Pointer ตัวชี้

เอกสารประกอบการอบรม

สอวน. สาขาคอมพิวเตอร์ ศูนย์โรงเรียนสวนกุหลาบวิทยาลัย นนทบุรี

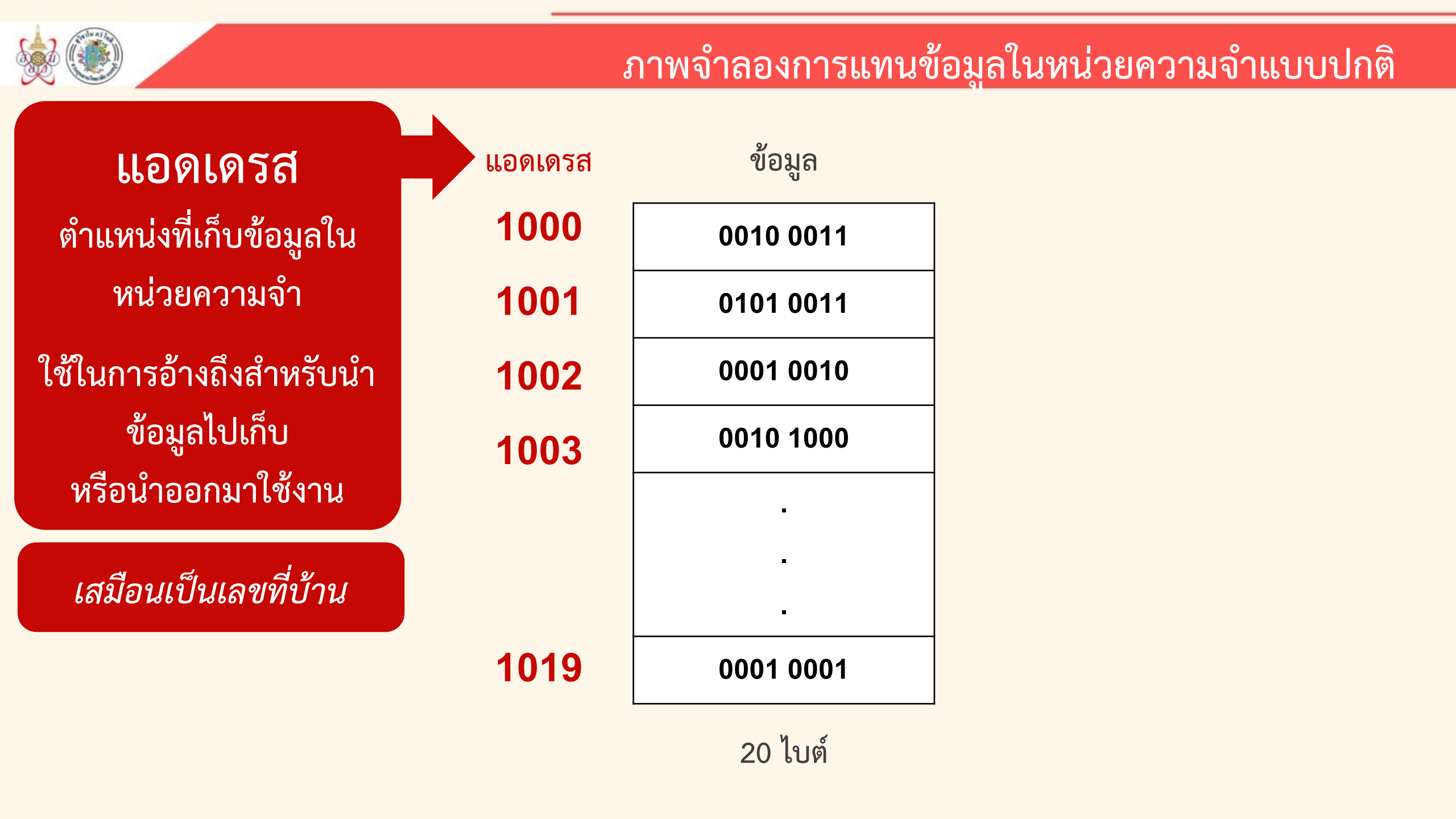
ภาพจำลองการแทนข้อมูลในหน่วยความจำแบบปกติ

แอดเดรส

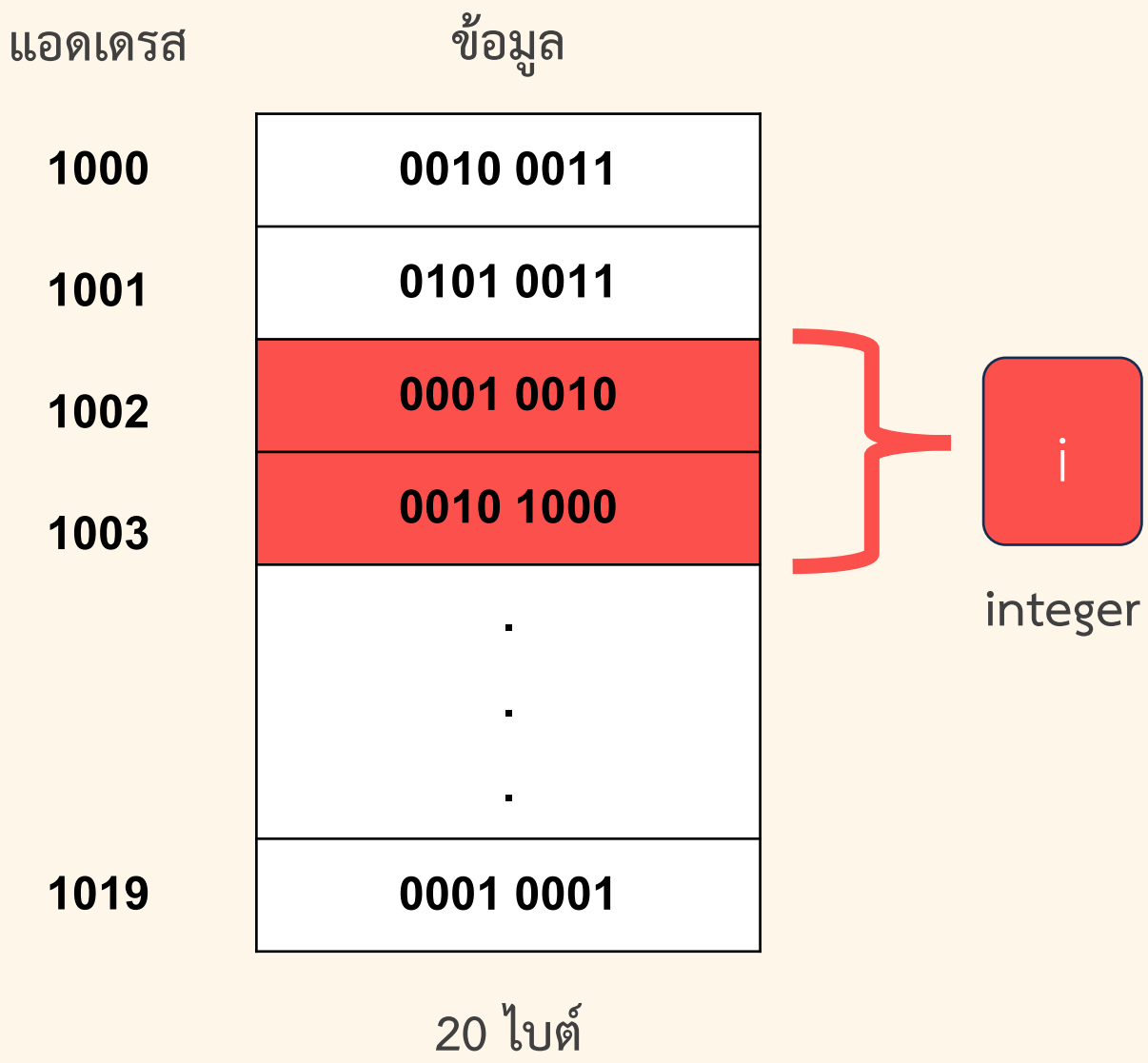
ตำแหน่งที่เก็บข้อมูลในหน่วยความจำ

ใช้ในการอ้างถึงสำหรับนำข้อมูลไปเก็บหรือนำออกมาใช้งาน

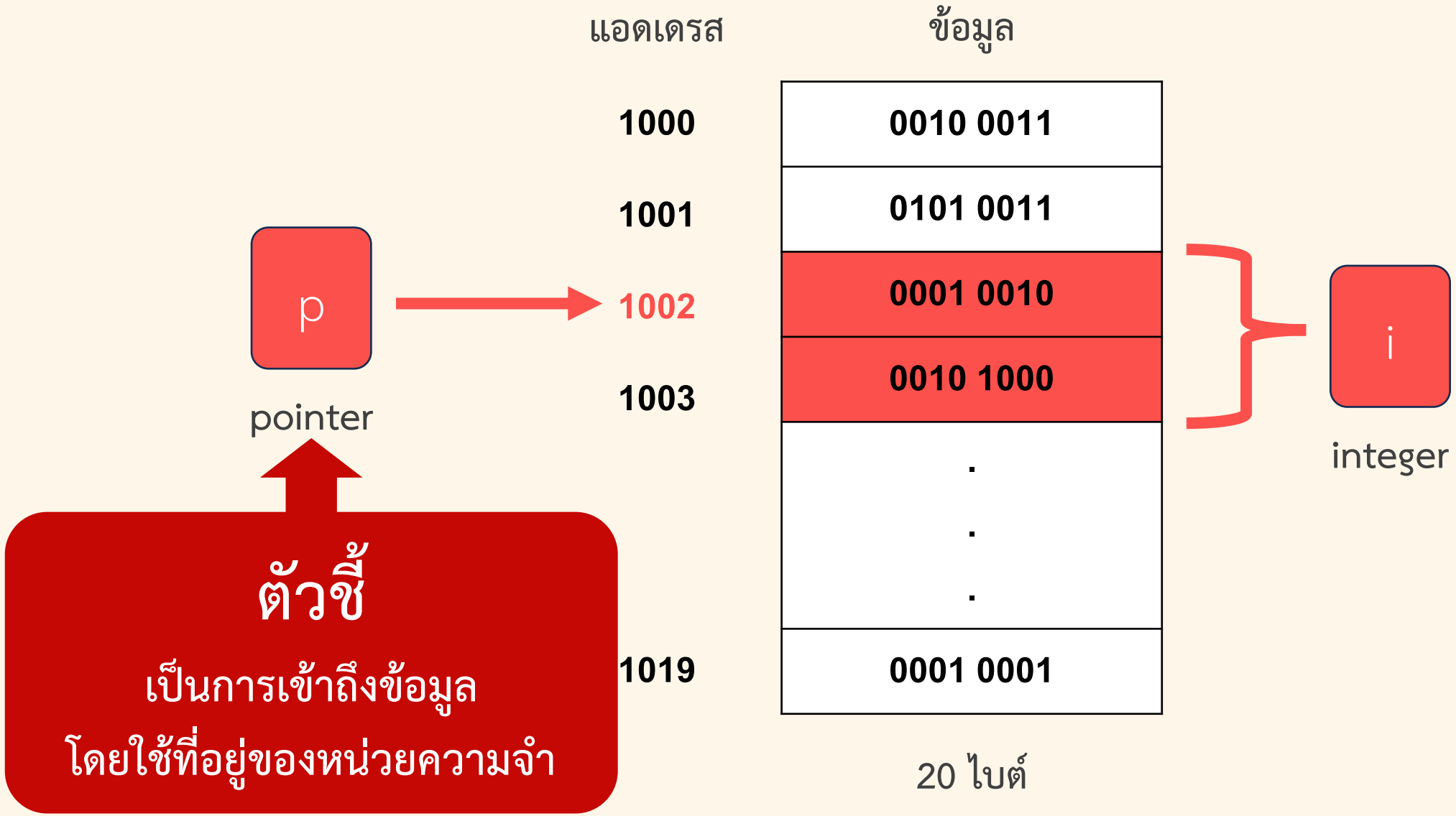
เสมือนเป็นเลขที่บ้าน



ภาพจำลองการแทนข้อมูลในหน่วยความจำแบบปกติ



ภาพจำลองการแทนข้อมูลในหน่วยความจำแบบปกติ



ตัวชี้

เป็นการเข้าถึงข้อมูล
โดยใช้ที่อยู่ของหน่วยความจำ

p

pointer

i

integer

ตัวแปรพอยน์เตอร์

เก็บค่าตำแหน่งของข้อมูล
ที่เก็บในหน่วยความจำ

ตัวแปรปกติ

เก็บค่าที่แท้จริงของข้อมูล

การใช้พอยน์เตอร์เป็นอีกวิธีที่จะเข้าถึงตัวแปรปกติได้

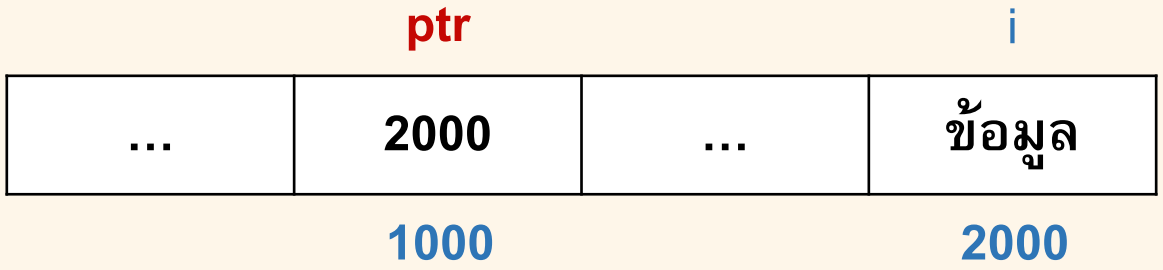
ตัวแปรพอยน์เตอร์จะเก็บค่าที่อยู่ของหน่วยความจำหลัก
ซึ่งต่างกับตัวแปรปกติที่เก็บค่าที่แท้จริงของข้อมูล

