ตัวแปรชนิดพอยเตอร์ (Pointer)

พอยเตอร์ (Pointer)

* **พอยเตอร์ (Pointer)** เป็นตัวแปรชนิดพิเศษในภาษา C ทำหน้าที่เก็บตำแหน่งที่อยู่(Address) ของตัวแปรชนิดอื่นๆ ที่อยู่ในหน่วยความจำแทนที่จะเก็บข้อมูลเหมือนกันตัวแปรพื้นฐานชนิดอื่นๆ
* ตัวแปรพอยเตอร์มีลักษณะคล้ายตัวแปรตารางอาเรย์ แต่ที่แตกต่างกันคือ ตัวแปรตารางอาเรย์จะเก็บเฉพาะค่าต่างๆ ที่เป็นชนิดกันเดียวกับตัวแปรอาเรย์ แต่ตัวแปรพอยเตอร์จะเก็บเฉพาะค่าตำแหน่ง Address ตัวแปรเท่านั้น

การประกาศตัวแปรพอยน์เตอร์ (Pointer)

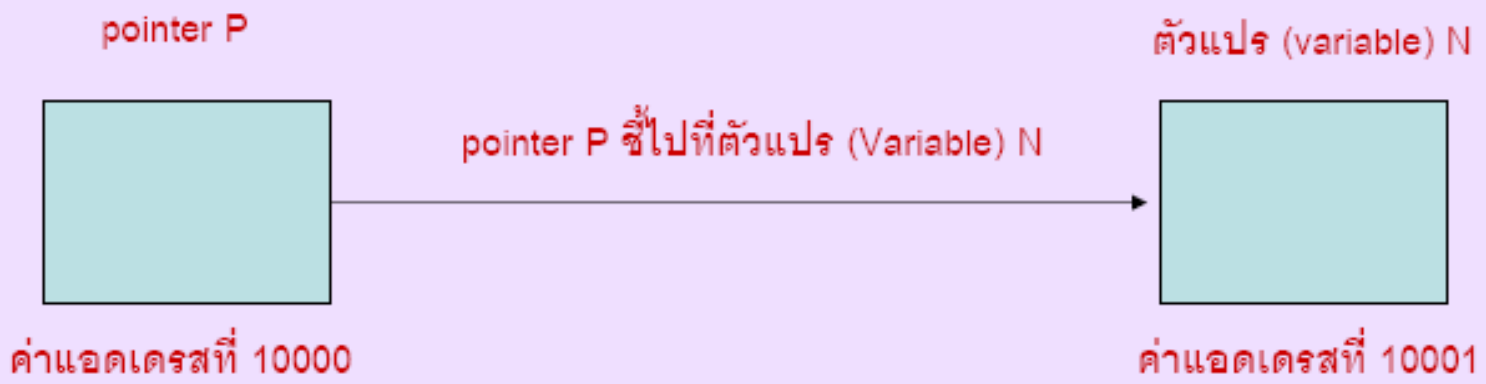
- รูปแบบ

Type * Variable Name

- Type หมายถึง ชนิดข้อมูล
- * Variable Name หมายถึง ตัวแปรพอยน์เตอร์ (pointer) จะมีเครื่องหมายดอกจันทร์ * (Asterisk) นำหน้าตัวแปร

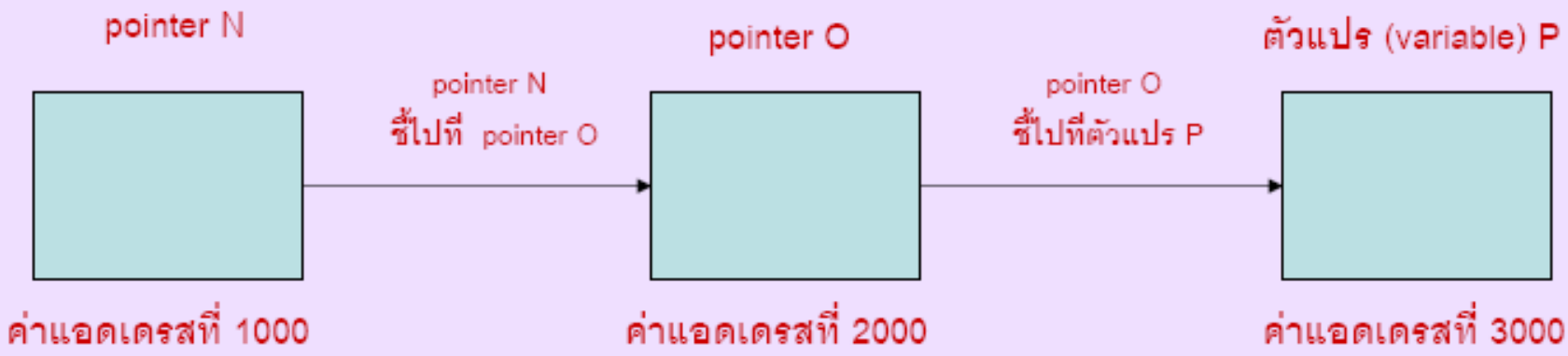
ประเภทของตัวแปรพอยน์เตอร์ (Pointer)

1. Direct Pointer หมายถึง การนำตัวแปรพอยน์เตอร์ (pointer) ไปชี้ค่าที่อยู่ (address) ของตัวแปร ตามรูปต่อไปนี้



ประเภทของตัวแปรพอยน์เตอร์ (Pointer)

2. Indirect Pointer หมายถึง การนำตัวแปรพอยน์เตอร์ (pointer) ตัวหนึ่งชี้ไปยังตัวแปรพอยน์เตอร์อีกตัวหนึ่ง ที่เรียกว่า Pointer to Pointer ตามรูปต่อไปนี้



การใช้งานพอยเตอร์ (Pointer)

