บทที่ 3
ตัวชี้/พอยน์เตอร์ (Pointer)

สุนทรี คุ้มไพโรจน์

พอยน์เตอร์

- ชนิดข้อมูลชนิดหนึ่งของภาษาซี
- มีความเร็วในการทำงานสูง
- ช่วยให้การเขียนภาษาซีมีความยืดหยุ่น
- การใช้งานพอยน์เตอร์ค่อนข้างซับซ้อน
- พอยน์เตอร์เป็นจุดเด่นอย่างหนึ่งในการเขียนภาษาซี

พอยน์เตอร์กับแอดเดรส
(Pointers and Addresses)

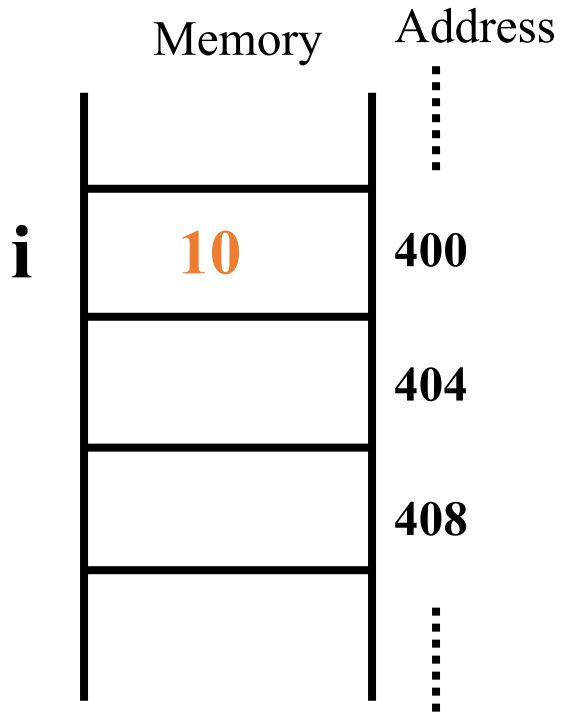
- **ตัวแปร (ชื่อที่ใช้แทนข้อมูล)**
- เราประกาศตัวแปรเป็นการกำหนดชื่อเพื่อใช้แทนข้อมูล
- เมื่อเราประกาศตัวแปร จะมีการจองเนื้อที่ในหน่วยความจำเพื่อเก็บข้อมูล เราสามารถเข้าถึงข้อมูลได้โดยอ้างถึงตัวแปร
- การประกาศตัวแปร เช่น

```
int  i;
```

- เป็นการประกาศ (Declaration)
  ตัวแปรชื่อ i เป็นตัวแปรประเภท int

- ภาพจำลองการแทนข้อมูลในหน่วยความจำ

```
int  i;

i = 10;
```



- แต่มีอีกวิธีที่จะเข้าถึงตัวแปร

  คือเราจะอ้างถึงตำแหน่งที่เก็บข้อมูล

- พอยน์เตอร์ ➢ชนิดข้อมูลชนิดหนึ่งของภาษาซี

  แตกต่างจากชนิดข้อมูลพื้นฐานอื่น ๆ

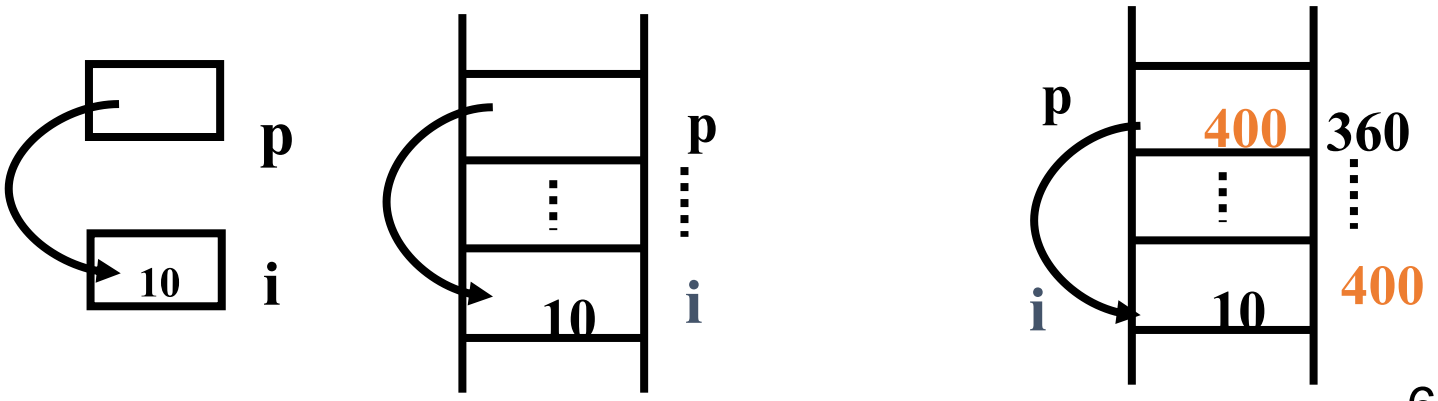
- ตัวแปรพอยน์เตอร์ ➢ตัวแปรที่ใช้เก็บค่าแอดเดรส

  ของตัวแปรชนิดอื่น ๆ

- หากมี ตัวแปร i เป็นตัวแปรประเภท int
- และ ตัวแปร p เป็นตัวแปรประเภทพอยน์เตอร์

  p เก็บค่าแอดเดรสของตัวแปร i

  (หรือ p ชี้ไปที่ตัวแปร i)
- สามารถจำลองการแทนข้อมูลในหน่วยความจำดังรูป



การประกาศตัวแปรพอยน์เตอร์

- ใช้การดำเนินการชนิดเอกภาค (Unary Operator) *
- ชื่อเรียกเป็นภาษาอังกฤษว่า Indirection

  หรือ Dereferencing Operator
- รูปแบบคำสั่ง Type *Variable-name;

Type ชนิดของตัวแปร

* เป็นเครื่องหมายที่แสดงว่า ตัวแปรที่ตามหลังเครื่องหมายนี้เป็นตัวแปรชนิดพอยน์เตอร์

Variable-name ชื่อตัวแปรที่เป็นตัวแปรพอยน์เตอร์

- ตัวแปรพอยน์เตอร์ เป็นตัวชี้ไปยังตัวแปรชนิดอื่น ๆ
- การประกาศชนิดของตัวแปรพอยน์เตอร์ต้องสอดคล้องกับชนิดของตัวแปรนั้น ๆ เช่น

```
char *prt;
int  *ip , *temp;
double  *dp;
```