การใช้ตัวแปรพอยน์เตอร์จะเป็นการเข้าถึงข้อมูล
หรือเป็นการอ้างถึงตำแหน่งที่เก็บข้อมูลของตัวแปรอื่น

ประเภทของพอยน์เตอร์

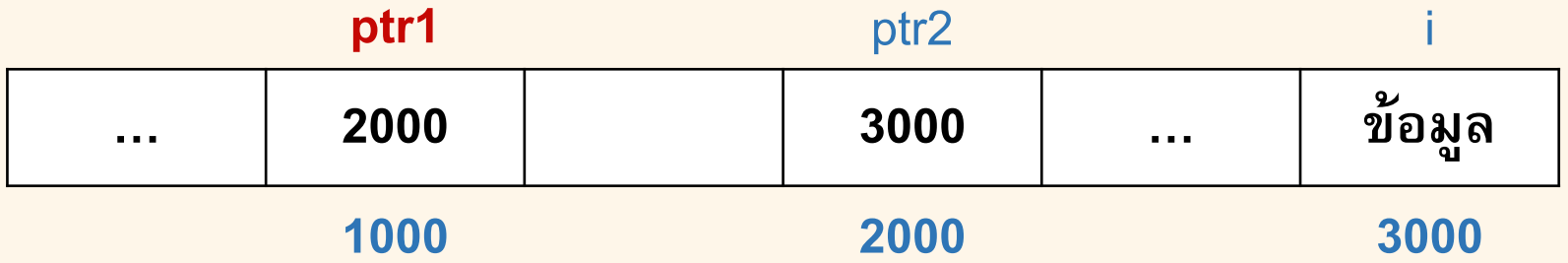
- **Direct Pointer variable**

  ตัวแปรพอยน์เตอร์ที่เก็บตำแหน่งของข้อมูลภายในหน่วยความจำโดยตรง



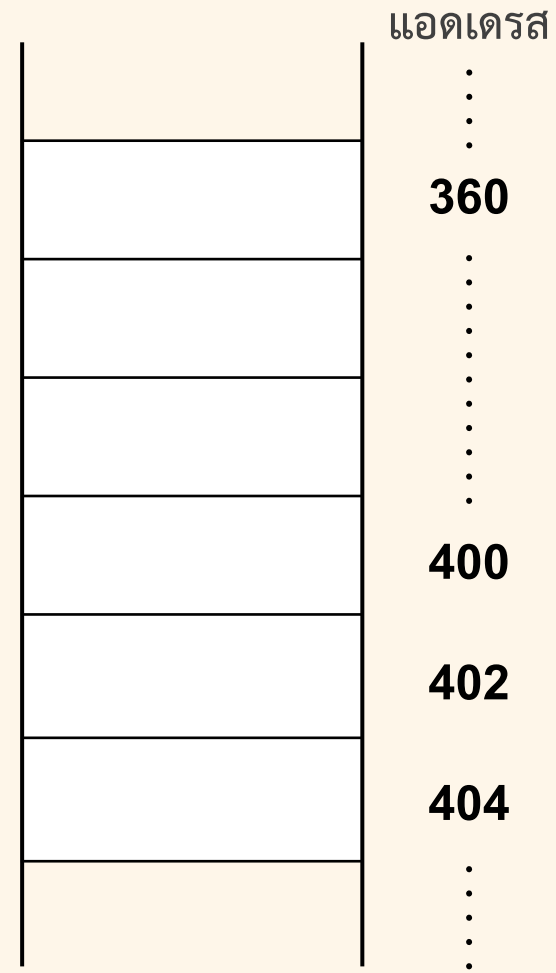
- **Indirect Pointer variable (Pointer to Pointer)**

  ตัวแปรชนิดพอยน์เตอร์ที่เก็บตำแหน่งของตัวแปรชนิดพอยน์เตอร์อีกตัวหนึ่ง



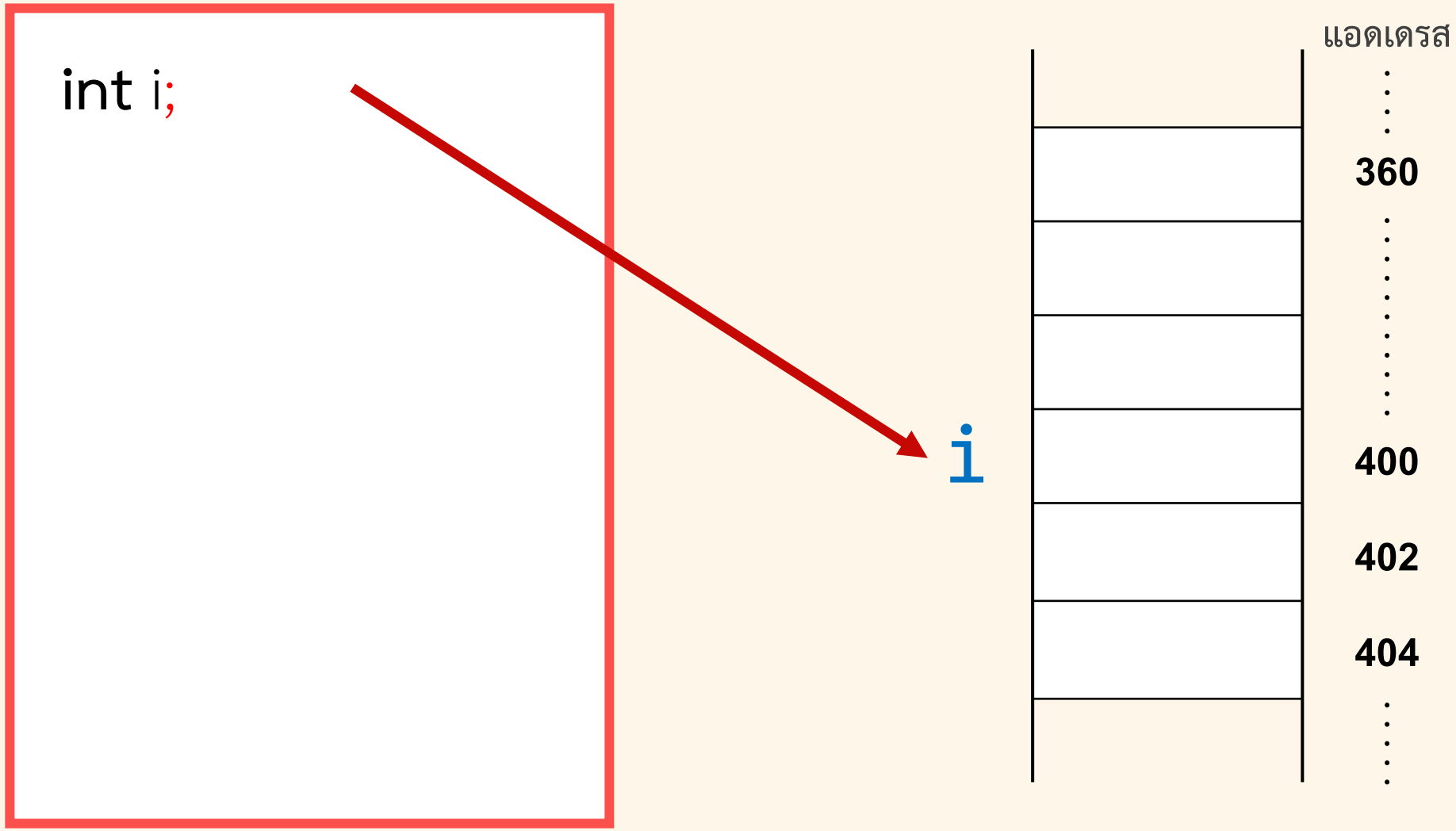
การแทนข้อมูลในหน่วยความจำ
ของตัวแปรประเภทพื้นฐาน และตัวแปรประเภทตัวชี้

การแทนข้อมูลในหน่วยความจำของตัวแปรพื้นฐาน และตัวแปรพอยน์เตอร์



การแทนข้อมูลในหน่วยความจำของตัวแปรประเภทพื้นฐาน

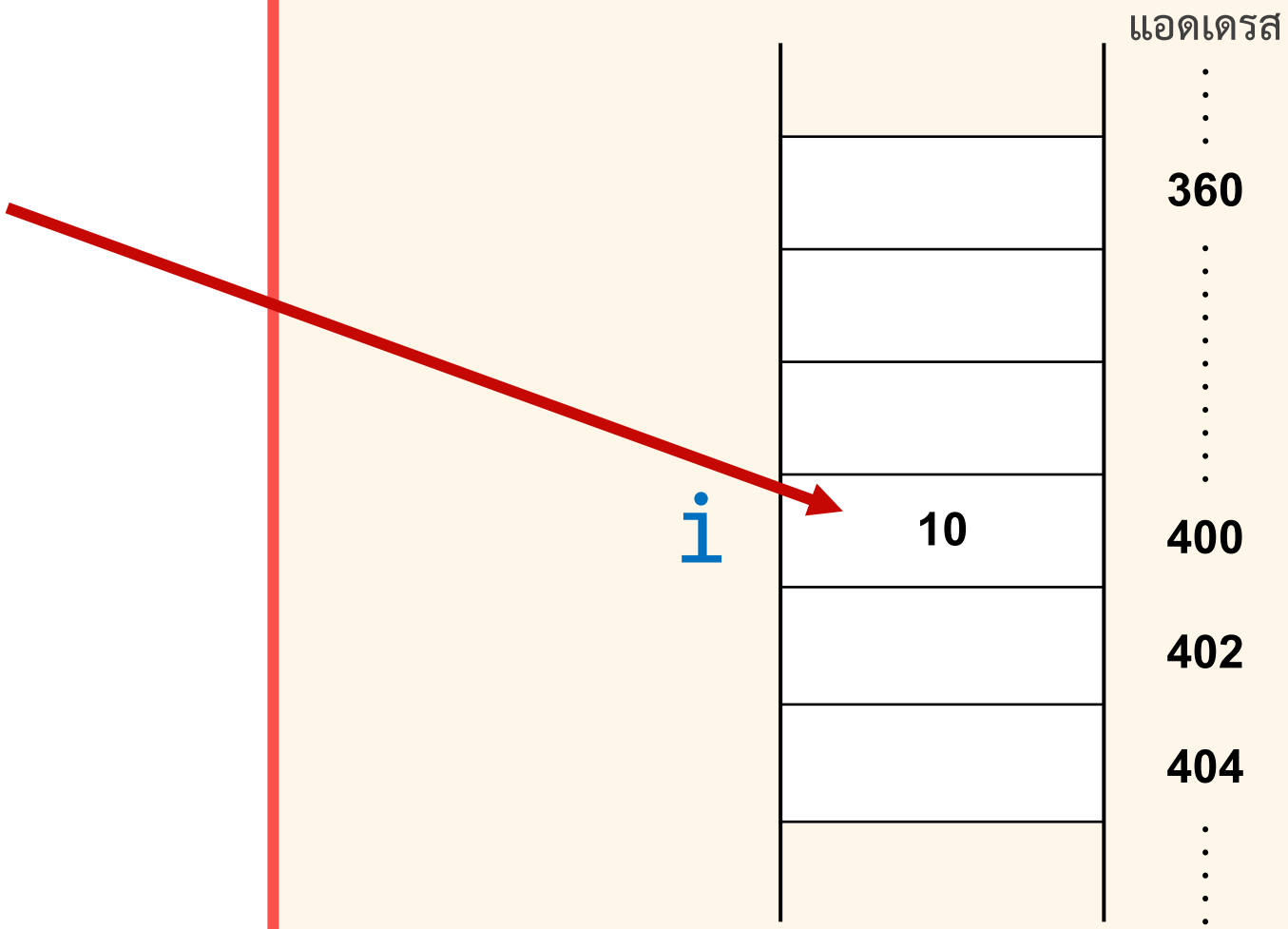
การแทนข้อมูลในหน่วยความจำของตัวแปรพื้นฐาน และตัวแปรพอยน์เตอร์