ตัวอย่างการเขียนคำสั่งเพื่อประกาศตัวแปร pointer

- **int x;** สร้างตัวแปรชนิด int ชื่อ x สำหรับเก็บค่าจำนวนเต็ม
- **int *pt_x** สร้างตัวแปร pointer ชนิด int ทำให้ pt_x ใช้สำหรับเก็บตำแหน่งที่อยู่ของตัวแปรชนิด int เท่านั้น
- **float *pt_num** สร้างตัวแปร pointer ชนิด float ทำให้ pt_num ใช้สำหรับเก็บตำแหน่งที่อยู่ของตัวแปรชนิด float เท่านั้น
- **char *pt_ch** สร้างตัวแปร pointer ชนิด char ทำให้ pt_ch ใช้สำหรับเก็บตำแหน่งที่อยู่ของตัวแปรชนิด char เท่านั้น

ประเภทของตัวแปรพอยน์เตอร์ (Pointer)

- ตามปกติ เมื่อเราประกาศค่าตัวแปร คอมไพเลอร์ (compiler) จะจองเนื้อที่หน่วยความจำให้กับตัวแปรนั้น โดยแต่ละค่าตัวแปร จะมีค่าที่อยู่ (address) ของตัวแปรแต่ละตัว สำหรับจัดเก็บข้อมูล
- เช่น

```
int     score = 90
char    name = "ABC"
```

สมมติว่า ตัวแปร score มีค่าที่อยู่ (address) ที่ 1000 เก็บข้อมูล 90 และตัวแปร name มีค่าที่อยู่ 1001 เก็บข้อมูล ABC สามารถแสดงค่าตารางได้ดังนี้

ตัวแปร / ค่าที่อยู่	1000	1001	1002	1003
Score	90			
Name		ABC		

ตัวดำเนินการที่ใช้กับตัวแปรพอยเตอร์

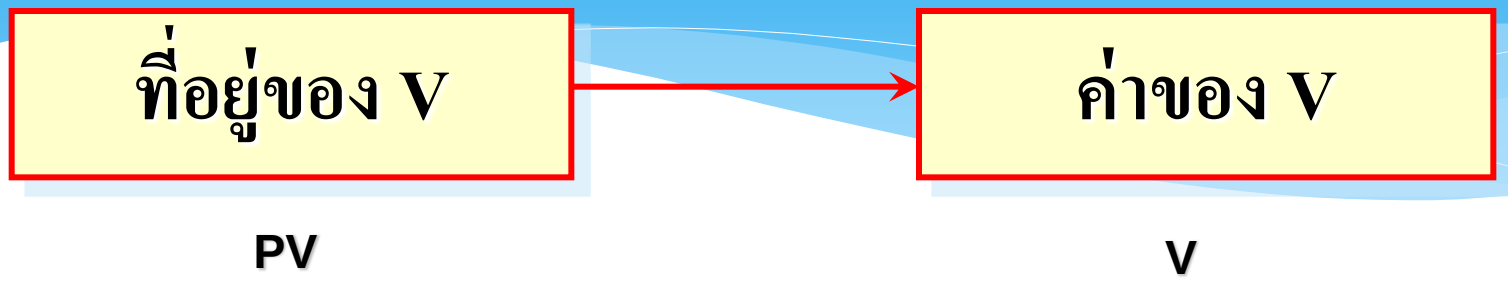
1. แสดงตำแหน่งข้อมูลด้วย & (address operation)

* เครื่องหมาย & (Ampersand) ใช้ในการกำหนดตำแหน่งที่อยู่ของตัวแปรให้กับพอยเตอร์ โดยเมื่อสร้างตัวแปรชนิดพอยเตอร์มาแล้ว วิธีการที่จะนำค่าตำแหน่งในหน่วยความจำของตัวแปรใดๆ มาเก็บไว้ในตัวแปรพอยเตอร์ได้นั้น ต้องใช้เครื่องหมาย & โดยให้เขียนเครื่องหมาย & นำหน้าชื่อตัวแปรในหน่วยความจำ

วิธีการประกาศตัวแปร

```
pointer=&variable;
```

ตัวดำเนินการที่ใช้กับตัวแปรพอยเตอร์



PV = &V;

เครื่องหมาย “&” หมายถึง ที่อยู่ของ V (Address Operator)

พอยเตอร์ (Pointer)

- พอยเตอร์จะเก็บค่าตำแหน่งหรือแอดเดรสในหน่วยความจำของตัวแปรอื่นๆ ไว้
- ปกติการประกาศตัวแปรในการเก็บข้อมูลเป็นดังนี้

```
int age =23;
char char = ‘a’;
```

- การประกาศแบบพอยเตอร์

```
int age;
age = 23;
int *pointer;
pointer = & age;
```

- ตัวแปรพอยเตอร์ age จะไม่ได้เก็บค่า 23 แต่จะเก็บค่าตำแหน่งที่ข้อมูล 23 เก็บอยู่แทน

พอยเตอร์ (Pointer)

* Pointer เป็นตัวแปรประเภทหนึ่งในภาษา C ที่ต่างจากตัวแปรทั่ว ๆ ไปคือ ตัวแปรทั่ว ๆ ไปจะประกาศแบบนี้คือ

```
int a,b,c;

a=b=10;

c=a;
```

โดย c จะเก็บค่า 10

พอยเตอร์ (Pointer)

* แต่ถ้าประกาศดังนี้

int a,b;

int *c;

a=b=10;

c=&a;

จะเห็นได้ว่า ตัวแปร c เป็น pointer และให้ c=&a ฉะนั้น c จะไม่ได้เก็บค่า 10 แต่ c จะเก็บตำแหน่งของตัวแปร a ที่อยู่ในหน่วยความจำ และทำให้ c สามารถเข้าถึงข้อมูลในตัวแปร a ได้

แสดงตำแหน่งข้อมูลด้วย & (address operation)

ตัวดำเนินการที่ใช้กับตัวแปรพอยน์เตอร์ (Pointer)

1. ตัวดำเนินการ & (Ampersand)

- ใช้เพื่อเก็บค่าที่อยู่ (address) ของตัวแปร ไปเก็บไว้ในตัวแปรพอยน์เตอร์
- และใช้ในการแสดงค่าแอดเดรส (Address) ของตัวแปรพอยน์เตอร์
- ตัวอย่าง

