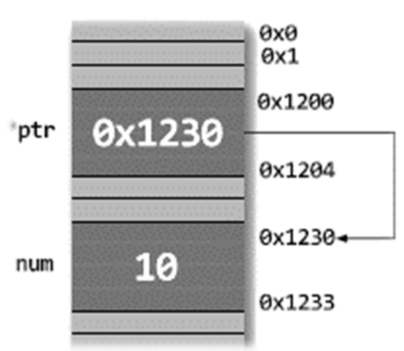
3. Pointer Arithmetic & Comparison

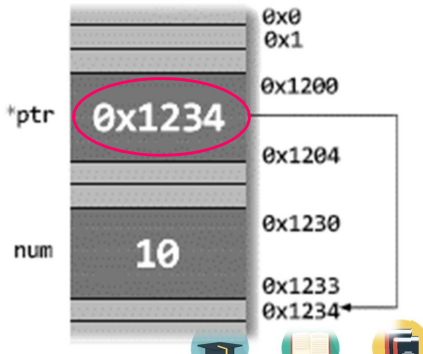
- ตัวแปรพอยน์เตอร์กับการกระทำทางคณิตศาสตร์

ตัวอย่าง 6.1 (ตัวดำเนินการเพิ่มค่า และ ลดค่า)

```
int num = 10, *ptr;
ptr = &num;
ptr++;
```

เลื่อนพอยท์เตอร์ไป 1 ตำแหน่งของ int (4 bytes)





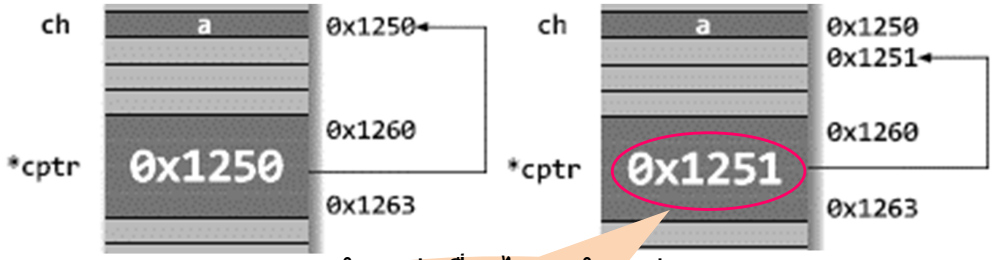
3. Pointer Arithmetic & Comparison

- ตัวแปรพอยน์เตอร์กับการกระทำทางคณิตศาสตร์

ตัวอย่าง 6.2 (ตัวดำเนินการเพิ่มค่า และ ลดค่า)

```
char ch = 'a', *cptr;
cptr = &ch;
cptr++;
```

ส่วนใหญ่ตัวดำเนินการ ++, -- จะใช้เมื่อนำพอยน์เตอร์มาใช้กับอาเรย์



ตำแหน่งเลื่อนไป 1 ตำแหน่งของ char (1 byte)

3. Pointer Arithmetic & Comparison

- ตัวแปรพอยน์เตอร์กับการกระทำทางคณิตศาสตร์

(ตัวดำเนินการเพิ่มค่า และ ลดค่า)

ตัวอย่าง 7

```
float *ptr=(float *)4000;
printf("Initial Address : %u\n",ptr);
ptr--;
printf("New Value of ptr:%u",ptr);
```

Initial Address: 4000
New Value of ptr : 3996

Data Type	Initial Address	Operation	Address after Operations	Required Bytes
int	4000	++	4004	4
int	4000	- -	3996	4
char	4000	++	4001	1
char	4000	- -	3999	1
float	4000	++	4004	4
float	4000	- -	3996	4

3. Pointer Arithmetic & Comparison

ตัวแปรพอยน์เตอร์กับการกระทำทางคณิตศาสตร์ (ตัวดำเนินการบวก/ลบ)

การบวกตัวแปรพอยน์ตอร์ (ด้วยจำนวนเต็ม): หมายถึง การเลื่อนตัวชี้จากตำแหน่งปัจจุบันขึ้น (เลื่อนตำแหน่งในหน่วยความจำ) ไป K*N ไบต์