การแทนข้อมูลในหน่วยความจำของตัวแปรประเภทพื้นฐาน

การแทนข้อมูลในหน่วยความจำของตัวแปรพื้นฐาน และตัวแปรพอยน์เตอร์

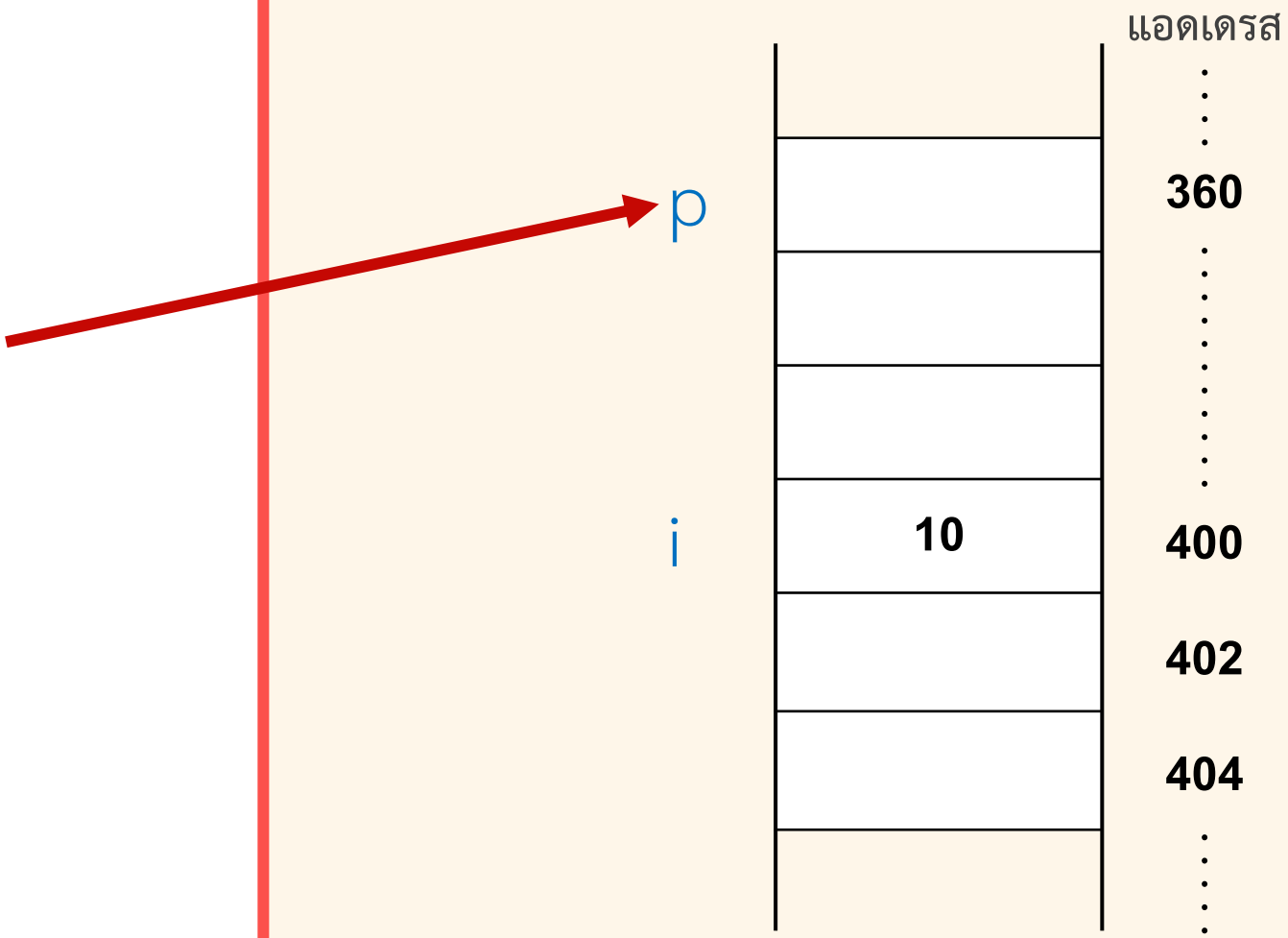
```
int i;
i = 10;
```



การแทนข้อมูลในหน่วยความจำของตัวแปรประเภทพื้นฐาน

การแทนข้อมูลในหน่วยความจำของตัวแปรพื้นฐาน และตัวแปรพอยน์เตอร์

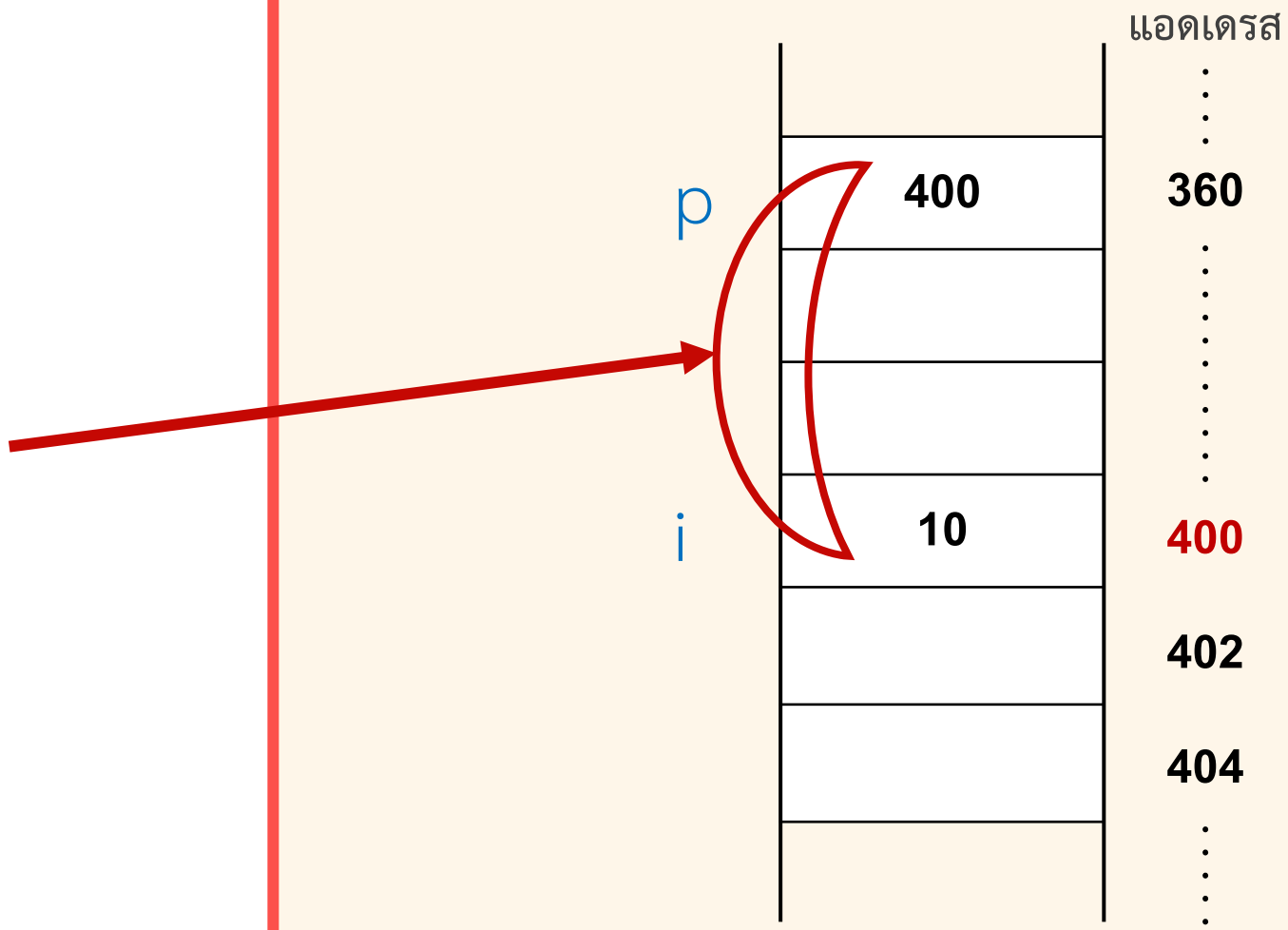
```
int i;
i = 10;
int *p;
```



การแทนข้อมูลในหน่วยความจำของตัวแปรประเภทตัวชี้

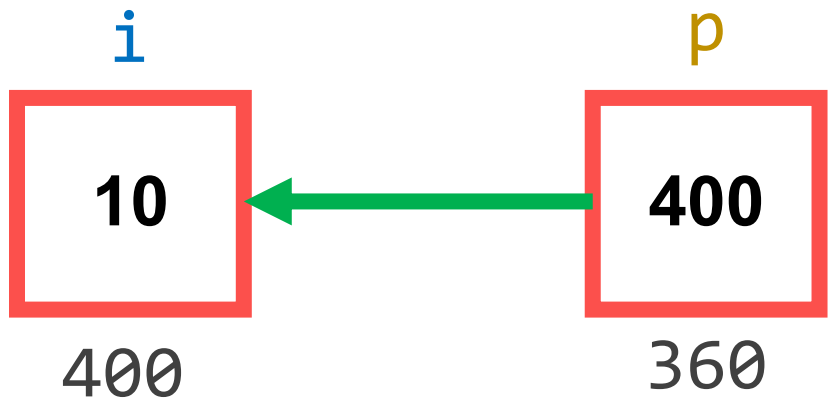
การแทนข้อมูลในหน่วยความจำของตัวแปรพื้นฐาน และตัวแปรพอยน์เตอร์

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



การแทนข้อมูลในหน่วยความจำของตัวแปรประเภทตัวชี้

```
int i;
i = 10;
int *p;
p = &i;
cout<<*p;
```



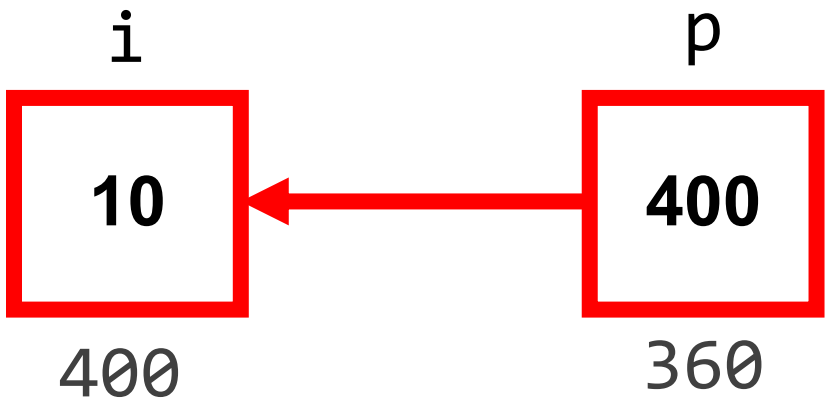

```
Output :
10
```

การแทนข้อมูลในหน่วยความจำของตัวแปรพื้นฐาน และตัวแปรพอยน์เตอร์

*****

เป็นการให้ชี้ไปที่ตำแหน่งของออบเจ็กต์นั้น
และดึงข้อมูลที่เก็บไว้ในออบเจ็กต์นั้น

```
p = &i;
cout<<*p;
```

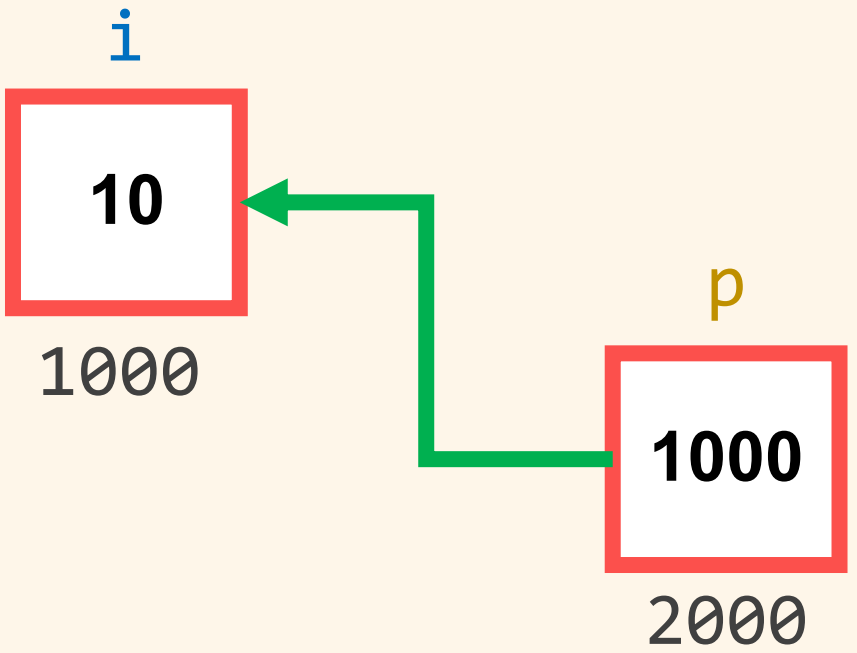


Output :