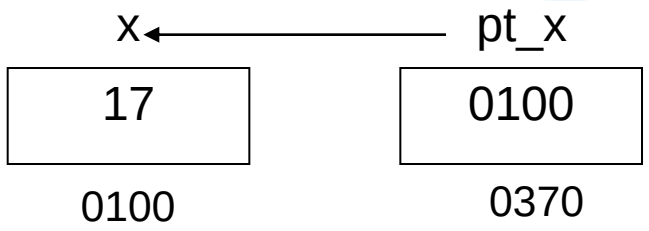
```
int     Num = 1000
int     *N
N = &Num
```

แสดงการชี้ของตัวแปรพอยน์เตอร์ *N ไปที่ตัวแปร Num

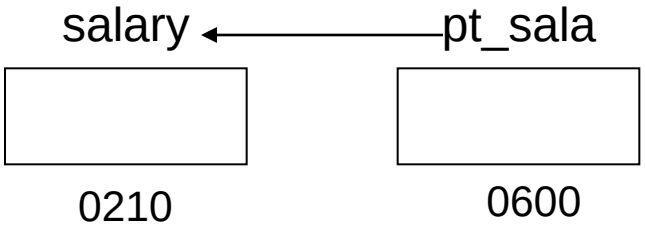
แสดงตำแหน่งข้อมูลด้วย & (address operation)

* ตัวอย่างการใช้เครื่องหมาย &

```
int x=17;
int *pt_x;
pt_x=&x;
```



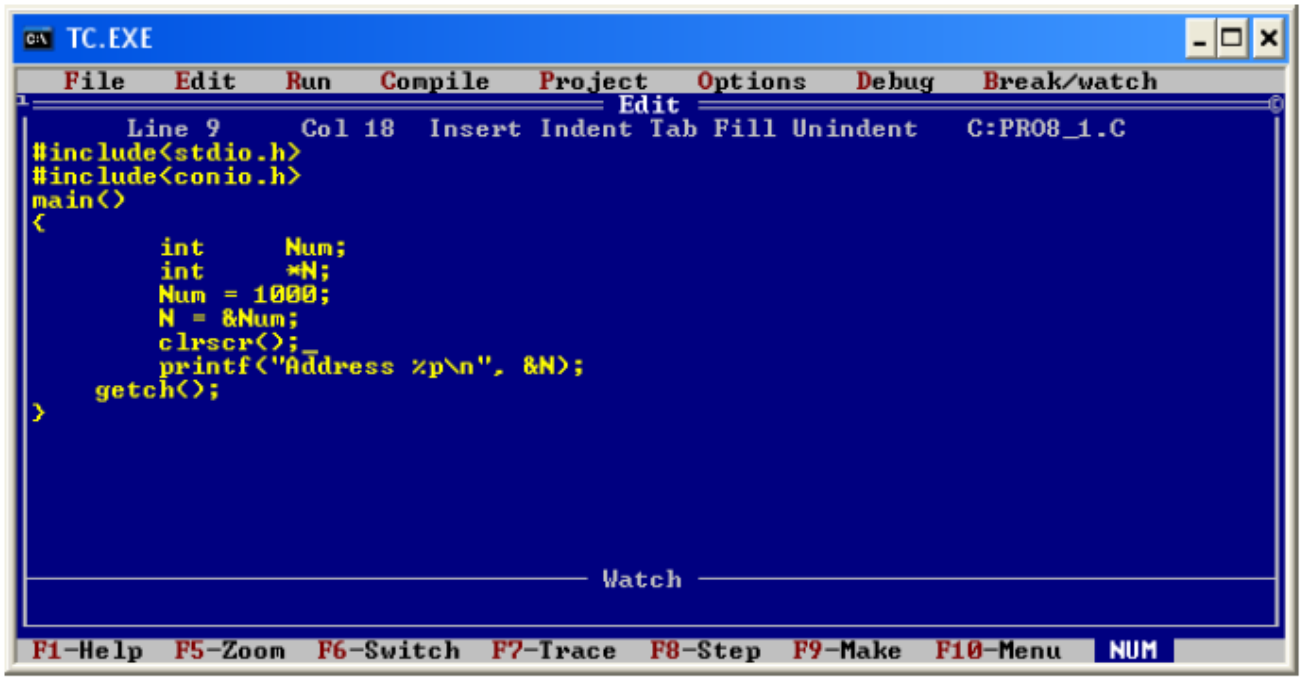

```
float salary=1200.00;
float *pt_sala;
pt_sala=&salary;
```