ประกาศตัวแปร prt ให้เป็นตัวแปรพอยน์เตอร์ที่ชี้ไปยังตัวแปรชนิด chr

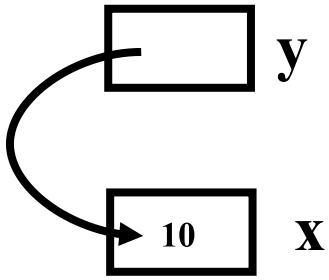
ประกาศตัวแปร ip และ ตัวแปร temp เป็นตัวแปรพอยน์เตอร์ที่ชี้ไปยังตัวแปรชนิด int

ประกาศตัวแปร dp เป็นตัวแปรพอยน์เตอร์ที่ชี้ไปยังตัวแปรชนิด double

การกำหนดค่าและการอ่านค่าตัวแปรพอยน์เตอร์

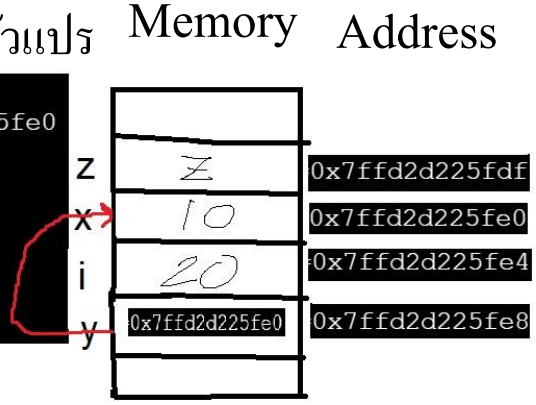
- การกำหนดค่าให้กับตัวแปรพอยน์เตอร์ เป็นการกำหนด แอดเดรส ของตัวแปรที่มีชนิดข้อมูลสอดคล้องกับชนิดข้อมูลของตัวแปรพอยน์เตอร์
- ใช้ตัวดำเนินการชนิดเอกภาค (Unary Operator) & เป็นตัวดำเนินการที่อ้างถึงแอดเดรส
- ตัวดำเนินการ & เป็นเครื่องหมายที่ใช้เมื่อต้องการให้เอาตำแหน่งที่อยู่ (address) ของตัวแปรที่เก็บในหน่วยความจำออกมาใช้
- ตัวดำเนินการ * เป็นเครื่องหมายที่ใช้เมื่อต้องการให้นำค่าที่เก็บในตำแหน่งที่ตัวแปรพอยน์เตอร์นั้นชี้อยู่ออกมาแสดง

- **ตัวอย่าง** int x=10; //ประกาศตัวแปร x เป็นชนิด integer
- int *y; //ประกาศตัวแปรพอยน์เตอร์ y ให้เป็นชนิด integer
- // ให้สอดคล้องกับตัวแปรที่ต้องการชี้
- y = &x; //กำหนดให้พอยน์เตอร์ y ชี้ไปยัง
- //ตำแหน่งแอดเดรสของตัวแปร x

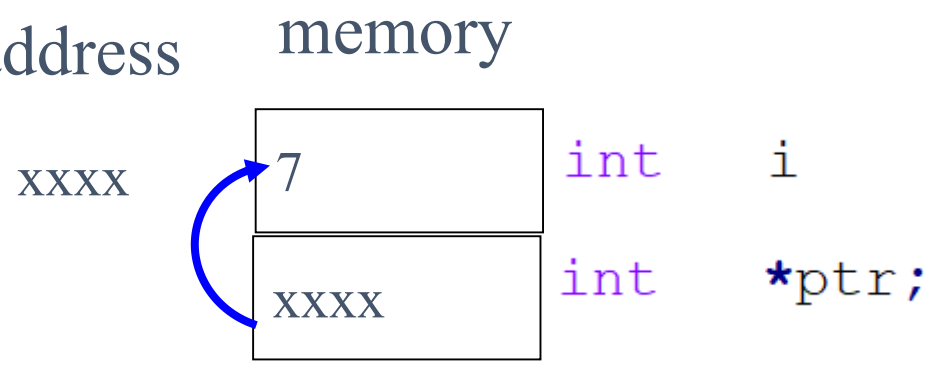



```c
#include <stdio.h>
int main() {
    int   x=10;
    int   *y;
    char  z='Z';
    int   i=20;
    y  = &x;
    printf("Address of x      =%p, value of x=%d\n",&x,x);
    printf("Address of y      =%p, value of y=%p\n",&y,y);
    printf("Address of z      =%p, value of z=%c\n",&z,z);
    printf("Address of i      =%p, value of i=%d\n",&i,i);

    return 0;
}
```

```
Address of x      =0x7ffd2d225fe0, value of x=10
Address of y      =0x7ffd2d225fe8, value of y=0x7ffd2d225fe0
Address of z      =0x7ffd2d225fdf, value of z=Z
Address of i      =0x7ffd2d225fe4, value of i=20


...Program finished with exit code 0
Press ENTER to exit console.
```



ตัวอย่างการอ้างถึง(L32.c)



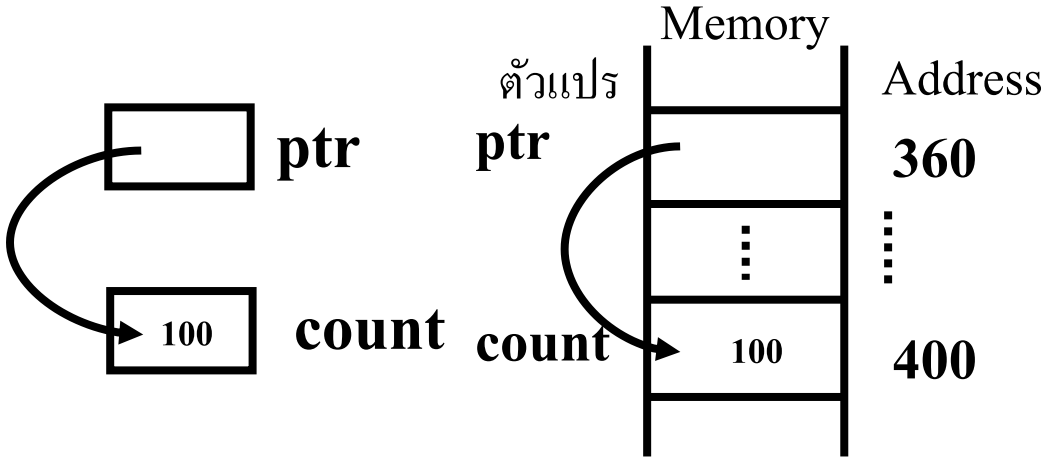

```c
#include <stdio.h>
int main() {
    int   i   =  7;
    int   j;
    int   k[10];
    int   *ptr;   // ptr เป็นตัวชี้ข้อมูล int
    k[0]  = 1;
    ptr   = &i;   // ptr ชี้ไปที่ i
    printf("Address of i =%p, value of i=%d\n",ptr,i);
    j     = *ptr;
    printf("j =%d\n",j);
    *ptr  = 0;
    printf("value of ptr=%p, value of *ptr =%d\n",ptr,*ptr);
    ptr   = &k[0];
    printf("value of ptr=%p, value of *ptr =%d\n",ptr,*ptr);
    return 0;
}
```