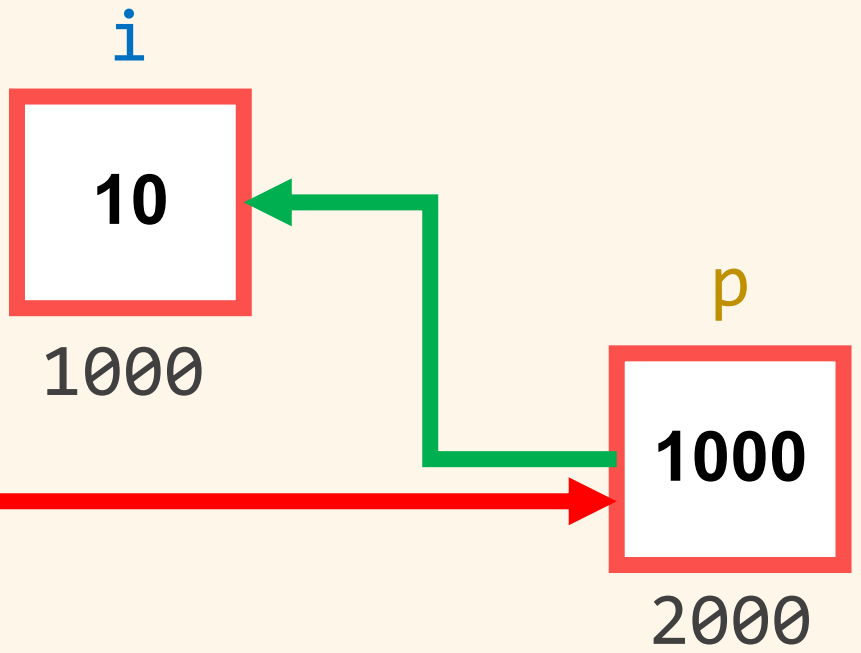
10

การเปลี่ยนค่าของออบเจ็กต์ที่ชี้โดยพอยน์เตอร์

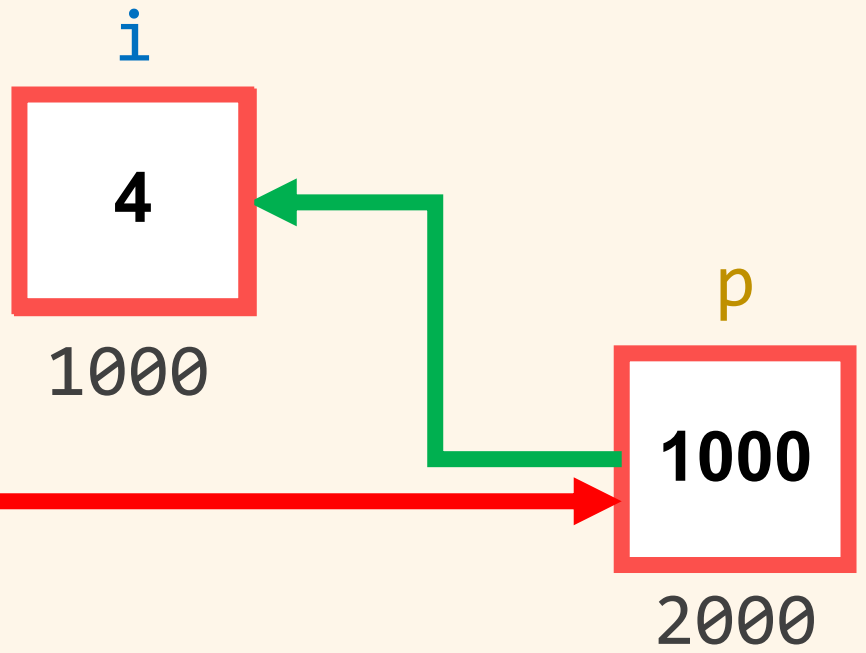
```
int i = 10;
int *p = &i;
```



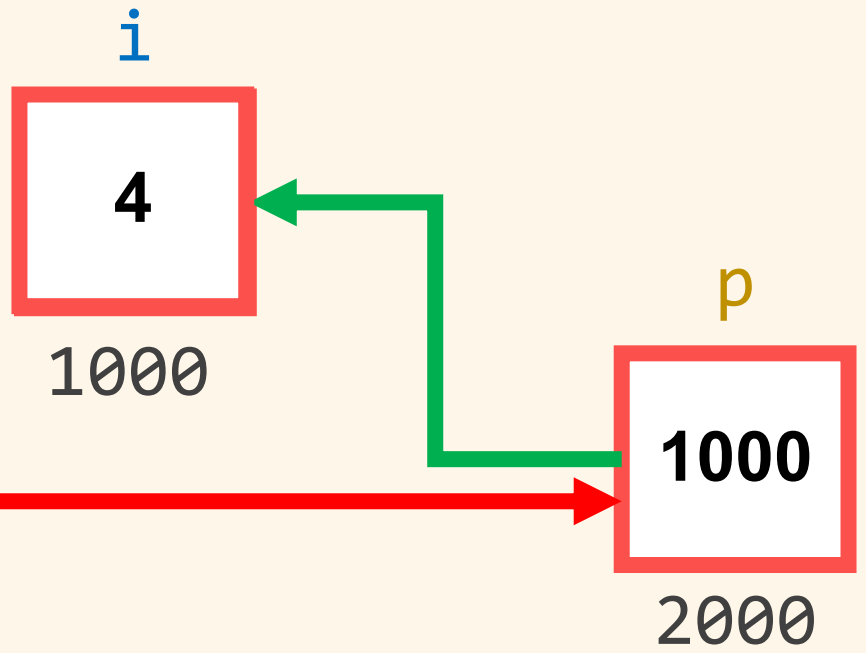
```
int i = 10;
int *p = &i;

*p = 4;
```



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

*p = 4;
```

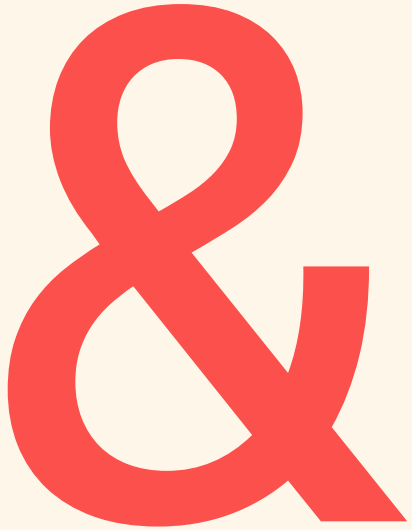


```
int i = 10;
int *p = &i;

*p = 4;

cout<<*p;
```




```
Output :
4
```

เครื่องหมายสำคัญที่ใช้ในพอยน์เตอร์



การประกาศตัวแปรแบบพอยน์เตอร์ ใช้สัญลักษณ์

(star)

Indirection operator
หรือ De-referencing operator

เป็นเครื่องหมายใช้เพื่ออ้างอิงข้อมูลที่ตัวแปรพอยน์เตอร์ชี้อยู่ โดยจะต้องประกาศตัวแปรพอยน์เตอร์ให้สอดคล้องกับประเภทของตัวแปรที่เราต้องการ

การประกาศตัวแปรพอยน์เตอร์

```
data_type *pointer_name;
```

ประเภทตัวแปร **ชื่อตัวแปร**

ต้องประกาศประเภทของตัวแปรพอยน์เตอร์
ให้สอดคล้องกับประเภทของ*ตัวแปรที่เราชี้ไป*

ยกเว้นตัวแปรพอยน์เตอร์
ประเภท void ที่สามารถชี้ไป
ยังตัวแปรประเภทใดก็ได้

ตัวแปรพอยน์เตอร์ไม่มี data type ของตนเอง เพราะเก็บแค่ค่า address ของออบเจ็กต์อื่น

ประเภทตัวแปร ชื่อตัวแปร

```
int *ptr;
```
ชี้ไปยังค่าที่เป็นตัวเลขจำนวนเต็ม (integer)

```
char *ptr;
```
ชี้ไปยังค่าที่เป็นตัวอักขระ (character)

```
float *ptr;
```
ชี้ไปยังค่าที่เป็นตัวเลขทศนิยม (float)

```
double *ptr;
```
ชี้ไปยังค่าที่เป็นตัวเลขทศนิยม ที่มีจำนวนบิตมากเป็นสองเท่าของ float (double)

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

เป็นการประกาศตัวแปร ptr เป็นตัวแปรพอยน์เตอร์ที่ชี้ไปยังตัวแปรประเภท double และประกาศฟังก์ชัน atof มีพารามิเตอร์เป็นตัวแปรพอยน์เตอร์ประเภท char

Declaration operator (*)

```
int i = 10;
int *ptr = &i; // ประกาศตัวแปรพอยน์เตอร์
cout *ptr; // เข้าถึงข้อมูลของตัวแปรพอยน์เตอร์
```

Dereference operator (*)

การประกาศตัวแปรแบบกำหนดค่าตำแหน่งแอดเดรสใช้สัญลักษณ์

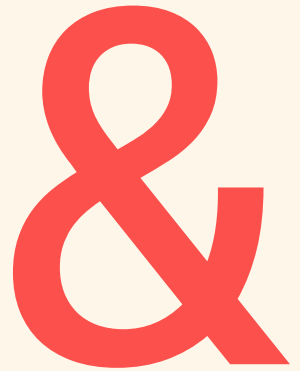
& Address of operator

(ampersand)

เพื่อคืนค่าตำแหน่งแอดเดรสของตัวแปรชนิดพอยน์เตอร์ โดยการกำหนดค่าชนิดนี้ต้องสอดคล้องกับชนิดข้อมูลของตัวแปรพอยน์เตอร์เท่านั้น

```
pointer_name = &variable_name
```

ชื่อตัวแปรพอยน์เตอร์ ชื่อตัวแปรที่เก็บข้อมูล *(ที่ต้องการดึงตำแหน่งมา)*



Address of operator

การใส่สัญลักษณ์ ampersand (&) หน้าตัวแปรนั้นจะทำให้เราได้รับค่าที่อยู่ของตัวแปรในหน่วยความจำ ที่อยู่นี้สามารถได้รับมาได้ในตอนที่โปรแกรมรันเท่านั้น ซึ่งโปรแกรมจะบอกว่าตัวแปรนั้นเก็บข้อมูลอยู่ที่ตำแหน่งใดของหน่วยความจำ

```
&i;
```

การใช้คำสั่งนี้จะทำให้ได้ค่า address ของตัวแปร i

```
int i = 8;
cout<<&i;
```

ตัวอย่างดังกล่าว โปรแกรมจะแสดง 0x6ffe1c ออกทางหน้าจอ แสดงว่าตัวแปร i ถูกเก็บอยู่ที่ตำแหน่ง 0x6ffe1c ในหน่วยความจำ ดังนั้นเราสามารถใช้ค่าที่อยู่นี้ในการถึงค่าของตัวแปร i ได้โดยตรง

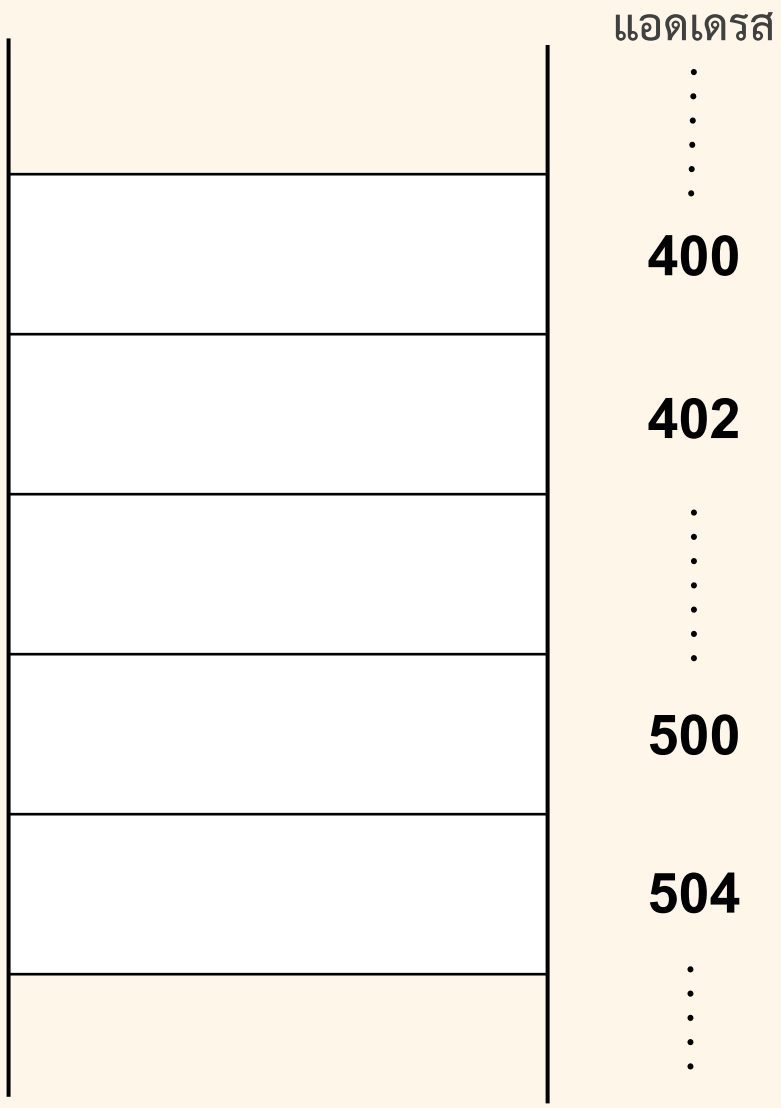
```
Output :
0x6ffe1c
```

โดยปกติเราจะใช้ %x เพื่อแสดงผลค่าที่อยู่ในเลขฐาน 16