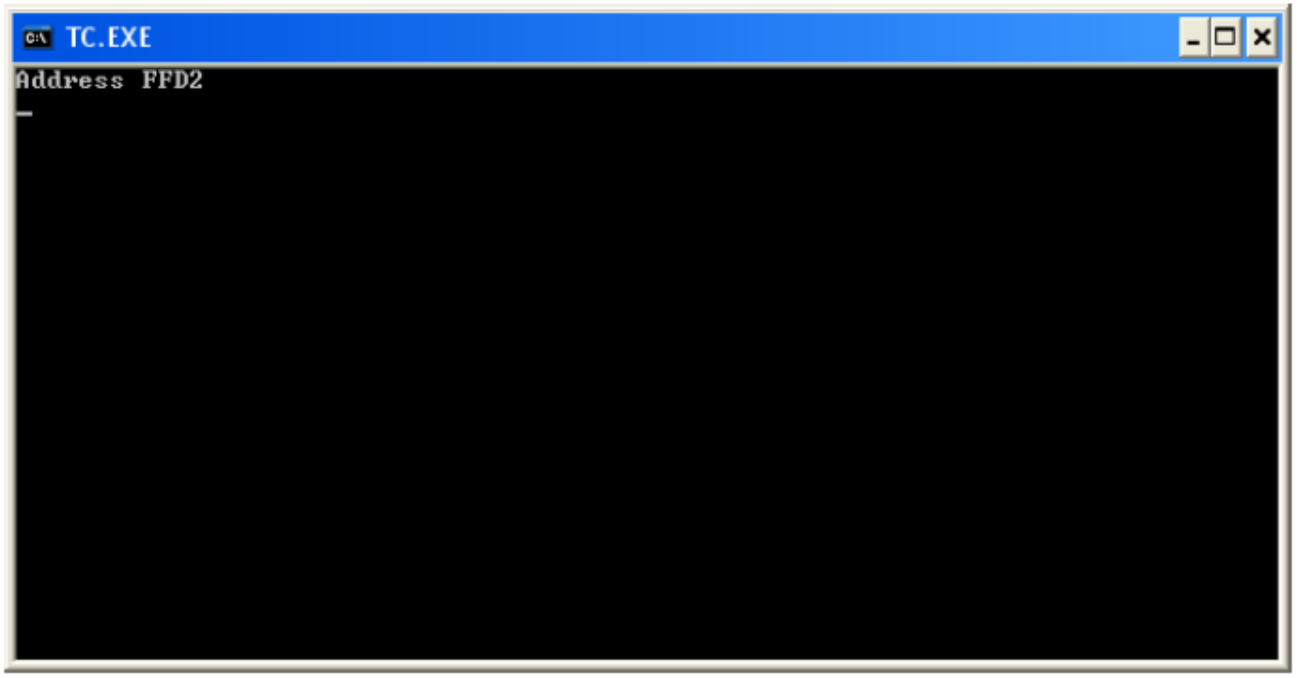
การแสดงตำแหน่งผลในโปรแกรมโดยใช้ pointer

* การเขียนโปรแกรมเพื่อให้แสดงตำแหน่งในหน่วยความจำที่เก็บไว้ในตัวแปร์พอยเตอร์ออกทางหน้าจอนั้น จะใช้คำสั่ง printf ตามปกติ แต่จะใช้รูปแบบการแสดงผลเป็น “**%p**” ซึ่งใช้สำหรับการแสดงตำแหน่งหน่วยความจำในตัวแปรพอยเตอร์โดยเฉพาะ

ตัวอย่างโปรแกรม



ผลลัพธ์ของโปรแกรม



ตัวอย่างโปรแกรมโดยใช้เครื่องหมาย & กับตัวแปรพอยเตอร์

```
#include<stdio.h>
void main()
{
    int x=17;
    float salary=1200.00;
    char letter='w';
    int *pt_x;
    float *pt_salary;
    char *pt_letter;
            pt_x=&x;
            pt_salary=&salary;
            pt_letter=&letter;
            printf("Address of x = %p \n",pt_x);
            printf("Address of salary = %p \n",pt_salary);
            printf("Address of letter = %p \n",pt_letter);
}
```

ตัวดำเนินการที่ใช้กับตัวแปรพอยเตอร์

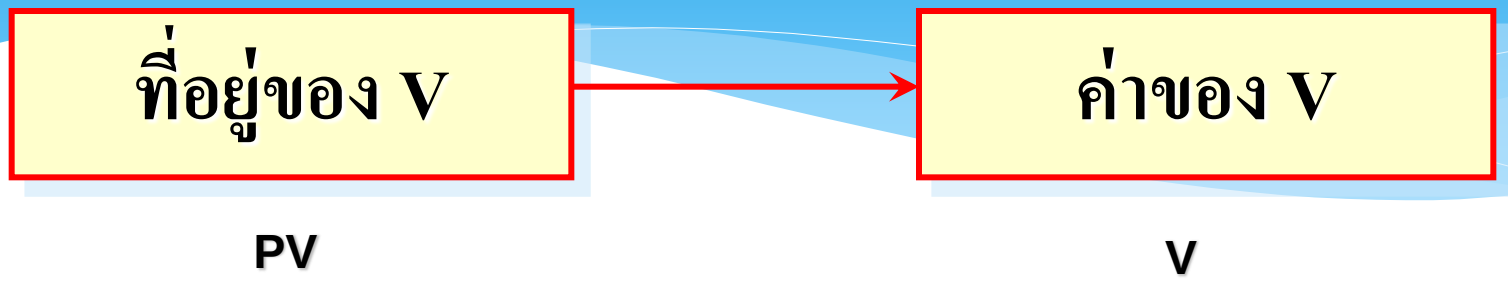
2. แสดงค่าข้อมูลด้วย * (indirect operation)

* **เครื่องหมาย * อ้างถึงข้อมูลหรือค่าที่เก็บไว้ในหน่วยความจำตำแหน่งที่ชี้**
  โดยสามารถหาค่าข้อมูลจากตำแหน่งในหน่วยความจำที่เก็บไว้ในตัวแปรพอยเตอร์ โดยการเขียนเครื่องหมาย * นำหน้าชื่อตัวแปรพอยเตอร์

วิธีการประกาศตัวแปร

```
variable=*pointer;
```

ตัวดำเนินการที่ใช้กับตัวแปรพอยเตอร์



PV = &V;

เครื่องหมาย “&” หมายถึง ที่อยู่ของ V (Address Operator)

U = *PV;

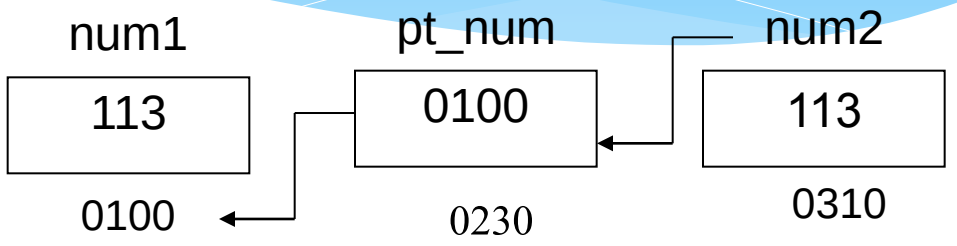
ดังนั้นหากเราต้องการอ้างอิงค่าของ V เราต้องใช้เครื่องหมาย “*” (Indirect Operator) ให้กับ Pointer แล้วนำค่าที่ได้ไปเก็บไว้ใน U