ตัวอย่าง ให้เขียนแผนภาพ ตัวแปร, memory , address และ การชี้

สมมติ address เริ่มต้นที่ 400

```
int count , val ,* ptr;
count =100;
ptr = &count;
val = *ptr;
```

ณ ตอนนี้ val มีค่าเท่าไร



ตัวอย่าง ให้เขียนแผนภาพ ตัวแปร, memory , address และ การชี้

```
int    a , *prt , b , c  ,*d;
a = 25;
prt = &a;
b = a;
c = *prt;
d= prt;
```

สมมติ address เริ่มต้นที่ 400

ตัวอย่าง ให้เขียนแผนภาพ ตัวแปร, memory , address และ การชี้

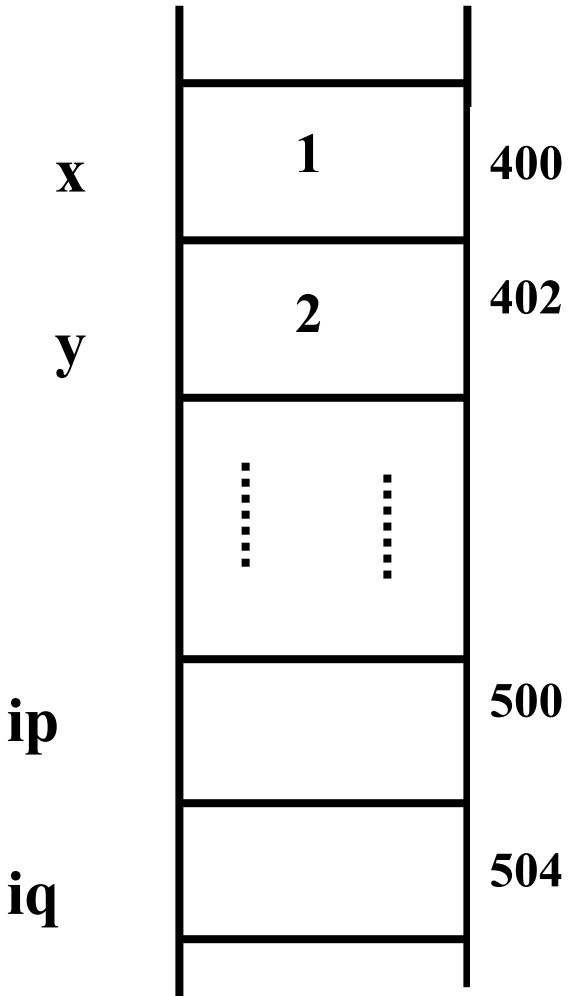
```
int  x = 1, y = 2;
int  *ip, *iq;
ip    =  &x;
y     =  *ip;
*ip   =  0;
y     = 5;
ip    =  &y;
*ip   =  3;
iq    = ip;
```

กำหนดให้ **x** และ **y** เป็นตัวแปรชนิด **int** เก็บค่า 1 และ 2 ตามลำดับ

ip และ **iq** เป็นตัวแปรพอยน์เตอร์ ซึ่งชี้ไปที่ชนิดข้อมูล **int**

1 **int x =1, y =2;**

2 **int *ip, *iq;**



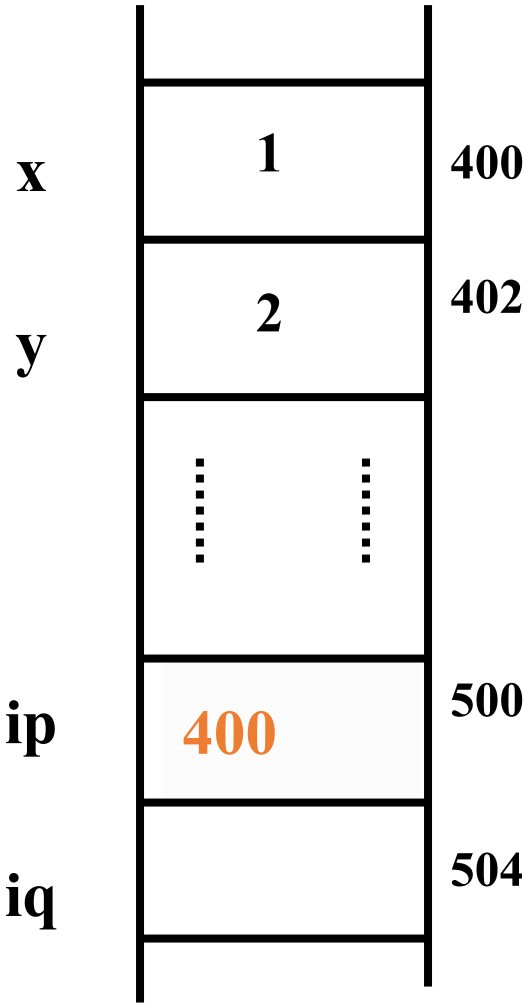
3

```
ip   =  &x;
```

ให้ ip ชี้ไปที่ x

ดังนั้น ที่ ip เก็บaddress ที่ ip ชี้ไป

address ที่ ip ชี้ไป คือ 400

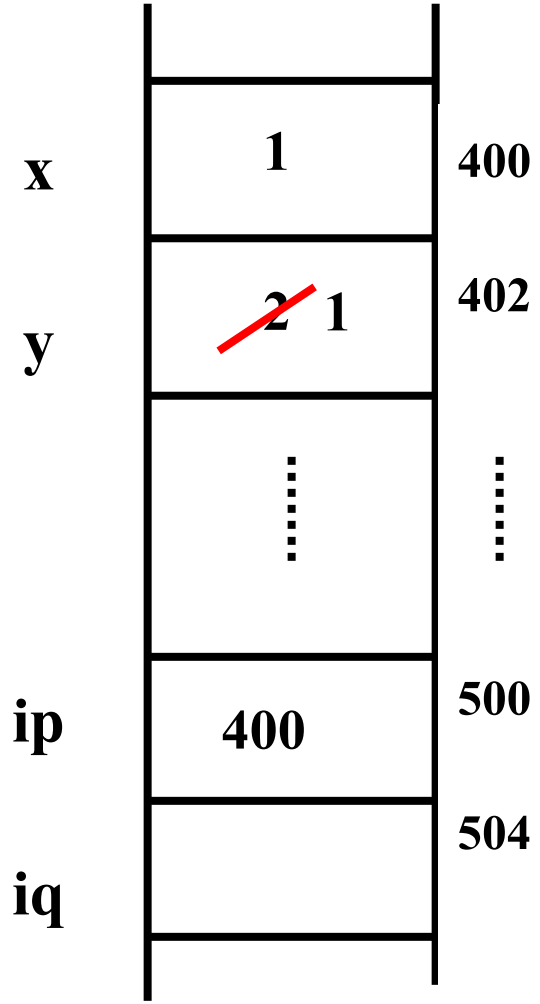


4 y = *ip;