```
0x6ffe1c
```

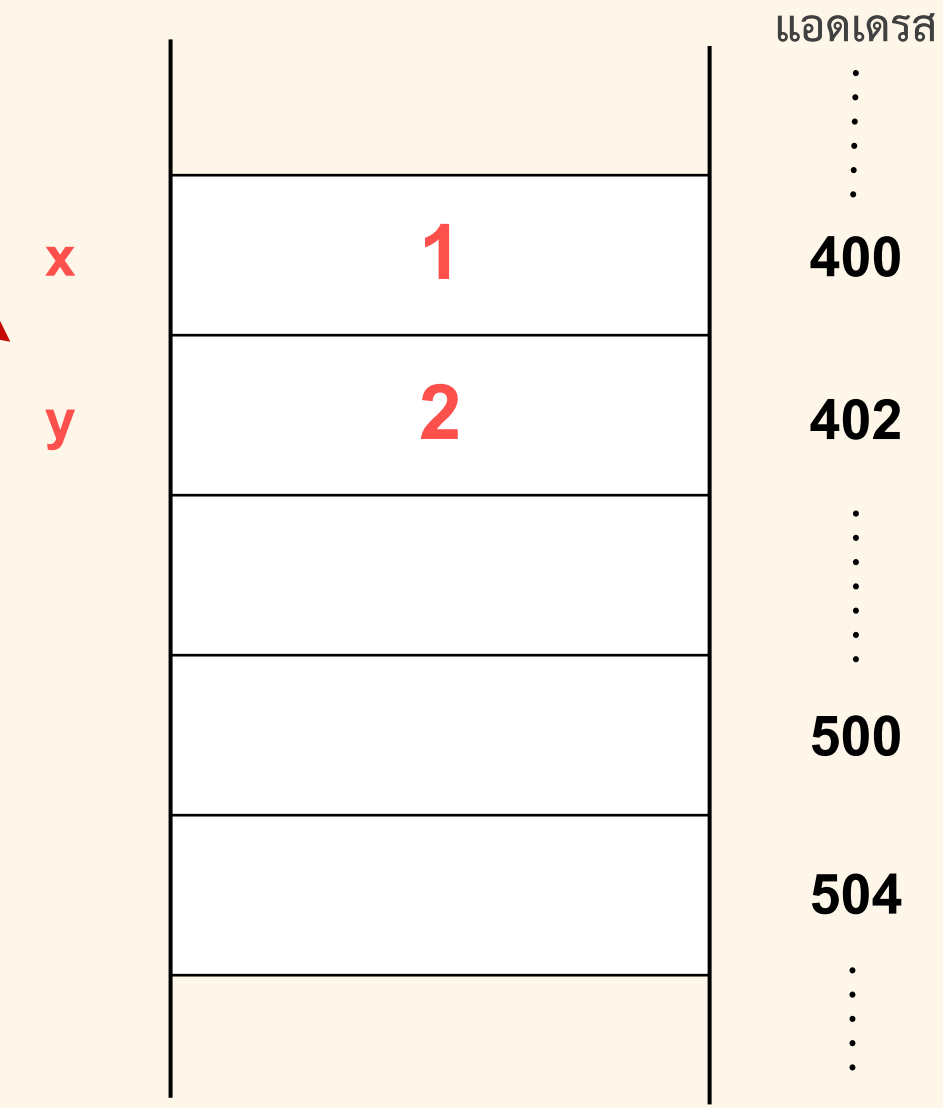
การกำหนดค่าและการอ่านค่าตัวแปรพอยน์เตอร์

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

ip = &x;
y = *ip;
*ip = 0;
y = 5;
ip = &y;
*ip = 3;
iq = ip;
```



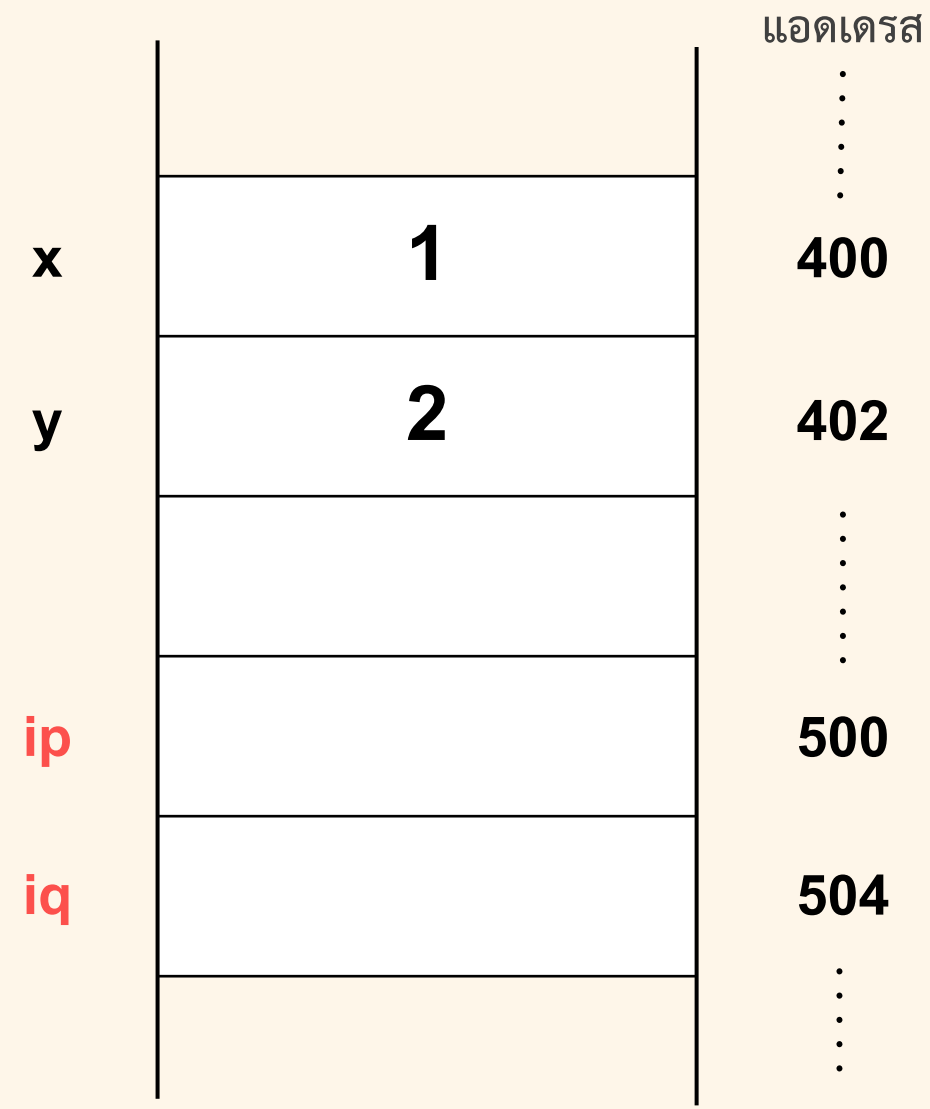
การกำหนดค่าและการอ่านค่าตัวแปรพอยน์เตอร์

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

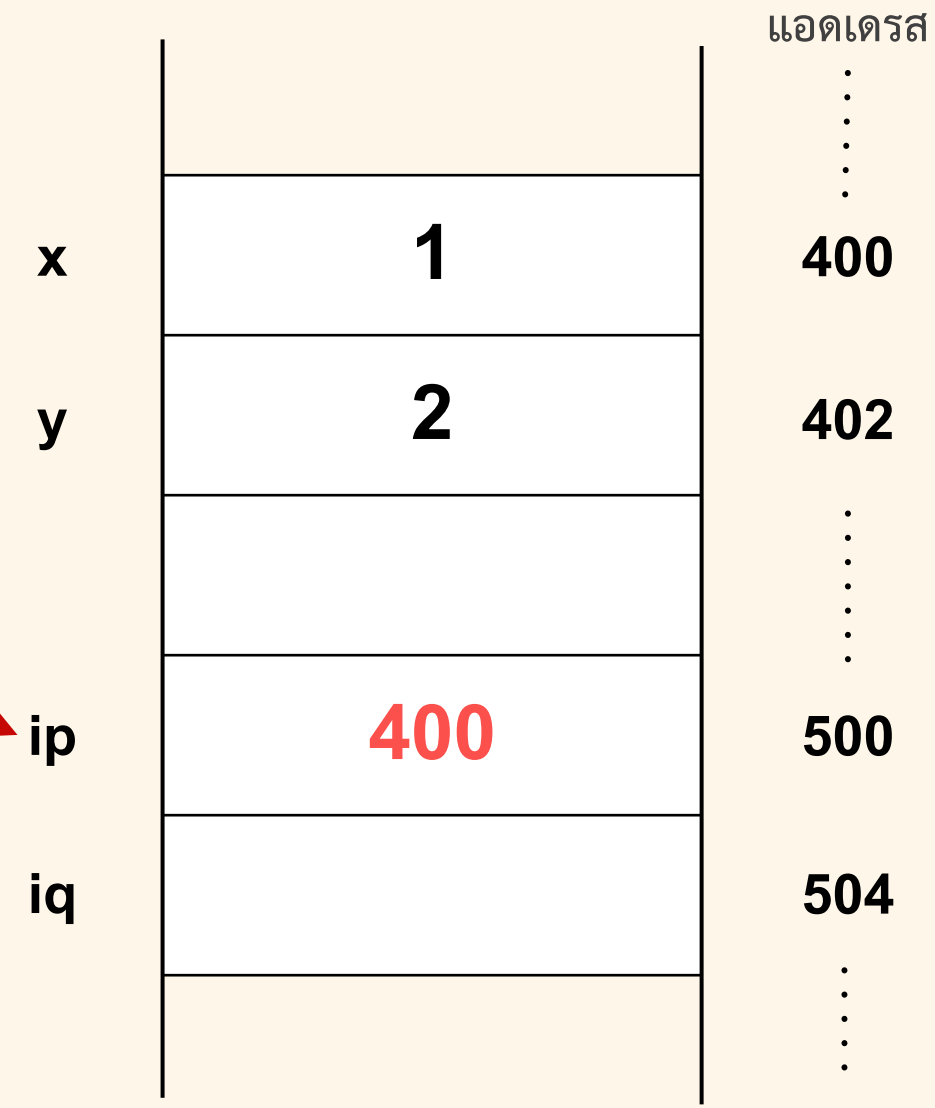


การกำหนดค่าและการอ่านค่าตัวแปรพอยน์เตอร์

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

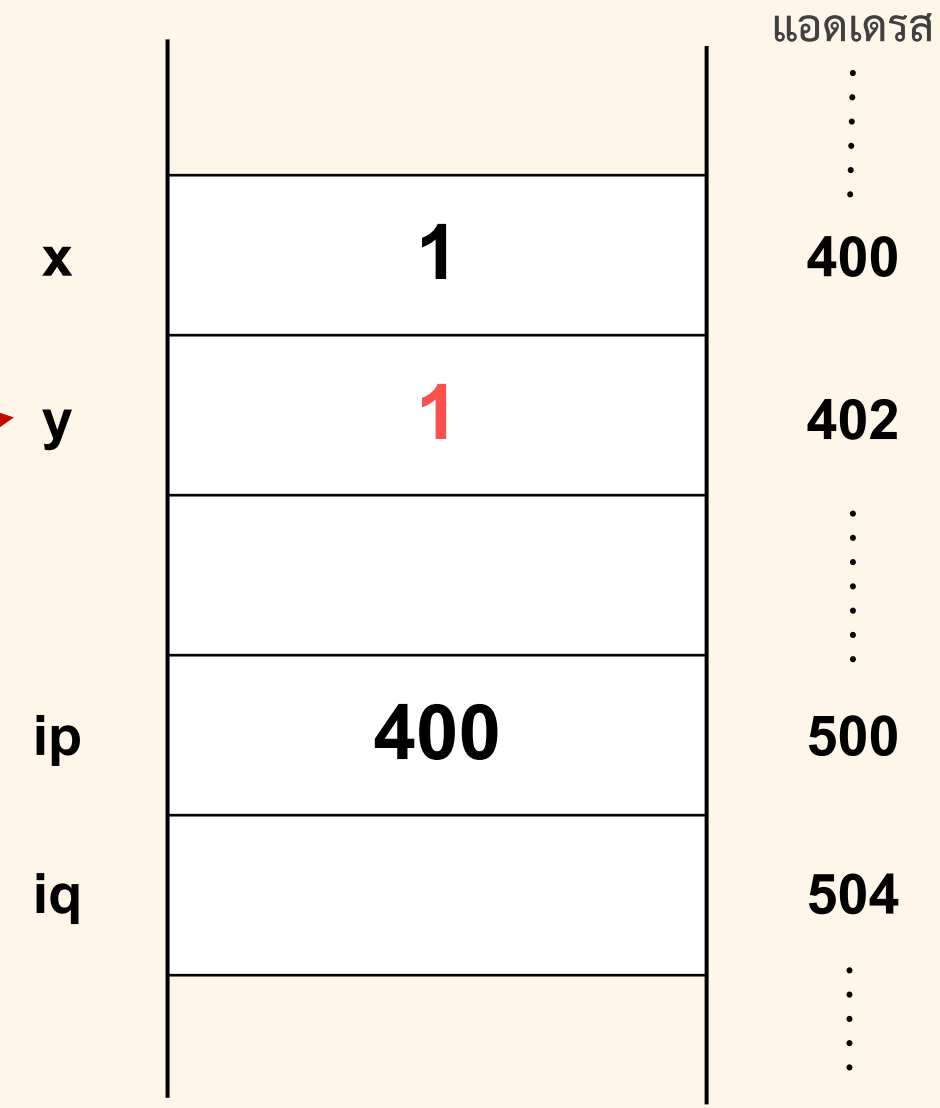


```
int x = 1, y = 2;
int *ip, *iq;

ip = &x;
```