ดังนั้น U จะมีค่าเท่ากับ V

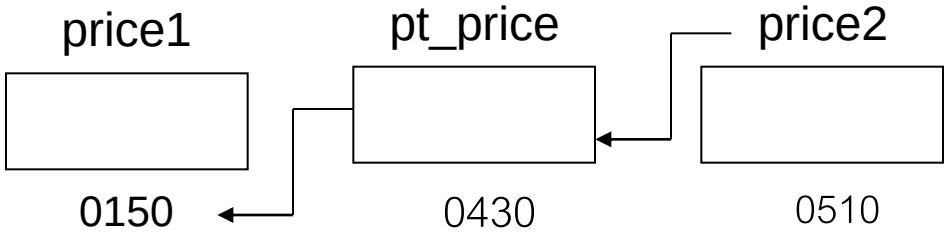
แสดงค่าข้อมูลด้วย * (indirect operation)

* ตัวอย่างการใช้เครื่องหมาย *

```
int num1=113,num2;
int *pt_num;
pt_num=&num1;
num2=*pt_num;
```




```
float price1=4.85,price2;
float *pt_price;
pt_price=&price1;
price2=*pt_price;
```



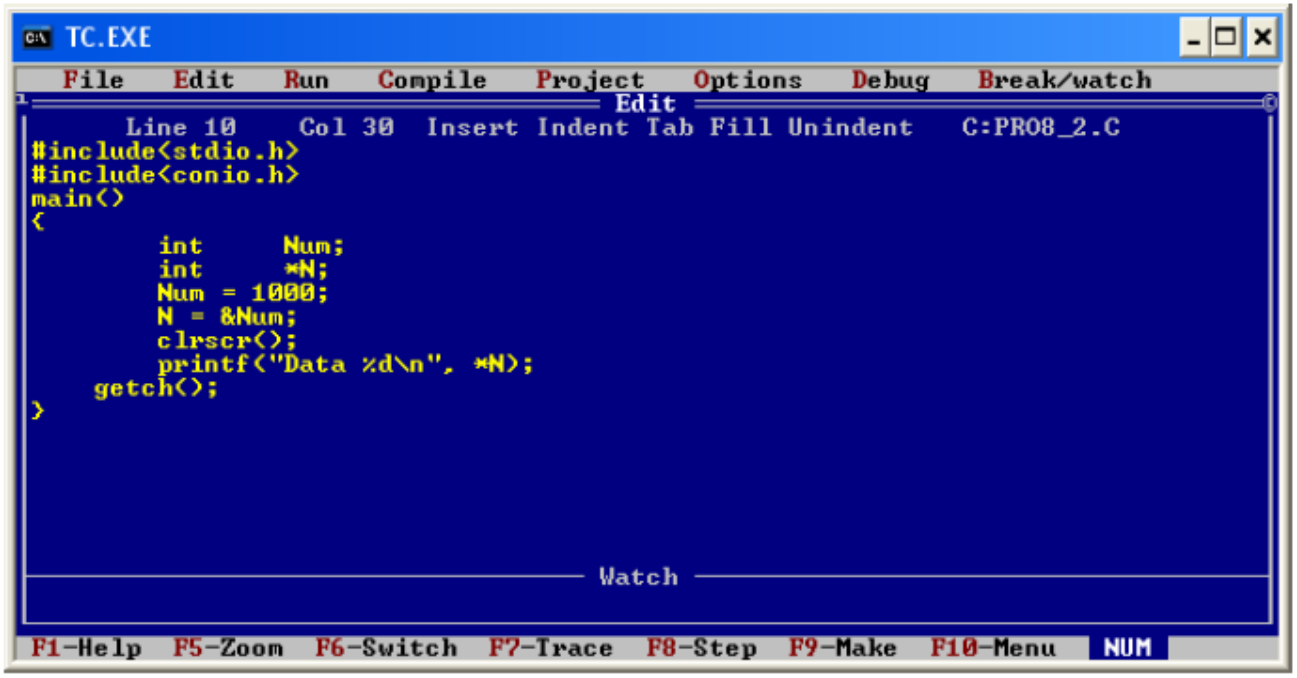
พอยเตอร์ (Pointer)

* ตัวอย่าง

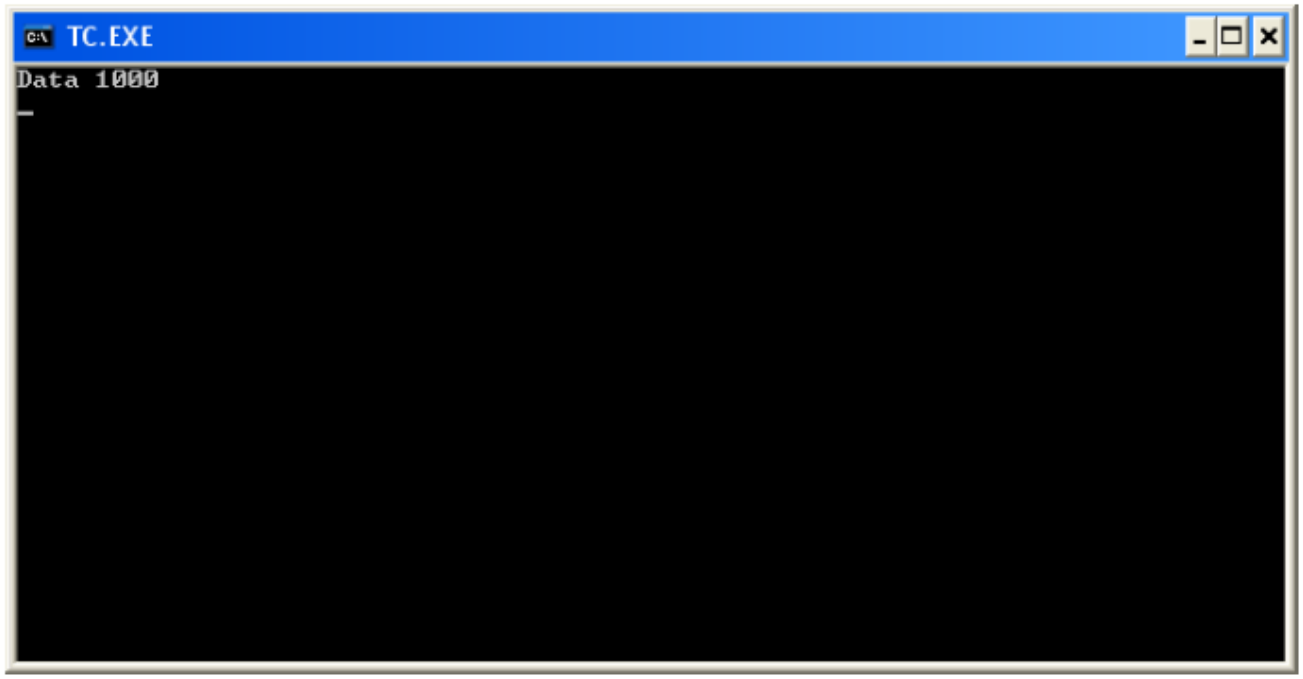
```
int a,b,*c;
a=b=10;
printf("a=%d",a);
c=&a;
*c=21;
printf("a=%d",a);
```