การลบค่าตัวแปรพอยน์ตอร์ (ด้วยจำนวนเต็ม) : หมายถึง การเลื่อนตัวชี้จากตำแหน่งปัจจุบันลง (เลื่อนตำแหน่งในหน่วยความจำ) ไป K*N ไบต์

K คือ ตัวแปร หรือ ค่าคงที่ (ชนิดจำนวนเต็มเท่านั้น)
N คือ ขนาดของแต่ละชนิดตัวแปรที่พอยน์เตอร์นั้นชี้

3. Pointer Arithmetic & Comparison

ตัวแปรพอยน์เตอร์กับการกระทำทางคณิตศาสตร์ (ตัวดำเนินการบวก/ลบ)

ตัวอย่าง 8

```
float *ptr=(float *)4000;
printf("Initial Address : %u\n",ptr);
ptr-=1;
printf("New Value of ptr:%u",ptr);
ptr+=8;
printf("New Value of ptr:%u",ptr);
```

```
Initial Address: 4000
New Value of ptr : 3996
New Value of ptr : 4028
```

3996 + 8*4

New address = (address) + **(number*size of data type)**

3. Pointer Arithmetic & Comparison

ตัวแปรพอยน์เตอร์กับการกระทำทางคณิตศาสตร์

ตัวอย่าง 9

```
int num , *ptr1 ,*ptr2 ;
ptr1 = &num ;
ptr2 = ptr1 + 2 ;
```

	Value of **Ptr1**	Value of **Ptr2**
	ค่าขยะ	ค่าขยะ
	1000	ค่าขยะ
	1000	1008

```
printf("Value of ptr2:%u\n",ptr2);
printf("%d",ptr2 - ptr1);
```

```
Value of ptr2 : 1008
2
```

(1008-1000)/sizeof(int)
= 8/4 =2

3. Pointer Arithmetic & Comparison

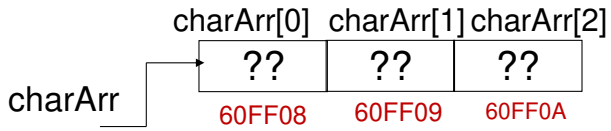
การเปรียบเทียบตัวแปรพอยน์เตอร์

นอกจากการกระทำทางคณิตศาสตร์ ตัวแปรพอยน์เตอร์ยังสามารถดำเนินการทางตรรกะได้ (เปรียบเทียบ) เช่นเดียวกับตัวแปรอื่นๆ

==	หมายถึง	ตำแหน่งเดียวกัน
!=	หมายถึง	ต่างตำแหน่งกัน
>	หมายถึง	ตำแหน่ง index สูงกว่า
>=	หมายถึง	ตำแหน่ง index สูงกว่าหรือเท่ากัน
<	หมายถึง	ตำแหน่ง index ต่ำกว่า
<=	หมายถึง	ตำแหน่ง index ต่ำกว่าหรือเท่ากัน

การเปรียบเทียบจะมีความหมายถูกต้องก็ต่อเมื่อพอยน์เตอร์ทั้งสองตัวชี้ที่อาเรย์ตัวเดียกัน

4. Pointer and Array

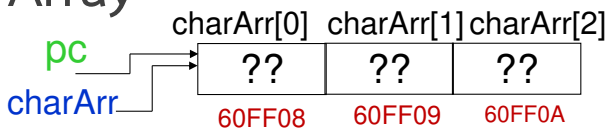


ชื่อตัวแปรอาเรย์ (Array identifier) จะมีค่าเทียบได้กับตำแหน่งแอดเดรสของอาเรย์ช่องแรกซึ่งเทียบได้กับตำแหน่งแรก

```
char charArr[3];
int i;

printf("Address of charArr[0] = %p\n", charArr);

for(i = 0; i < 3; ++i)
printf("Address of charArr[%d] = %p\n", i, &charArr[i]);
```

```
Address of charArr[0] = 0060FF08
Address of charArr[0] = 0060FF08
Address of charArr[1] = 0060FF09
Address of charArr[2] = 0060FF0A
```