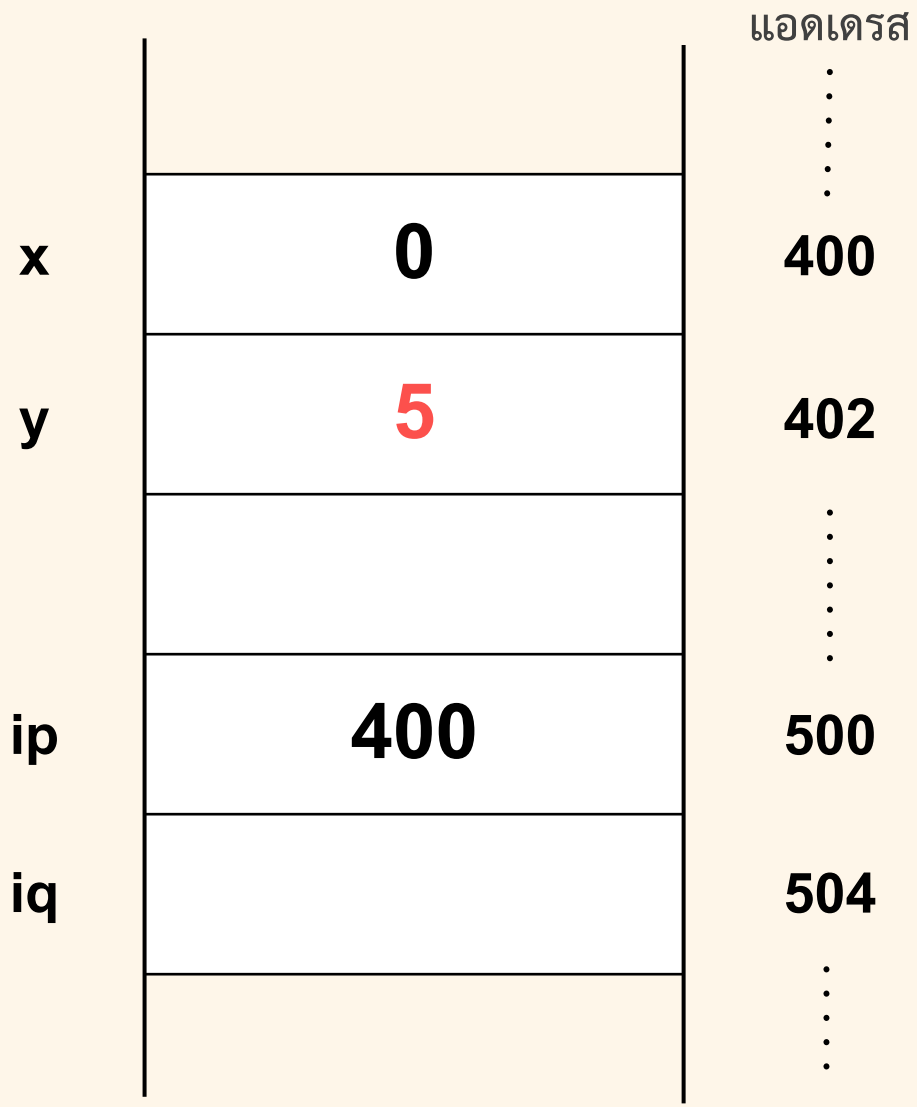
การกำหนดค่าและการอ่านค่าตัวแปรพอยน์เตอร์

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

ip = &x;
y = *ip;
```



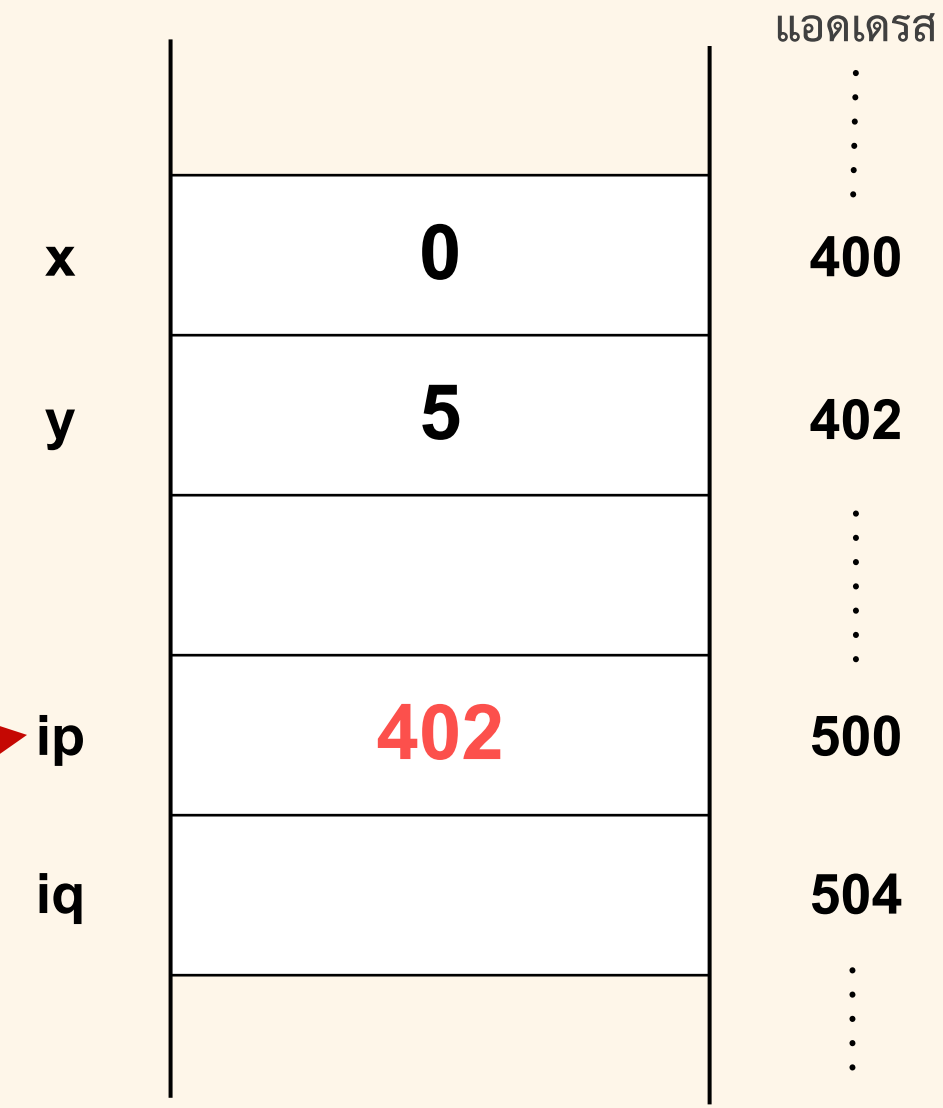
การกำหนดค่าและการอ่านค่าตัวแปรพอยน์เตอร์

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

ip = &x;
y = *ip;
*ip = 0;
y = 5;
```



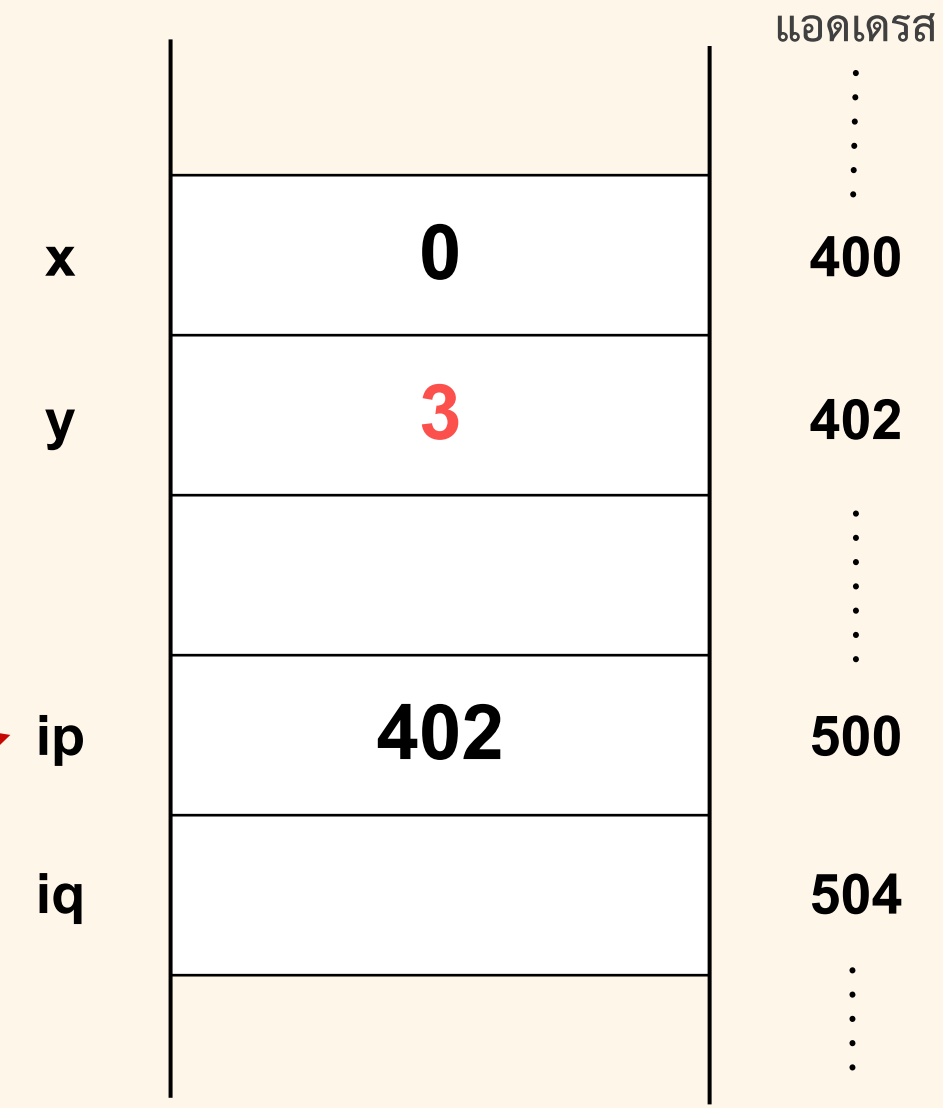
การกำหนดค่าและการอ่านค่าตัวแปรพอยน์เตอร์

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

ip = &x;
y = *ip;
*ip = 0;
y = 5;
ip = &y;
```



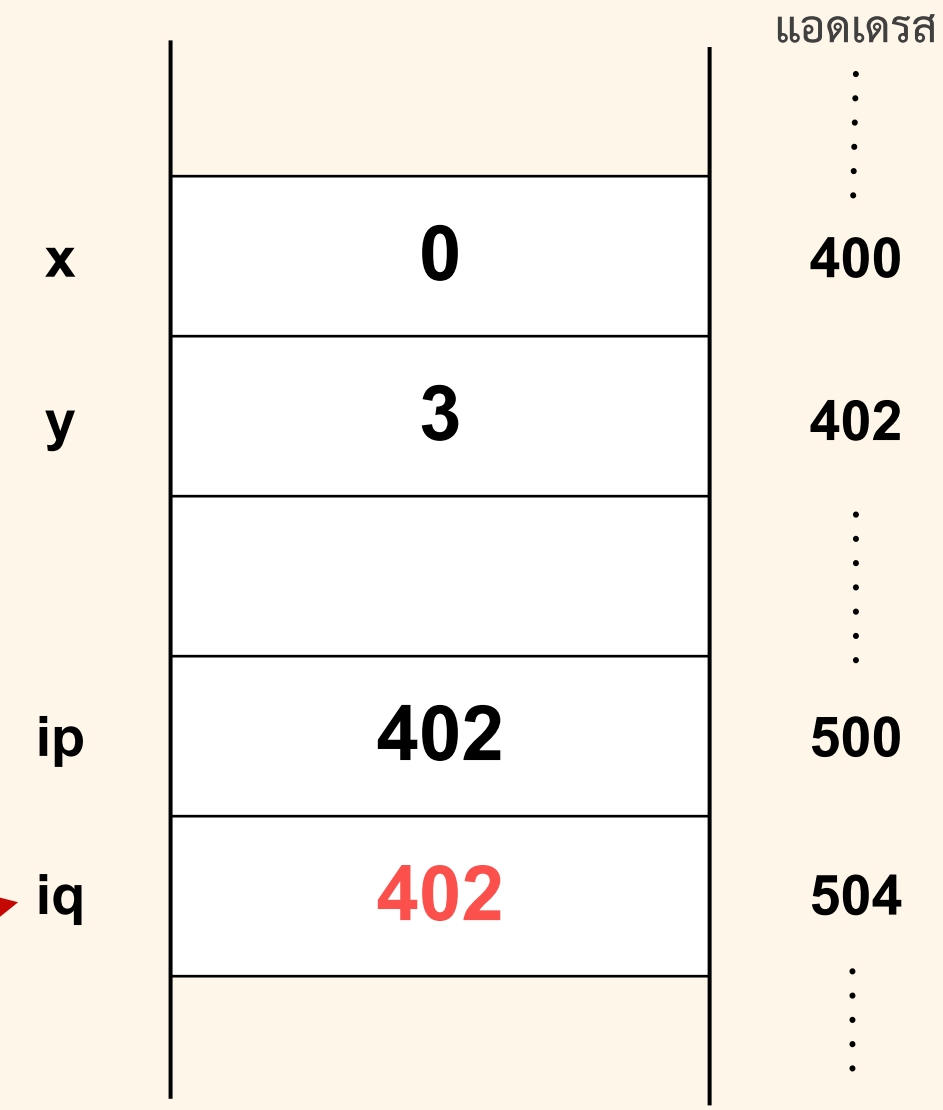
การกำหนดค่าและการอ่านค่าตัวแปรพอยน์เตอร์

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

ip = &x;
y = *ip;
*ip = 0;
y = 5;
ip = &y;
*ip = 3;
```