เมื่อรันโปรแกรมนี้จะได้ผลลัพธ์อย่างไร?

ตัวอย่างโปรแกรม



ผลลัพธ์ของโปรแกรม



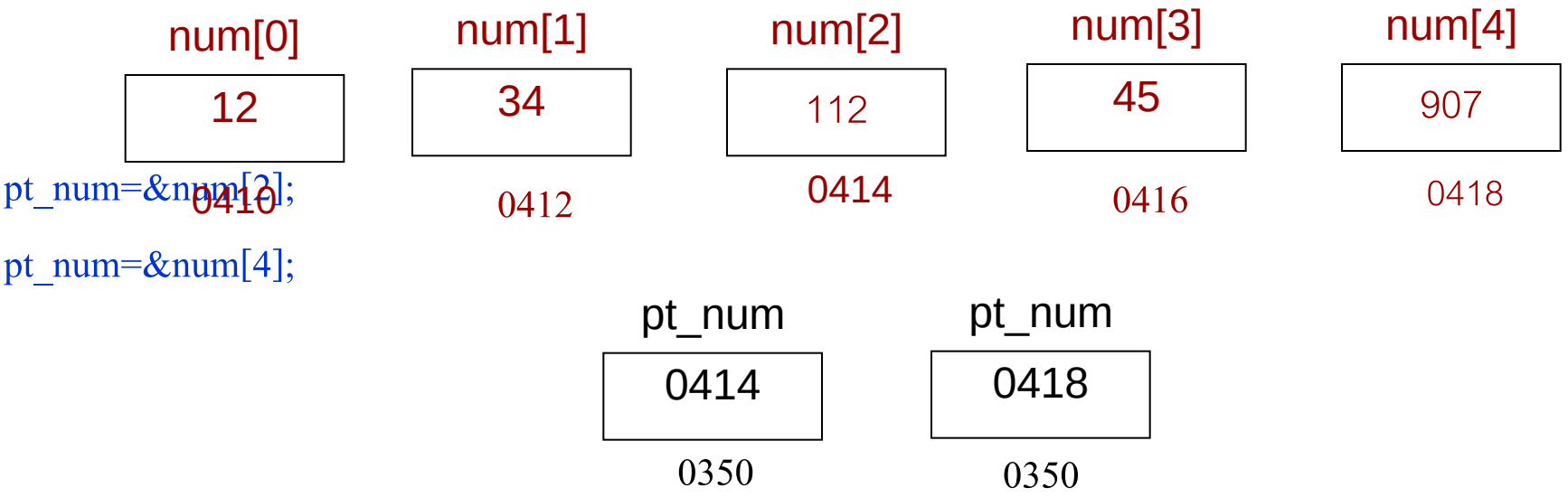
ตัวอย่างโปรแกรมโดยใช้เครื่องหมาย * กับตัวแปรพอยเตอร์

```
#include<stdio.h>
void main()
   {       int num1=113,num2;
           float price=4.85;
           char hint1='a',hint2;
           int *pt_num;
           float *pt_price;
           char *pt_hint1;
                   pt_num=&num1;
                   pt_price=&price;
                   pt_hint1=&hint1;
                   num2=*pt_num;
                   hint2=*pt_hint1;
                   printf("Variable of num2 = %d \n",*pt_num);
                   printf("Variabie of Price = %p \n",&pt_price);
                   printf("Variabie of hint2 = %c \n",hint2);
   }
```

ตัวแปรพอยเตอร์กับอาร์เรย์

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

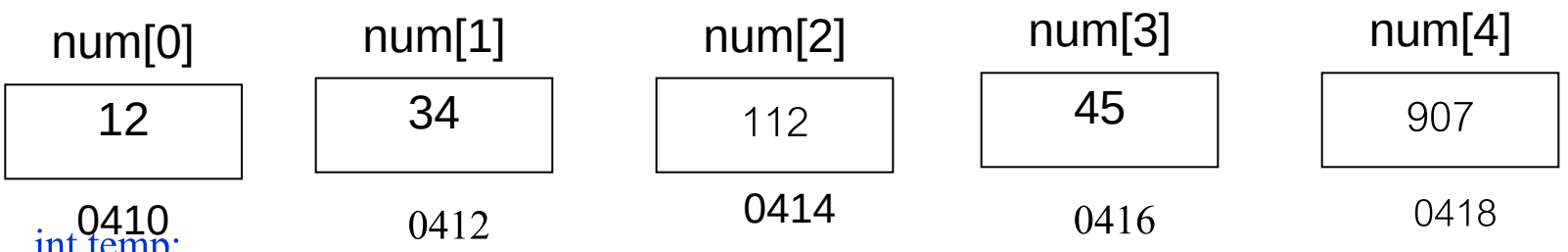
  int *pt_num;



ตัวแปรพอยเตอร์กับอาร์เรย์

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

  int *pt_num;

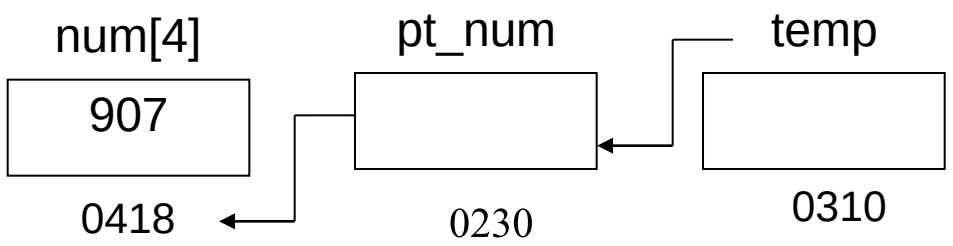


int temp;

pt_num=&num[4];

temp=*pt_num;

ค่าของ temp จะมีค่าเท่าใด?