y = ค่า ที่ addrerss นี้ point ไป

นั่นคือ ip ชี้ ที่ 400

ค่าที่ตำแหน่ง 400 คือ 1

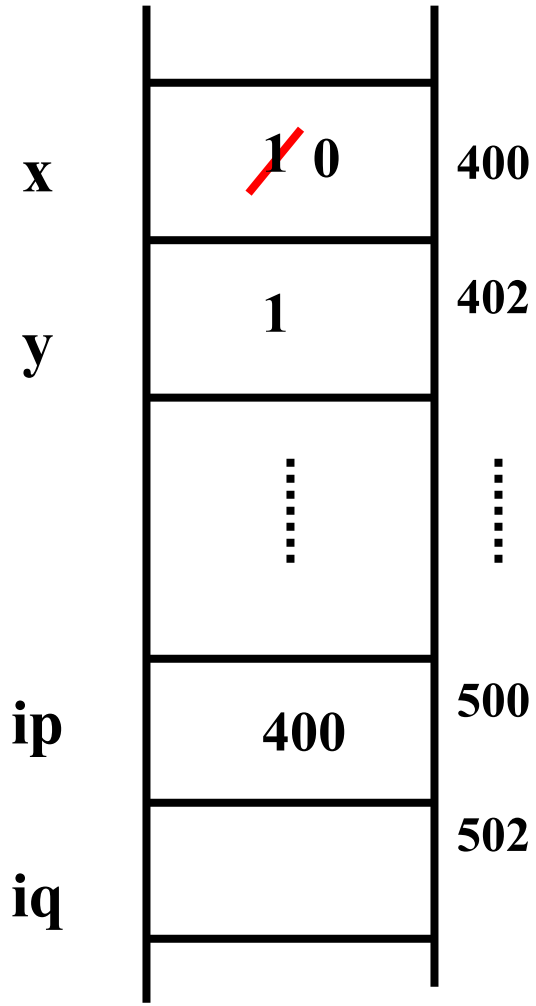


5 *ip = 0;

กำหนดให้ ค่า ที่ addrerss นี้ชี้ไปมีค่า เท่ากับ 0

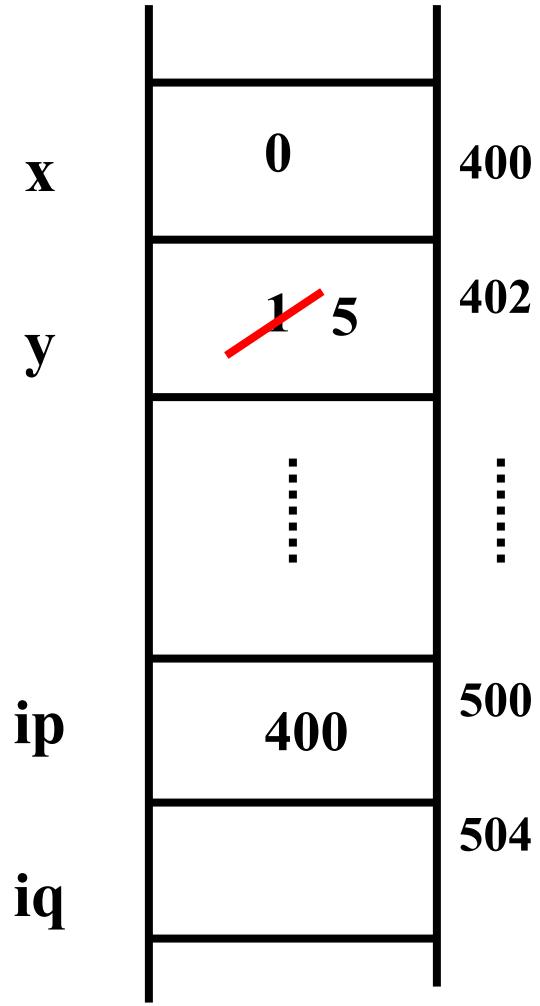
นั่นคือ ip ชี้ ที่ 400

ค่าที่ตำแหน่ง 400 คือ 1 เปลี่ยนเป็น 0



6

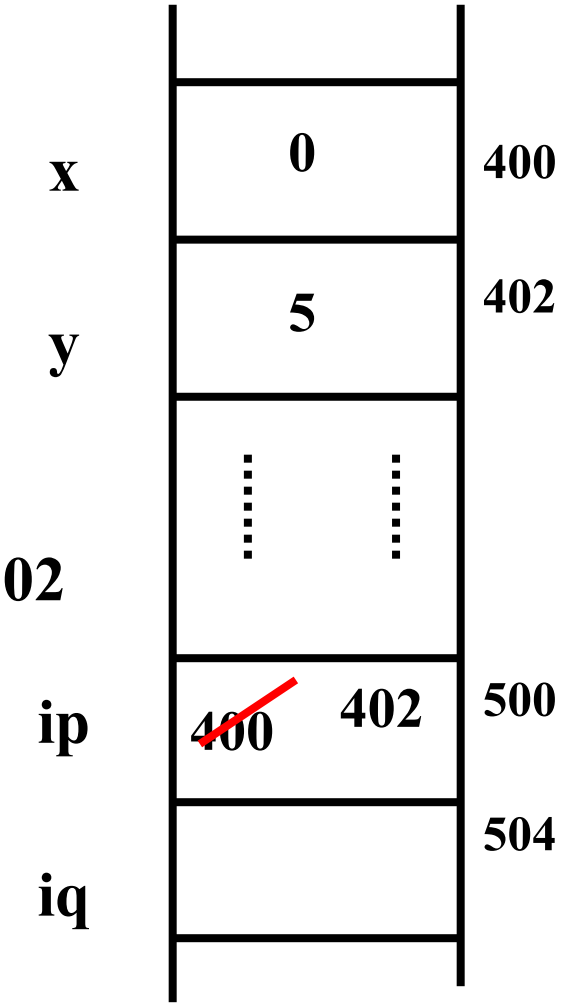
y = 5;



7 ip = &y;