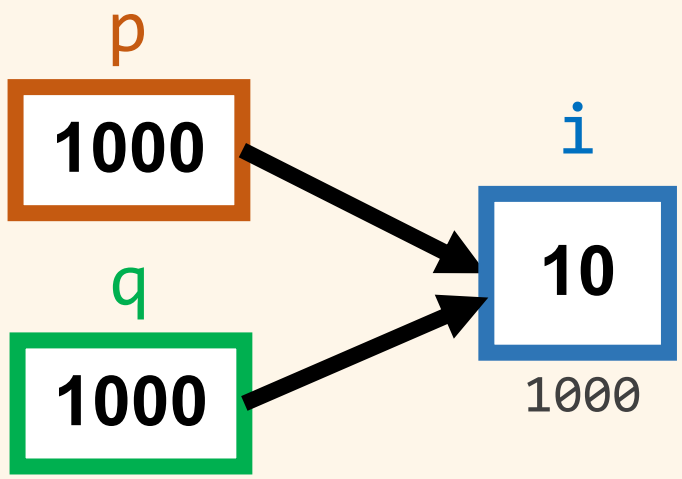
การกำหนดค่าและการอ่านค่าตัวแปรพอยน์เตอร์

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

ip = &x;
y = *ip;
*ip = 0;
y = 5;
ip = &y;
*ip = 3;
iq = ip;
```

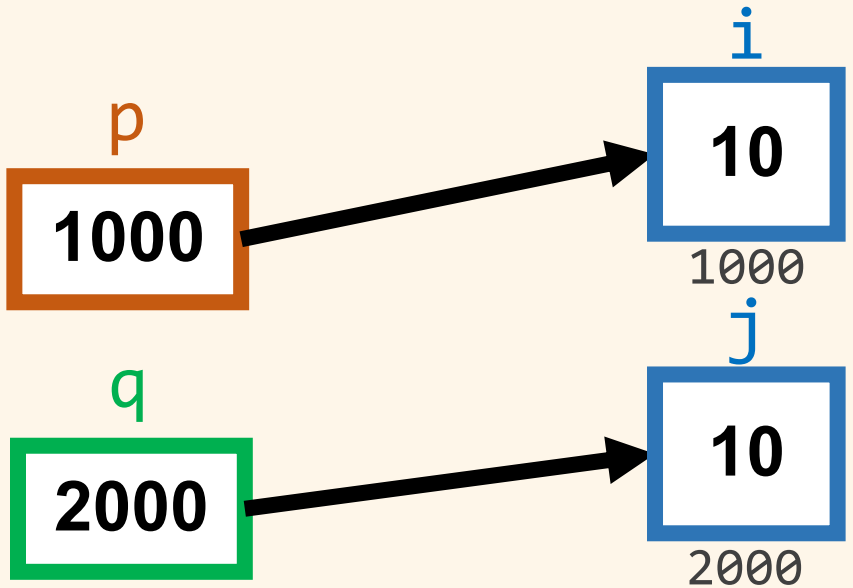


คำถามชวนคิด q = p กับ *q = *p เหมือนกันหรือไม่

```
int i = 10;
int *p, *q;
p = &i;
q = p;
```




```
int i = 10, j = 20;
int *p, *q;
p = &i;
q = &j;
*q = *p;
```



ข้อควรระวังในการใช้ Pointer

ไม่ใช้ indirection operator (*) กับตัวชี้ที่ยังไม่ได้กำหนดค่าเริ่มต้น

```
int *ptr;
cout<<*ptr;
```

กรณีดังกล่าวไม่ถูกต้อง เพราะเราไม่ทราบว่าพอยน์เตอร์ ptr จะชี้ไปที่ใด

```
Output :
Undefined behavior
```

ข้อควรระวังในการใช้ Pointer

ห้ามกำหนดค่าให้กับตัวชี้ที่ยังไม่ได้กำหนดค่าเริ่มต้น

```
int *ptr;
ptr = 1;
```

```
Output :
Segmentation Fault (SIGSEGV)
```

Segmentation Fault เกิดจากการที่โปรแกรมพยายามอ่านหรือเขียนตำแหน่งหน่วยความจำที่ไม่ถูกต้อง

ข้อควรระวังในการใช้ Pointer

- **ไม่ใช้ indirection operator (*) กับตัวชี้ที่ยังไม่ได้กำหนดค่าเริ่มต้น**
- **ห้ามกำหนดค่าให้กับตัวชี้ที่ยังไม่ได้กำหนดค่าเริ่มต้น**

การทำงานกับตัวแปรพอยน์เตอร์ต้องกำหนดค่าเริ่มต้นให้กับตัวแปรพอยน์เตอร์ก่อนเสมอว่าต้องการให้ชี้ไปยัง address ของตัวแปรใด *(ใช้สัญลักษณ์ & ในการกำหนด)* จากนั้นจึงจะสามารถกระทำการใดๆ กับค่าข้อมูลในพอยนต์เตอร์นั้นชี้อยู่ได้ *(ใช้สัญลักษณ์ * ในการจัดการข้อมูล)*

สรุปจุดเด่นของ Pointer

- มีความเร็วในการทำงานสูง
- ประหยัดเนื้อที่ในหน่วยความจำหลักขณะประมวลผลเมื่อเทียบกับอาร์เรย์
- ใช้ตัวชี้ร่วมกับฟังก์ชันเพื่อเพิ่มประสิทธิภาพการเขียนโปรแกรม
- ช่วยให้การเขียนภาษา C มีความยืดหยุ่น เป็นจุดเด่นของภาษา C

พอยน์เตอร์ กับ ฟังก์ชัน

ปัญหาของฟังก์ชันในภาษาซี

ภาษาซีมีการส่งอากิวเมนต์ให้กับฟังก์ชันแบบ Pass by Value

ฟังก์ชันสามารถคืนค่า (return) ค่าได้เพียงหนึ่งค่าเท่านั้น

การใช้พอยน์เตอร์สามารถช่วยแก้ปัญหาเหล่านี้

ส่งค่าแบบ Pass by Reference (ส่งแอดเดรสของตัวแปรไปให้พอยน์เตอร์)

แก้ไขค่าที่ต้องการได้เลย โดยมิต้องส่งค่ากลับ

Pass by Value

คือการส่งค่า (value) เป็นอากิวเมนต์ของฟังก์ชัน
ดังนั้นค่าที่ทำในฟังก์ชันจึงไม่ส่งผลต่อตัวแปรนอกฟังก์ชัน

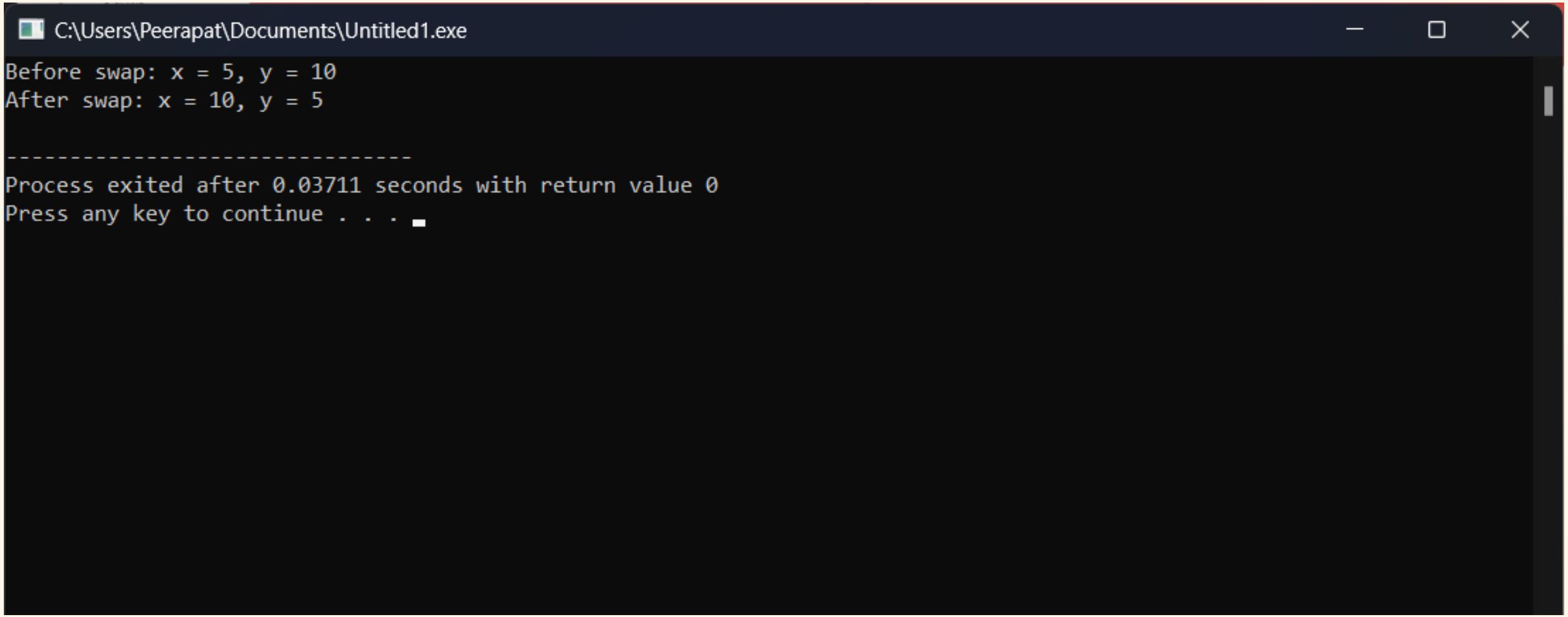
Pass by Reference

คือการส่งตัวแปร (variable) เป็นอากิวเมนต์ของฟังก์ชัน ดังนั้นตัวแปรที่มีการดำเนินการใดๆ ในฟังก์ชัน จะส่งผลให้ตัวแปรนอกฟังก์ชันมีการเปลี่ยนแปลงด้วยโดยมิต้องส่งค่ากลับ

โปรแกรมการสลับค่าตัวแปร 2 ตัวโดยผ่านฟังก์ชัน

- เขียนฟังก์ชันเพื่อสลับค่าของตัวแปร 2 ตัว
- ผลลัพธ์ที่ต้องการจากฟังก์ชันนี้จะมี 2 ค่าของตัวแปรที่ทำการสลับค่า
- ต้องใช้พอยน์เตอร์ช่วย เพราะหากอากิวเมนต์เป็นตัวแปรธรรมดาจะไม่สามารถแก้ปัญหานี้ได้
- โดยการส่งค่าแอดเดรสของตัวแปรทั้ง 2 ให้ฟังก์ชันที่จะสลับค่าของตัวแปรทั้ง 2 ผ่านทางตัวแปรพอยน์เตอร์ที่เป็นอากิวเมนต์ของฟังก์ชั่น

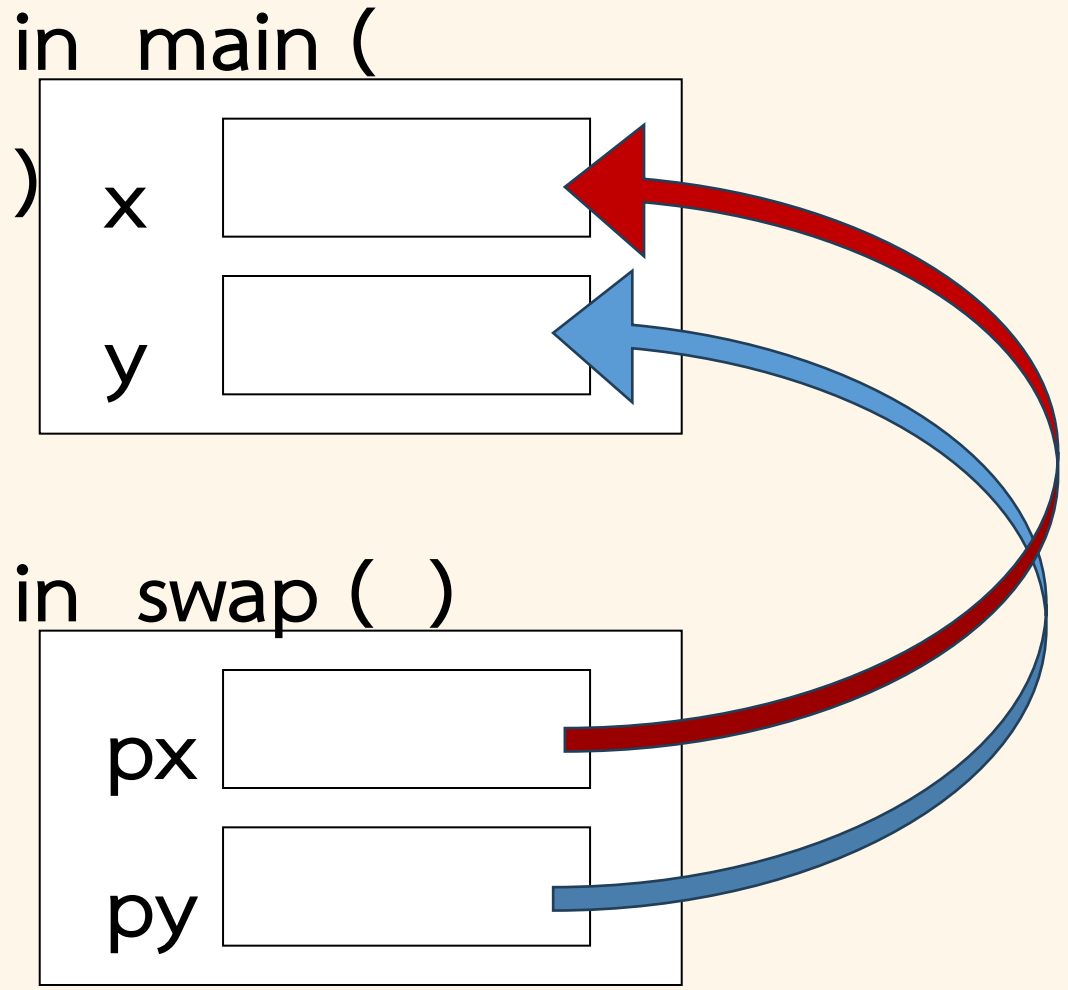
```
#include <iostream>
using namespace std;