ตัวอย่างโปรแกรมตัวแปรพอยเตอร์กับอาร์เรย์

```
#include<stdio.h>
    int x[5]={11,43,523,789,101};
    int *pt_1,*pt_2,*pt_3,*pt_4,*pt_5;
void main()
    {       pt_1=&x[0];
            pt_2=&x[1];
            pt_3=&x[2];
            pt_4=&x[3];
            pt_5=&x[4];
            printf("Address of x[0] = %p \n",pt_1);
            printf("Address of x[1] = %p \n",pt_2);
            printf("Value of x[2] = %d \n",*pt_3);
            printf("Value of x[3] = %d \n",*pt_4);
            printf("Value of x[4] = %d \n",*pt_5);
            x[3]=*pt_5;    // กำหนดค่าของตัวแปรอาร์เรย์ x[3] ใหม่ด้วยข้อมูลที่เก็บไว้ในตัวแปรพอยเตอร์ pt_5
            printf("Value of x[3] = %d \n",x[3]);
    }
```

ตัวแปรพอยเตอร์กับอาร์เรย์

ตัวแปรพอยน์เตอร์(Pointer) กับตัวแปรชุดหรืออาร์เรย์(Array)

- ตัวแปรชุดหรืออาร์เรย์ (Array) เป็นโครงสร้างข้อมูล (Data Structure) ที่มีหน้าที่ในการจองเนื้อที่หน่วยความจำสำหรับจัดเก็บข้อมูล สามารถอ้างอิงการใช้ข้อมูลในอาร์เรย์ได้โดยผ่านตัวชี้ข้อมูล (Index) หรือ Subscript ของตัวแปรชุด (Array) ได้
- สำหรับตัวแปรพอยน์เตอร์ (pointer) นั้น จะใช้ในการอ้างอิงตำแหน่งที่อยู่ (address) ของตัวแปรชุดหรืออาร์เรย์ (array) เพื่อนำที่อยู่ (address) หรือข้อมูล (data) ของตัวแปรนั้นมาแสดงผล
- ตัวอย่าง

```
int     Serial_Number {100,200,300,400,500}
```

จำนวนข้อมูลที่อยู่ในอาร์เรย์ (Array)

Serial_Num[0]	Serial_Num[1]	Serial_Num[2]	Serial_Num[3]	Serial_Num[4]
100	200	300	400	500

การอ้างถึงตำแหน่งในอาร์เรย์ผ่านตัวชี้

- นอกจากนี้ยังสามารถใช้พอยน์เตอร์แทนอาร์เรย์ การอ้างถึงค่าในอาร์เรย์โดยใช้ a[i] สามารถใช้ *(a+i)

- เนื่องจากทุกครั้งที่อ้างถึง a[i] ภาษาซีจะทำหน้าที่แปลงเป็น *(a+i) เพราะฉะนั้นการเขียนในรูปแบบใดก็ให้ผลลัพธ์ในการทำงานเช่นเดียวกัน

- การอ้างถึงแอดเดรส เช่น &a[i] จะมีผลเท่ากับการใช้ a+i

การอ้างถึงตำแหน่งในอาร์เรย์ผ่านตัวชี้

- ในลักษณะเดียวกันการใช้งานพอยน์เตอร์ก็สามารถใช้คำสั่งในลักษณะอาร์เรย์ก็ได้ เช่น การอ้างถึง *(pa+i) สามารถเขียนด้วย pa[i] ก็ได้ผลเช่นเดียวกัน
- สิ่งที่แตกต่างกันของอาร์เรย์และพอยน์เตอร์ คือ พอยน์เตอร์เป็นตัวแปร แต่อาร์เรย์ไม่ใช่ตัวแปร

การใช้งาน Pointer กับอาร์เรย์ 1 มิติ

* ตัวอย่าง

```
int x[3] = {10, 11, 12};
```

// Address ของอาร์เรย์ตัวที่ 0 ไม่ได้มีค่าเท่ากับ 10 แต่อาจมีค่าเป็น FFF5 ขึ้นอยู่กับการจองหน่วยความจำ ณ ขณะนั้น