4. Pointer and Array




```
char charArr[3];
char *pc;
printf("Address of charArr[0] = %p\n", charArr);
pc = charArr;
printf("Address of *pc = %p\n", pc);
```

หรือเขียนเป็น `pc =&charArr[0];`

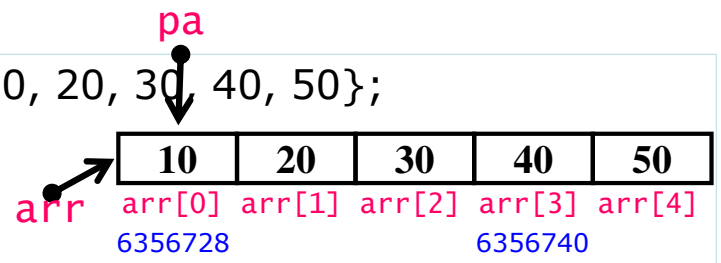
```
Address of charArr[0] = 0060FF08
Address of pc* = 0060FF08
```

เนื่องจาก**ชื่อตัวแปรอาเรย์** มีค่าเทียบได้กับตำแหน่งในหน่วยความจำของสมาชิกตัวแรกของอาเรย์ (ซึ่งเทียบได้กับตำแหน่งแรก --Index 0)

--> **การกำหนดพอยเตอร์ให้ชี้อาเรย์โดยระบุชื่ออาเรย์** จะมีค่าเท่ากับให้พอนย์เตอร์ชี้ที่**สมาชิกตัวแรกของอาเรย์**

4. Pointer and Array

ตัวอย่าง 10




```
int arr[] = {10, 20, 30, 40, 50};
int *pa=arr;
if(pa==arr)
  printf("Address of pa is equivalent to arr\n");
pa+=3;
if(pa>arr)
printf("pa now points at %u\n,arr point at %u", pa,arr);
```

การเปรียบเทียบพอยน์เตอร์

```
Address of pa is equivalent to arr
pa now points at 6356740,
arr point at 6356728
```

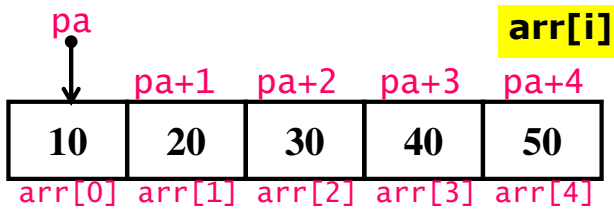

4. Pointer and Array

อาเรย์มีความเกี่ยวข้องกับตัวแปรพอยเตอร์อย่างใกล้ชิด การกำหนดให้พอยน์เตอร์ไปชี้อาเรย์ เสมือนว่า ตัวแปรพอยน์เตอร์ และ อาเรย์นั้นเป็นข้อมูลชุดเดียวกัน

```
arr[0]=10, arr[3]=40
arr[0]=10, arr[3]=40
```

ตัวอย่าง 11

```
int arr[] = {10, 20, 30, 40, 50};
int *pa=&arr[0];
printf("arr[0]=%d, arr[3]=%d\n", arr[0],arr[3]);
printf("arr[0]=%d, arr[3]=%d\n", *pa,*(pa+3));
```



arr[i] มีค่าเท่ากับ ***(pa + i)**

Quick check3

- จงแสดงผลลัพธ์ทางจอภาพของโปรแกรมต่อไปนี้

num[0]	num[1]	num[2]	num[3]	num[4]
12	34	112	45	907
0410	0414	0418	0422	0426

```
int main(){
  int num[5]={12,34,112,45,907};
  int *pt_num, temp;

  pt_num=&num[4];
  temp=*pt_num;
  printf("temp=%d\n",temp);
  printf("%d\n", *--pt_num);
  printf("%u\n", pt_num);
}
```