ให้ ipชี้ ไปที่ y

ดังนั้น ที่ ip จึงเก็บ address ของ y คือ 402

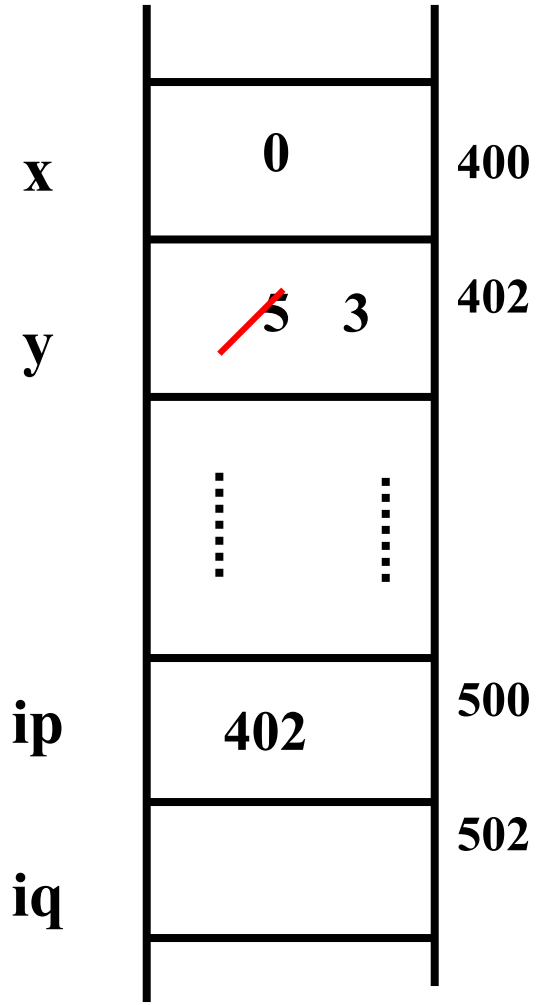


8 *ip = 3;

กำหนดให้ ค่า ที่ addrerss นี้ชี้ไป มีค่า เท่ากับ 3

นั่นคือ ip ชี้ ที่ 402

ค่าที่ตำแหน่ง 402 คือ 5 เปลี่ยนเป็น 3

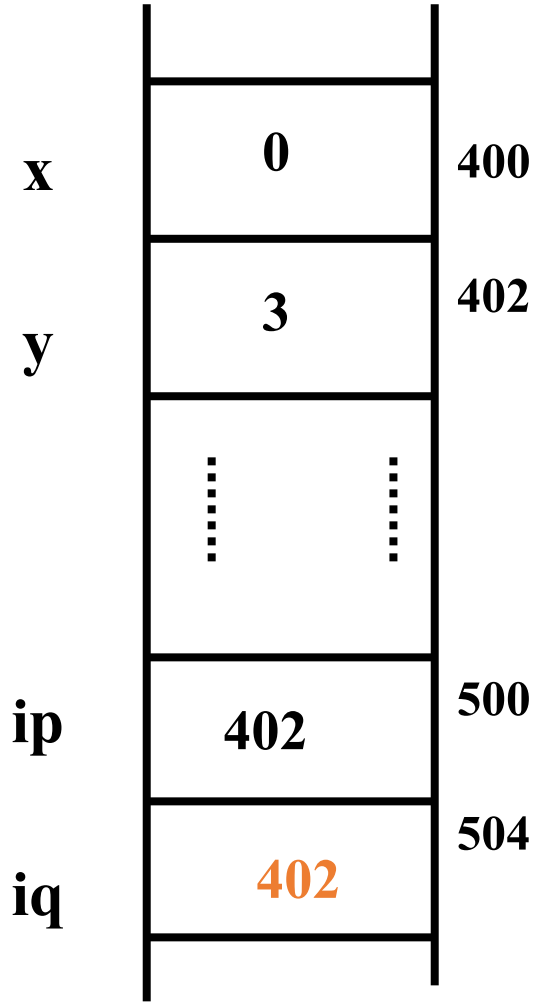


9 iq = ip;

เป็นนำค่าaddress ที่ เก็บใน ip ให้กับ iq

ดังนั้น iq จึงมีค่า 402

หมายความว่า iq ชี้ ไปที่ y



การแสดงค่า address

- ฟังก์ชัน printf สามารถแสดง address ของข้อมูลได้โดยใช้
  - รูปแบบ %p เพื่อแสดงตำแหน่งที่อยู่เป็นเลขฐาน 16
  - รูปแบบ %u เพื่อแสดงตำแหน่งที่อยู่เป็นเลขฐาน 10
- ผลลัพธ์ที่ได้จะอยู่ในรูปแบบ xxxx:yyyy หรือ XXXX ขึ้นอยู่กับ memory model ที่ใช้

ตัวอย่างการแสดงค่าaddress (L31.c)

```c
#include <stdio.h>
#include <string.h>
int main() {
    char  a   = 'A';
    int   i   =  15;
    float f   =  7.5;

    char  *pt_a;
    int   *pt_i;
    float *pt_f;

    pt_a = &a;
    pt_i = &i;
    pt_f = &f;

    printf("Address of a =%p, value of a=%c\n",pt_a,a);
    printf("Address of i =%p, value of i=%d\n",pt_i,i);
    printf("Address of f =%p, value of f=%f\n",pt_f,f);

    return 0;
}
```

ตัวอย่างโปรแกรม (L33.c)

```c
#include <stdio.h>
int main() {
    int   i = 7, j;
    float f = 2.5;
    char  c = 'C', d;

    int   *ptr_i;
    float *ptr_f;
    char  *ptr_c;

    ptr_i = &i;
    ptr_f = &f;
    ptr_c = &c;

    j     = *ptr_i;
    d     = *ptr_c;
    printf("value of i =%d\n",i);
    printf("value of f =%f\n",f);
    printf("value of d =%f\n",d);

    return 0;
}
```

- ให้เขียนแผนภาพการจัดเก็บข้อมูล ใน memory
- และเขียนการชี้ของ pointer

ตัวอย่างโปรแกรม (L34.c)

```c
#include <stdio.h>
#include <conio.h>
int main() {
    int   i;
    int   *ptr_i;

    i = 3;
    ptr_i = &i;

    printf("Address of i   =%p\n",&i);
    printf("value of ptr_i =%p\n",ptr_i);
    printf("value of i     =%d\n",i);
    printf("value of *ptr_i=%d\n",*ptr_i);
    printf("Comparing the variable belows\n");
    printf("&*ptr_i =%p \n",&*ptr_i);
    printf("*&ptr_i =%p \n",&*ptr_i);

    return 0;
}
```

Pointer ซ้อน pointer

```
type **ptt_name;
```

Type	คือ ชนิดของตัวแปร พอยน์เตอร์
**	คือ เครื่องหมายที่แสดงว่าเป็นตัวแปร พอยน์เตอร์ซ้อนพอยน์เตอร์
ptt_name	คือ ชื่อของตัวแปร พอยน์เตอร์ซ้อนพอยน์เตอร์

ตัวอย่างโปรแกรม (L35.c)