&x[0] และก็มีค่าเท่ากับ x + 0 // FFF5

&x[1] และก็มีค่าเท่ากับ x + 1 // FFF7

&x[2] และก็มีค่าเท่ากับ x + 2 // FFF9

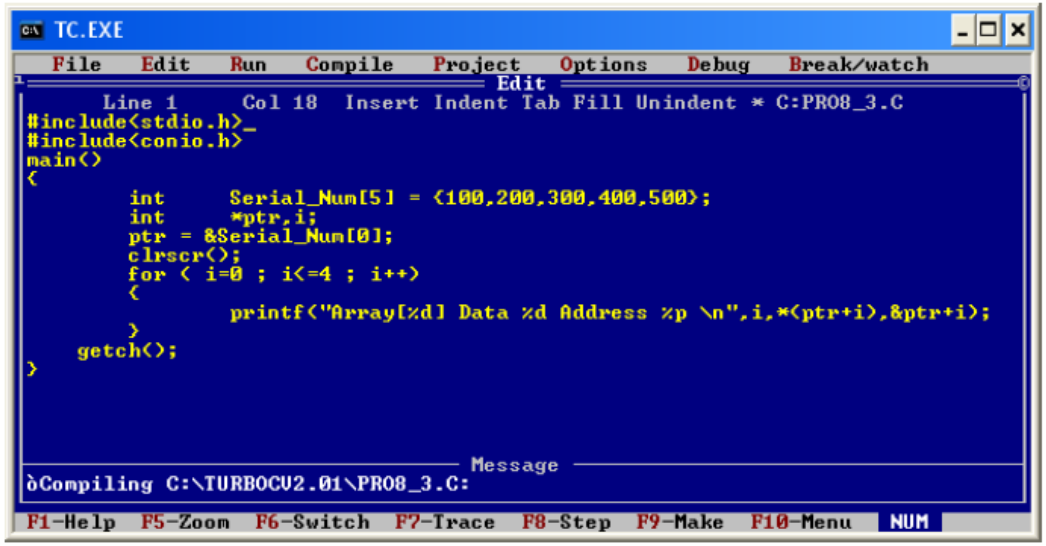
x[0] และก็มีค่าเท่ากับ *(x + 0) // 10

x[1] และก็มีค่าเท่ากับ *(x + 1) // 11

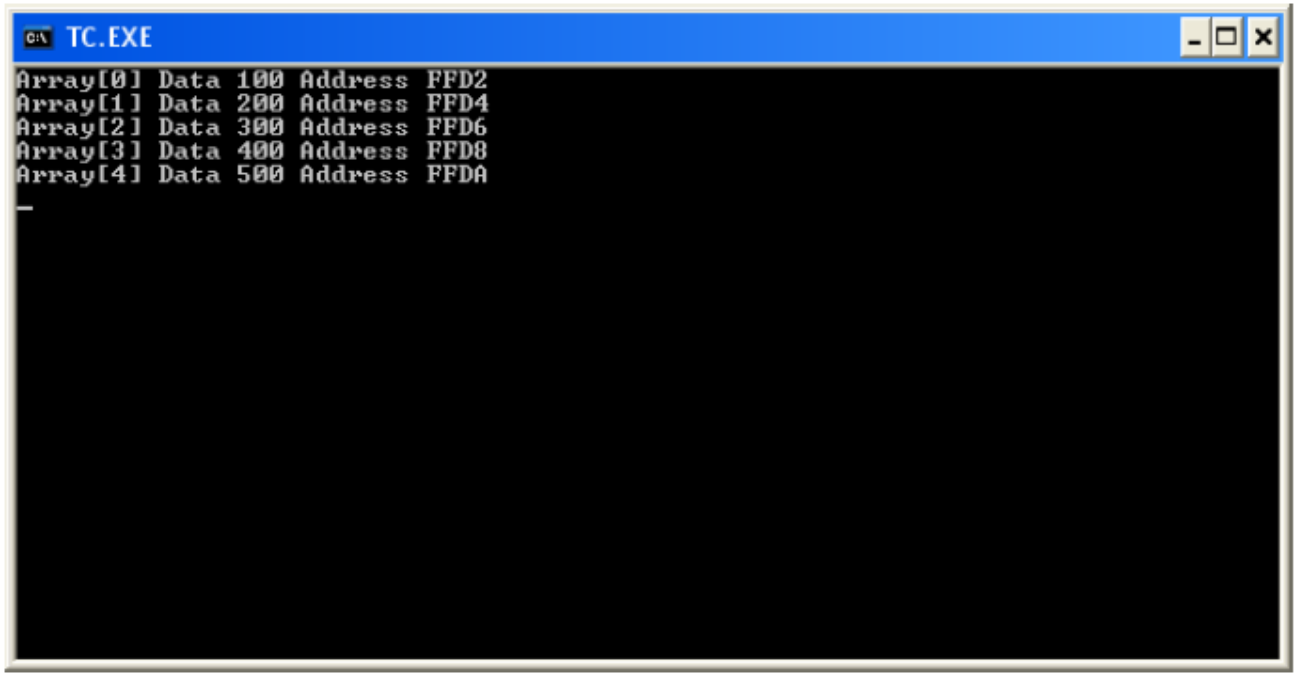
x[2] และก็มีค่าเท่ากับ *(x + 2) // 12

ตัวแปรพอยน์เตอร์(Pointer) กับตัวแปรชุดหรืออาร์เรย์(Array)

ตัวอย่างโปรแกรม รูป 8.3



ผลลัพธ์ของโปรแกรม



ประโยชน์ของ Pointer

* มีประโยชน์เมื่อต้องเขียนโปรแกรมจัดการกับโครงสร้างข้อมูลขนาดใหญ่ ที่มีข้อมูลจำนวนมาก การจัดเรียงหรือการประมวลผลต่อข้อมูลโดยใช้ pointer และ array แม้จะทำได้รวดเร็ว แต่ในกรณีที่มีข้อมูลจำนวนมหาศาล การจัดเรียงหรือประมวลผลหากกระทำต่อตัวข้อมูลใน array โดยตรงจะยังช้าเกินไป
* หลีกเลี่ยงการใช้ array เก็บตัวข้อมูล เมื่อต้องการจัดเรียงหรือประมวลผลกับข้อมูล เราจะทำทางอ้อมโดยจัดเรียงหรือประมวลผลกับค่าของ pointer ที่ถูกชี้ซึ่งทำได้รวดเร็วกว่าการกระทำกับข้อมูลโดยตรงมาก ทำให้การจัดการกับข้อมูลในโครงสร้างข้อมูลที่มีข้อมูลจำนวนมาก สามารถทำได้ด้วยความเร็วสูง

ประโยชน์ของของ Pointer

* **ข้อสรุปเรื่อง Pointer**
  * ทำหน้าที่ชี้ไปยังตำแหน่งเก็บข้อมูลในหน่วยความจำ
  * การประกาศ pointer ต้องกำหนด data type ด้วย
  * ใช้ pointer ชี้ไปยัง pointer หรือชี้ไปยัง array หรือ array ของ pointer ได้
  * การอ้างถึงตำแหน่งของตัวแปร ใช้เครื่องหมาย & หน้าตัวแปร pointer เช่น &pt
  * การอ้างถึงค่าในตัวแปร ใช้เครื่องหมาย * หน้าตัวแปร pointer เช่น *pt
  * pointer ที่ชี้ไปยัง pointer ใช้ดอกจันสองตัว เช่น int **p