temp	pt_num
0310	0230

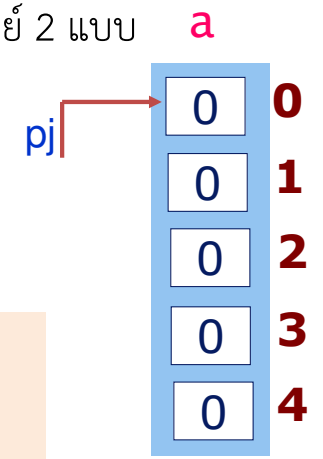
4. Pointer and Array

- การวนลูปเข้าถึงสมาชิกแต่ละตัวของอาเรย์ 2 แบบ

```
int j,n=5;
int a[5]={};
for(j = 0; j < n; j++)
    a[j]++;
```

```
int *pj;
for(pj = a; pj < a + n; pj++)
    (*pj)++;
```

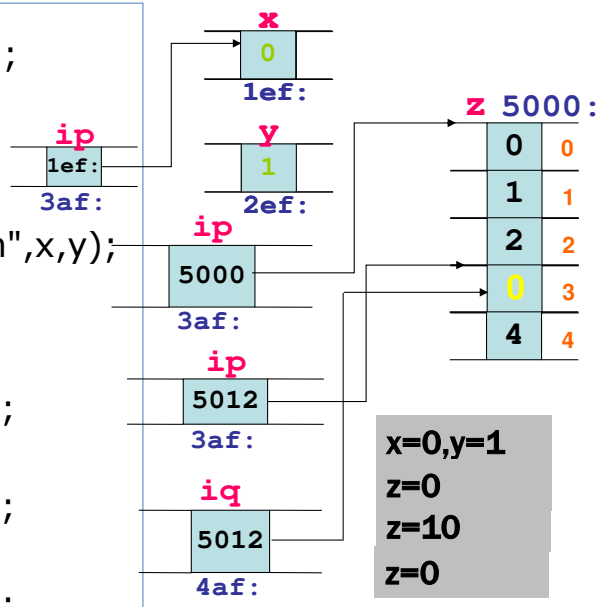
```
for(j = 0; j < n; j++)
    (*(a+j))++;
```



4. Pointer and Array

ตัวอย่าง 12

```
int x = 1, y = 2;
int z[5] = {0,1,2,3,4};
int *ip, *iq;

ip     = &x;
y      = *ip;
*ip    =  0;
printf("x=%d,y=%d\n",x,y);

ip     = &z[0];
ip     = ip + 3;
*ip    =  0;
printf("z=%d\n",z[3]);

*ip    = *ip + 10;
printf("z=%d\n",z[3]);
iq     = ip;
*iq = 0;
printf("z=%d\n",z[3]);
```



x=0,y=1
z=0
z=10
z=0

Summary

- พอยน์เตอร์ทำหน้าที่ชี้ไปยังตำแหน่งเก็บข้อมูลในหน่วยความจำ
- การประกาศพอยน์เตอร์ ต้องกำหนด data type ด้วย พร้อมทั้งมีเครื่องหมาย * หน้าชื่อตัวแปรพอยน์เตอร์

Operation	Description
1. Assignment (=)	การกำหนดค่าให้ตัวแปรพอยน์เตอร์ใช้เครื่องหมาย = ซึ่งค่าที่กำหนดต้องเป็นตำแหน่งในหน่วยความจำ (address)
2. Indirection (*)	การอ้างถึงค่าในตำแหน่งที่พอยน์เตอร์นั้นชี้ ใช้เครื่องหมาย * หน้าพอยน์เตอร์
3. Address of (&)	การอ้างถึงตำแหน่งของตัวแปรชนิดพื้นฐานใดๆ ใช้เครื่องหมาย & หน้าตัวแปร
4. Incrementing /Decrementing	เลื่อนพอยน์เตอร์ขึ้น/ลง ไปตามชนิดของตัวแปรที่พอยน์เตอร์นั้นชี้
5. Differencing	พอยน์เตอร์ 2 ตัวสามารถลบกันได้ ค่าที่ได้เป็นจำนวนเต็ม
6. Comparison	การเปรียบเทียบ พอยน์เตอร์ 2 ตัว จะมีความหมายก็ต่อเมื่อพอยน์เตอร์ทั้งสองชี้ที่อาเรย์ตัวเดียวกัน