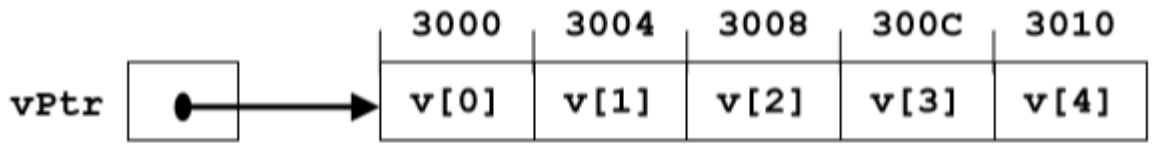
```c
#include <stdio.h>
#include <conio.h>
int main() {
    int   i=7;
    int   *ptr_i;
    int   **pptr_i;

    ptr_i  = &i;
    pptr_i = &ptr_i;

    int   j = *ptr_i;
    int   k = **pptr_i;

    printf("Address of i       =%p, value of i=%d\n",&i,i);
    printf("Address of ptr_i   =%p, value of ptr_i =%p\n",&ptr_i,ptr_i);
    printf("Address of pptr_i =%p, value of pptr_i=%p\n",&pptr_i,pptr_i);
    printf("j =%d, k=%d\n",j,k);

    return 0;
}
```

Pointer vs. Array

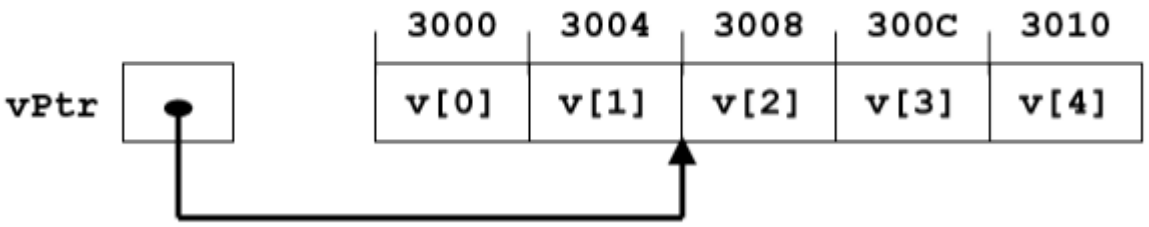
- Pointer และ array ในภาษา C นั้น มีความใกล้ชิดกันอย่างมาก
- ดูตัวอย่างการประกาศใช้ array และ pointer
- และการกำหนด pointer ให้ชี้ไปยังตัวแปร array ได้ตามตัวอย่าง

```c
#include <stdio.h>
int main() {
    int   i;
    int   a[4] = {0,1,2,3};
    int   *ptr_a1;
    int   *ptr_a2;

    ptr_a1  = a;
    ptr_a2  = &a[0];
    for (i=0; i<4; i++){
       printf("*ptr_a1 =%d, *ptr_a2=%d,",*ptr_a1,*ptr_a2);
       ptr_a1++;
       ptr_a2++;
       printf("  a[%d] = %d \n",i,a[i]);
    }
    return 0;
}
```

Pointer vs. Array



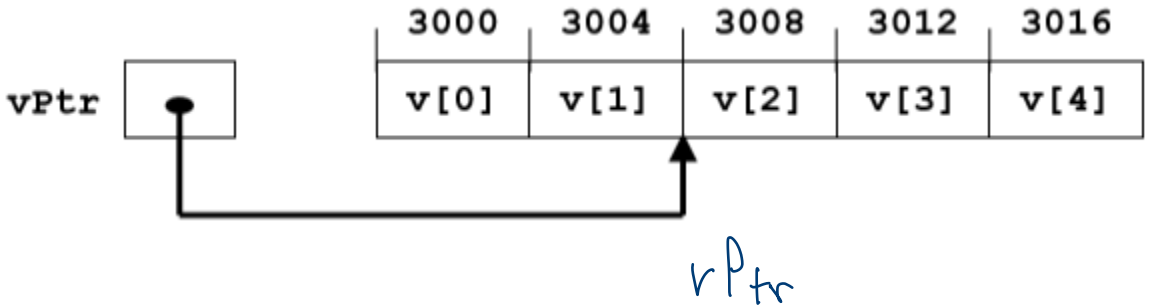
- เมื่อบวกหรือลบจำนวนเต็มกับตัวชี้แล้ว
  - ค่าของตัวชี้**มิได้เพิ่มหรือลดลงตามตัวเลขจำนวน**นั้น
  - ค่าของตัวชี้**เพิ่มหรือลดตามตัวเลขจำนวนนั้นคูณกับขนาดของวัตถุที่ตัวชี้นั้นชี้อยู่**
  - ขนาด (ไบท์) ขึ้นกับประเภทของข้อมูลที่ใช้ในวัตถุนั้น
- ตัวอย่าง (กำหนดขนาดของวัตถุ ชนิด float คือ 4 ไบท์)
  - `vPtr += 2;  //  vPtr = vPtr+(2*4)`
    `// or vPtr = 3000+(2*4)`
  - หลังจากคำสั่งข้างต้นแล้ว `vPtr` จะชี้ไปที่ `v[2]`



ที่มา [8] สไลด์ วิชา 90102003 Computer and Programming ภาควิชาวิศวกรรมคอมพิวเตอร์ คณะวิศวกรรมศาสตร์ สถาบันเทคโนโลยีพระจอมเกล้าเจ้าคุณทหารลาดกระบัง http://www.ce.kmitl.ac.th

Pointer vs. Array

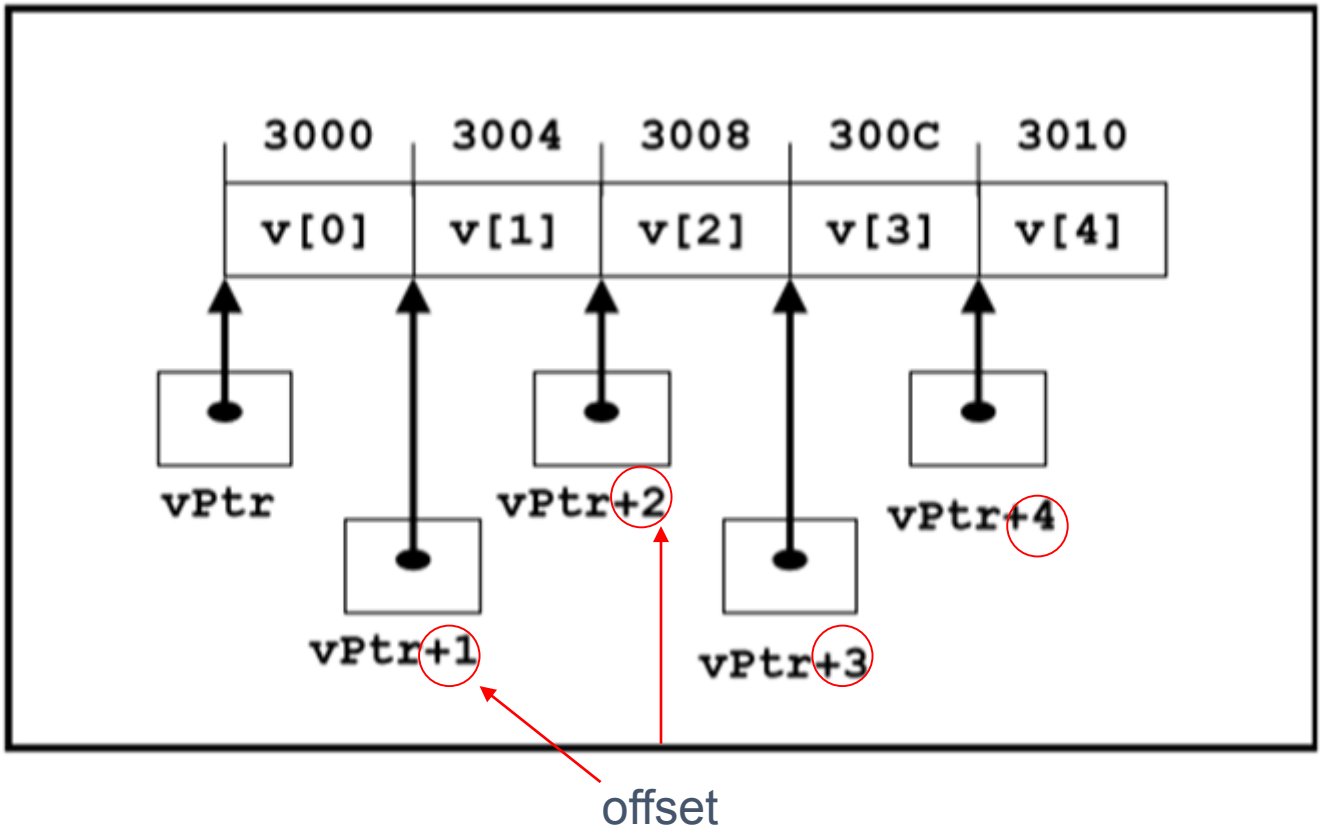
- สมมติ **pointer vPtr** ปัจจุบันชี้ที่หน่วยความจำดังภาพนี้