void swap(int *, int *);
int main() {
    int x = 5, y = 10;
    cout<<"Before swap: x = "<< x <<", y="<< y<<endl;
    swap(&x,&y);
    cout<<"After swap: x = "<< x<< ", y="<< y<<endl;
    return 0;
}
void swap(int *px, int *py) {
    int temp;
    temp = *px;
    *px = *py;
    *py = temp;
}
```


C:\Users\Peerapat\Documents\Untitled1.exe
Before swap: x = 5, y = 10
After swap: x = 10, y = 5

Process exited after 0.03711 seconds with return value 0
Press any key to continue . . .

อากิวเมนท์ที่เป็นพอยน์เตอร์จะช่วยให้ฟังก์ชันสามารถเปลี่ยนค่าให้กับตัวแปรที่ส่งเข้ามาได้ เนื่องจากอาร์กิวเมนท์นั้นจะเก็บแอดเดรสของตัวแปรที่ส่งเข้ามา เมื่อมีการเปลี่ยนแปลงค่าของอาร์กิวเมนท์ผ่าน Dereferencing Operator (*) ค่าของตัวแปรที่ส่งเข้ามาจะถูกเปลี่ยนค่าพร้อมกันในทันที

ความสัมพันธ์ของการส่งอาร์กิวเมนท์แบบพอยน์เตอร์กับฟังก์ชัน



พอยน์เตอร์ กับ อาร์เรย์

การทำงานใดๆ ของอาร์เรย์ สามารถใช้พอยน์เตอร์เข้ามาช่วย
ซึ่งจะทำให้มีความเร็วในการทำงานสูงขึ้น

```
#include <iostream>
using namespace std;

int main() {
    int i, x[5] = {10, 20, 30, 40, 50};

    for (i = 0; i < 5; i++)
        cout<<"element  "<<i + 1<<" data is " <<x[i]<<endl;

    return 0;
}
```

10	20	30	40	50
x[0]	x[1]	x[2]	x[3]	x[4]
*x	*(x+1)	*(x+2)	*(x+3)	*(x+4)

กำหนดอาร์เรย์ a และพอยน์เตอร์ pa

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

กำหนดให้พอยน์เตอร์ pa ชี้ไปยังอาร์เรย์ a ด้วยคำสั่ง

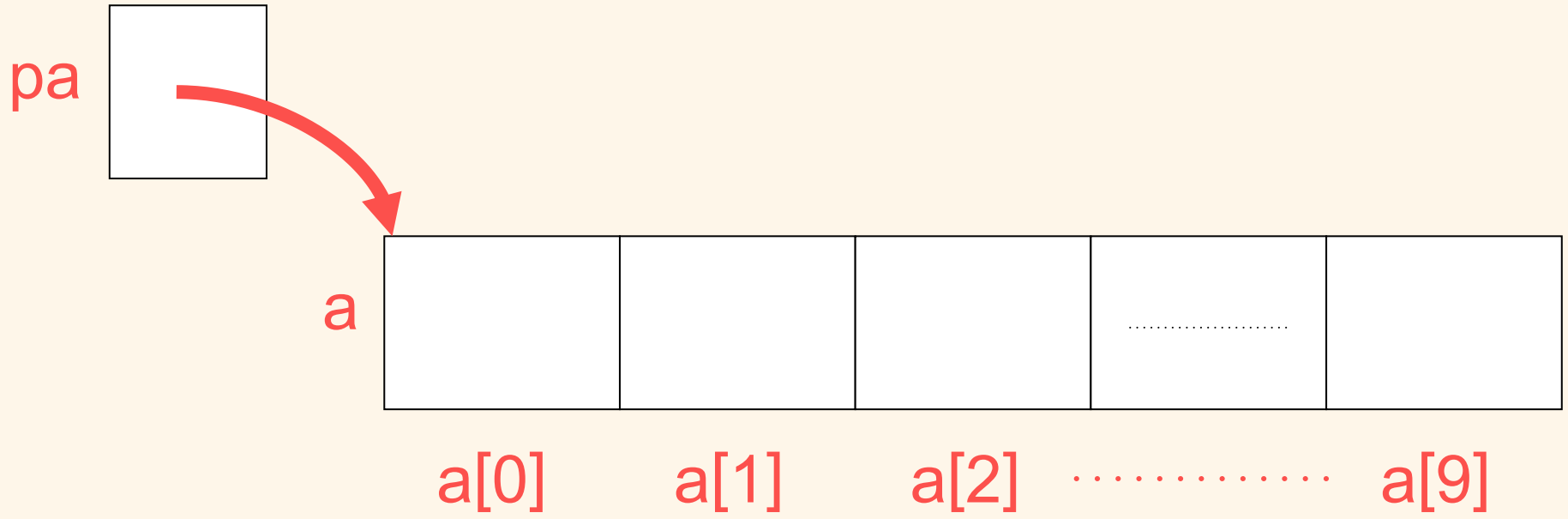
```
pa = &a[0];
```

หรือใช้คำสั่ง

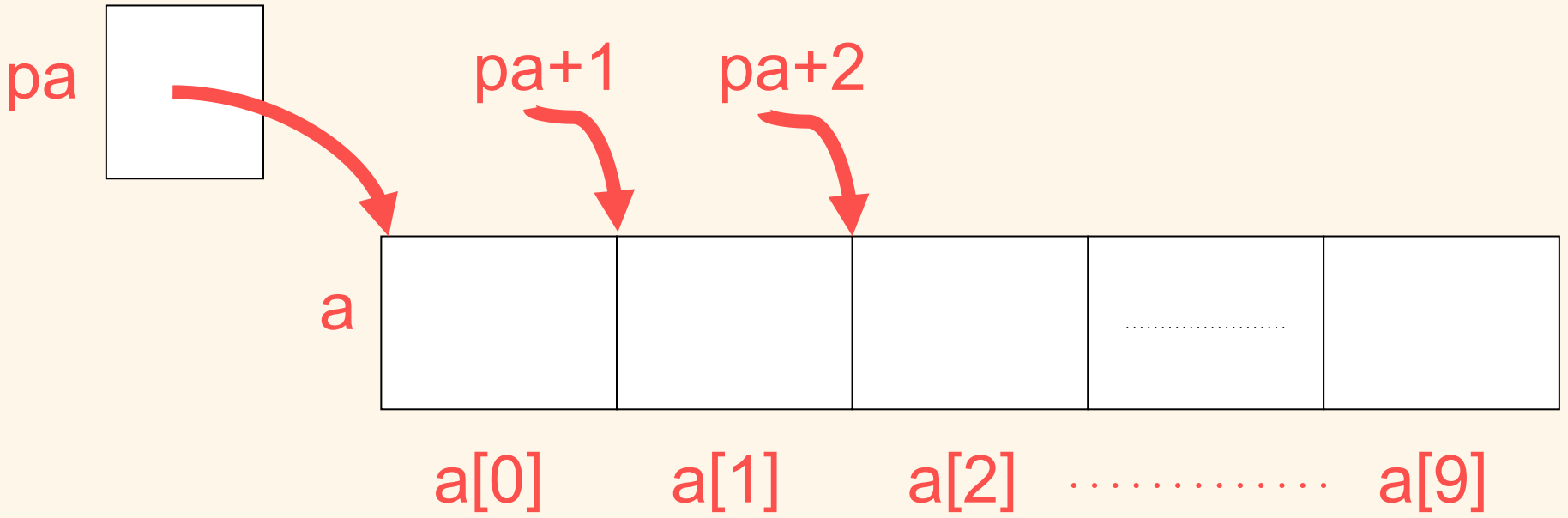
```
pa = a;
```

pa จะเก็บค่าแอดเดรสเริ่มต้นของอาร์เรย์ a

ภาพแสดงพอยน์เตอร์ชี้ไปยังแอดเดรสเริ่มต้นของอาร์เรย์



ภาพแสดงการอ้างถึงตำแหน่งในอาร์เรย์ผ่านพอยน์เตอร์

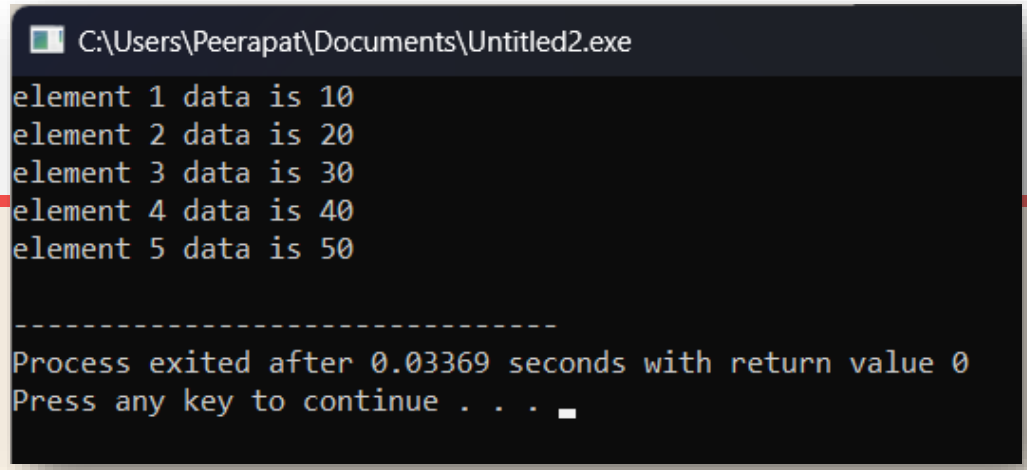


เช่น ต้องการคำนวณตำแหน่ง a[2] ผ่านพอยน์เตอร์ก็คือ pa + 2

การเข้าถึงสมาชิกในอาร์เรย์กระทำได้โดยผ่านตัวกระทำ *pa

ตัวอย่างเช่น x = *(pa+2)

```cpp
#include <iostream>
using namespace std;

int main() {
    int i, x[5] = {10, 20, 30, 40, 50};

    for (i = 0; i < 5; i++)
        cout<<"element  "<<i + 1<<" data is " <<x[i]<<endl;

    return 0;
}
```



การส่งผ่านอาร์เรย์แบบ pass by reference

```
#include <iostream>
using namespace std;

void print(int *pr) {
    for (int i = 0; i < 5; i++)
        cout<<"element" << i + 1<<" data is "<< *(pr+i)<<endl;
}

int main() {
    int i, data[5] = {10, 20, 30, 40, 50};
    print(data);
    return 0;
}
```

แหล่งข้อมูล

- Pointer - ตัวชี้, อ.ดร.ลือพล พิพานเมฆากรณ์ ภาควิชาวิทยาการคอมพิวเตอร์และสารสนเทศ คณะวิทยาศาสตร์ประยุกต์ มหาวิทยาลัยเทคโนโลยีพระจอมเกล้าพระนครเหนือ
- ข้อมูลชนิดอาร์เรย์,สตริง,พอยเตอร์ , เอกสารประกอบการอบรม สอวน.สาขาคอมพิวเตอร์ ศูนย์โรงเรียนสามเสนวิทยาลัย
- พอยน์เตอร์ , อ.กาญจนา ทองบุญนาค สาขาวิชาคอมพิวเตอร์ คณะวิทยาศาสตร์และเทคโนโลยี มหาวิทยาลัยราชภัฏเชียงใหม่
- พอยน์เตอร์ , ผศ.ดร.ณัฐชามญฑ์ ศรีจำเริญรัตนา ภาควิชาคอมพิวเตอร์ธุรกิจ คณะวิทยาการจัดการ มหาวิทยาลัยราชภัฏนครปฐม
- พอยน์เตอร์ (Pointer) , ธนกฤต ขันธวัฒน์ , โรงเรียนร้อยเอ็ดวิทยาลัย
- C++ Language Reference , Microsoft Learn
- C Programming & Data Structures, Neso Academy
- Pointer คืออะไร ?, borntodev.com/