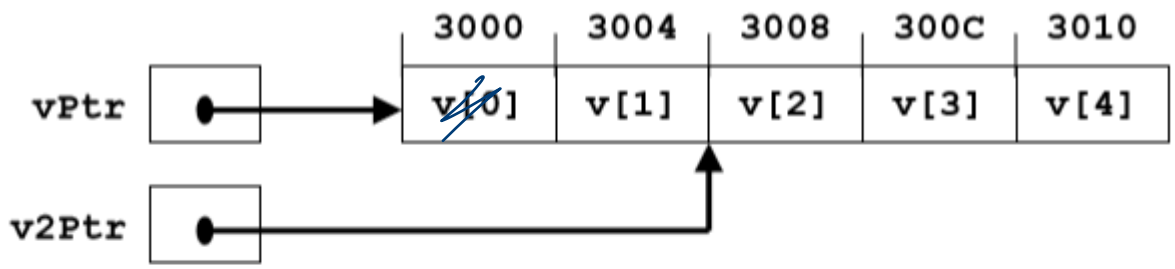
- ให้เขียนแผนภาพการชี้ เมื่อใช้คำสั่ง

```
vPtr -= 2;
```

ที่มา [8] สไลด์ วิชา 90102003 Computer and Programming ภาควิชาวิศวกรรมคอมพิวเตอร์ คณะวิศวกรรมศาสตร์ สถาบันเทคโนโลยีพระจอมเกล้าเจ้าคุณทหารลาดกระบัง http://www.ce.kmitl.ac.th



ที่มา [8] สไลด์ วิชา 90102003 Computer and Programming ภาควิชาวิศวกรรมคอมพิวเตอร์ คณะวิศวกรรมศาสตร์ สถาบันเทคโนโลยีพระจอมเกล้าเจ้าคุณทหารลาดกระบัง http://www.ce.kmitl.ac.th




```
• vPtr = &v[0];        //     vPtr = 3000
• v2Ptr = &v[2];       // or v2Ptr = 3008
• x = v2Ptr - vPtr;    // x = ?
```

ค่าที่ **x** ได้รับคือจำนวนหน่วย (element) ของตัวแปรแถวลำดับนับจาก **vPtr** ถึง **vPtr2** ในกรณีนี้คือ **2**

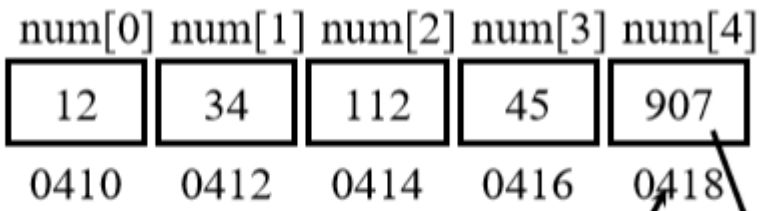
ที่มา [8] สไลด์ วิชา 90102003 Computer and Programming ภาควิชาวิศวกรรมคอมพิวเตอร์ คณะวิศวกรรมศาสตร์ สถาบันเทคโนโลยีพระจอมเกล้าเจ้าคุณทหารลาดกระบัง http://www.ce.kmitl.ac.th

- เห็นได้ว่าตัวชี้และตัวแปรแถวลำดับมีความสัมพันธ์กัน และสามารถใช้แทนกันได้เกือบทุกกรณี

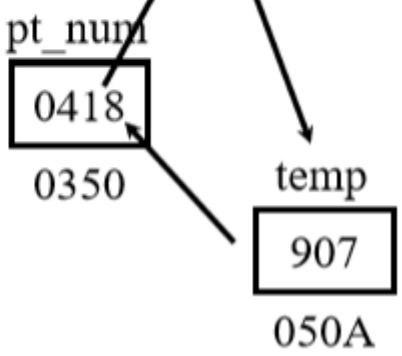
```
int  b[5];
int *bPtr;
bPtr = b;  //equivalent to bPtr = &b[0];

&b[3] equivalent to bPtr+3
b[3] equivalent to *(bPtr+3)
```

ที่มา [8] สไลด์ วิชา 90102003 Computer and Programming ภาควิชาวิศวกรรมคอมพิวเตอร์ คณะวิศวกรรมศาสตร์ สถาบันเทคโนโลยีพระจอมเกล้าเจ้าคุณทหารลาดกระบัง http://www.ce.kmitl.ac.th

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

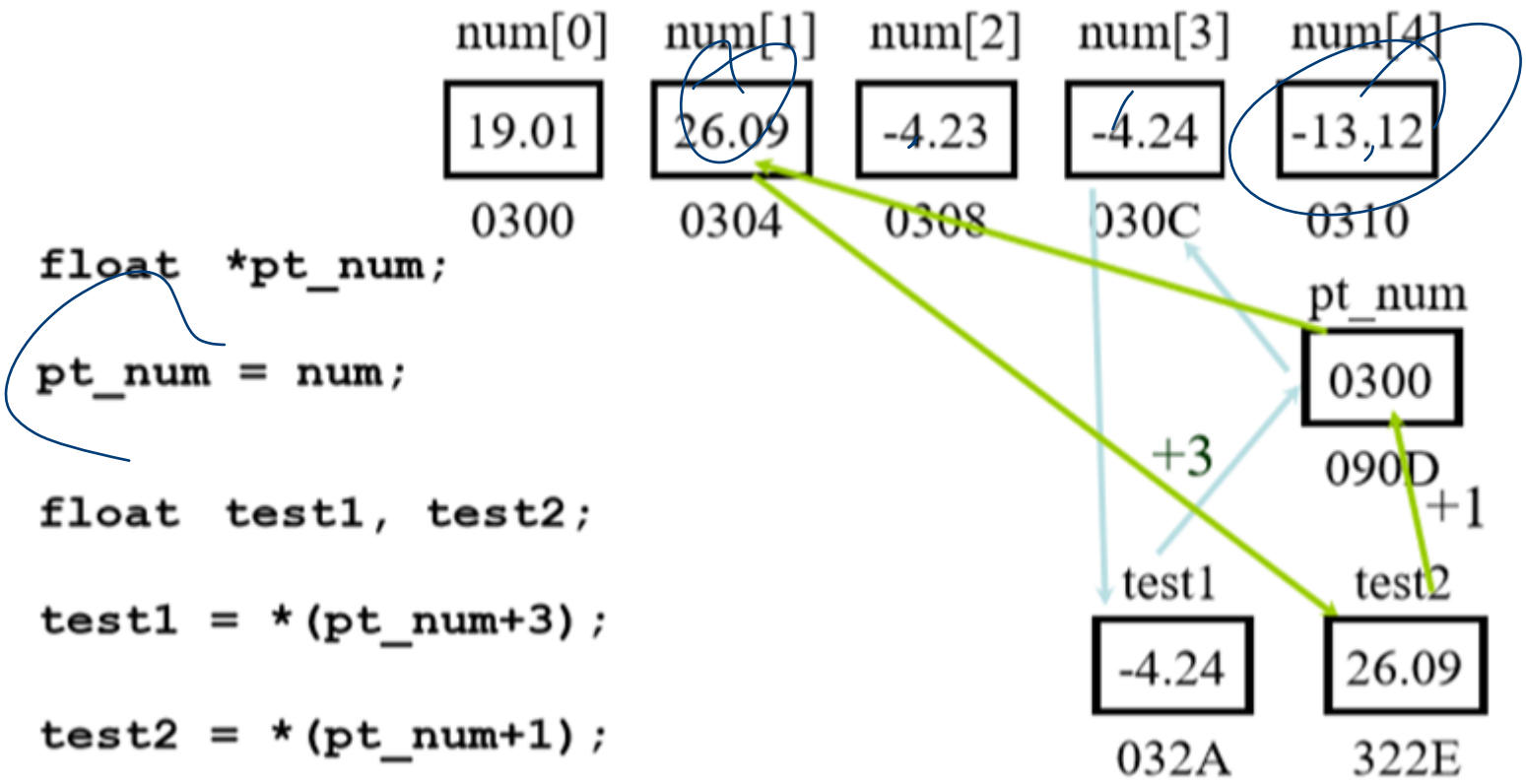



```
pt_num = &num[1];
pt_num = &num[4];

int  temp;
temp = *pt_num;
```



ที่มา [8] สไลด์ วิชา 90102003 Computer and Programming ภาควิชาวิศวกรรมคอมพิวเตอร์ คณะวิศวกรรมศาสตร์ สถาบันเทคโนโลยีพระจอมเกล้าเจ้าคุณทหารลาดกระบัง http://www.ce.kmitl.ac.th

```
float num[] = {19.01, 26.09, -4.23, -4.24, -13.12};

float  *pt_num;

pt_num = num;

float  test1, test2;

test1 = *(pt_num+3);

test2 = *(pt_num+1);
```



ที่มา [8] สไลด์ วิชา 90102003 Computer and Programming ภาควิชาวิศวกรรมคอมพิวเตอร์ คณะวิศวกรรมศาสตร์ สถาบันเทคโนโลยีพระจอมเกล้าเจ้าคุณทหารลาดกระบัง http://www.ce.kmitl.ac.th

เปรียบเทียบ pointer กับ array

L36.c

```c
#include <stdio.h>

int main(){
    int i, offset;
    int a[]= {10,20,30,40};
    int *ptr_a = a;

    for (i = 0; i < 4; i++){
        printf("a[%d]   = %d\n", i, a[i] );
        printf("*(a+%d)= %d\n", i,*(a+i));
        printf("ptr_a[%d]   = %d\n", i, ptr_a[i] );
        printf("*(ptr_a+%d)   = %d\n\n", i, *(ptr_a+i));
    }
    return 0;
}
```

Exercise พิมพ์ค่าที่ได้จากการ run program นี้

```
#include <stdio.h>
void main( ) {
        int j, k, *intPtr;
        k=2;
        intPtr = &k;
        printf("k is stored at %p. \n",&k);
        printf("value of  k is %d. \n",k);
        *intPtr += 5;
        printf("value of  k is %d. \n",k);

        j = *intPtr;

        j++;

        printf("value of  j is %d. \n",j);

        printf("value of  k is %d. \n",k);

        printf("value of  *intPtr is %d. \n",*intPtr);

}
```

ตัวอย่างการใช้ references ในภาษา C++

```
// Demonstrates the definition and use of references.
// ------------------------------------------------
#include <iostream>
#include <string>
using namespace std;

float x = 10.7F;                                  // Global

int main()
{
   float  &rx = x;                 // Local reference to x
// double &ref = x;                // Error: different type!

   rx *= 2;
   cout << "   x = " <<  x << endl    //  x = 21.4
        << "  rx = " << rx << endl;   // rx = 21.4
   const float& cref = x;     // Read-only reference
   cout << "cref = " <<  cref << endl;   // ok!
// ++cref;                          // Error: read-only!
   const string str = "I am a constant string!";
// str = "That doesn't work!";  // Error: str constant!
// string& text = str;          // Error: str constant!
   const string& text = str;    // ok!
   cout << text << endl;        // ok! Just reading.
   return 0;
}
```

ตัวอย่างการใช้ pointer ในภาษา C++

```
// pointer1.cpp
// Prints the values and addresses of variables.
// ---------------------------------------------
#include <iostream>
using namespace std;

int var, *ptr;     // Definition of variables var and ptr

int main()         // Outputs the values and addresses
{                  // of the variables var and ptr.
   var = 100;
   ptr = &var;

   cout << " Value of var:      " <<  var
        << "   Address of var: " <<  &var
        << endl;
   cout << " Value of ptr: "      <<  ptr
        << "   Address of ptr: " <<  &ptr
        << endl;
   return 0;
}
```

Refereces

www.cs.science.cmu.ac.th/course/comp105/slide/slidec5.ppt

www.ce.kmitl.ac.th