

พอยน์เตอร์ (Pointer)

- นิยาม
- การประกาศตัวแปรพอยน์เตอร์
- การกำหนดและการเรียกใช้ค่าของตัวแปรพอยน์เตอร์
- การดำเนินการกับพอยน์เตอร์



ตัวแปร

ตัวแปร คือชื่อที่ใช้แทนข้อมูล

การประกาศตัวแปรเป็นการกำหนดชื่อเพื่อใช้แทนข้อมูล เมื่อ
เราประกาศตัวแปร จะมีการจดเนื้อที่ในหน่วยความจำเพื่อ
เก็บข้อมูล เราสามารถเข้าถึงข้อมูลได้โดยอ้างถึงตัวแปร การ
ประกาศตัวแปร เช่น int i;

เป็นการประกาศ (Declaration) ตัวแปรชื่อ i เป็นตัวแปรที่เก็บ
ข้อมูลเป็นเลขจำนวนเต็ม



- เป็นชนิดข้อมูลชนิดหนึ่งของภาษาซี
- ค่าข้อมูลชนิดนี้หมายถึง ตำแหน่งที่อยู่ใน
หน่วยความจำ
- มีความเร็วในการทำงานสูง
- การใช้งานพอยน์เตอร์ค่อนข้างซับซ้อน
- พอยน์เตอร์เป็นจุดเด่นอย่างหนึ่งในการเขียนโปรแกรม
ด้วยภาษาซี



ภาพจำลองหน่วยความจำ

int x = 88;
char c = 'A';

x



ตำแหน่ง(address) ในหน่วยความจำ



Why pointers?

- Passing large structures (or a large array) to a function would be inefficient
- We may want to call a function which updates a local variable
- Consider the `scanf()` function:

```
int main(void)
{
    int input;
    scanf("%d", &input);

    printf("You entered %d", input);
}
```

CS112



5

Why pointers?

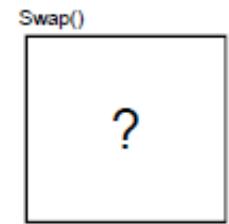
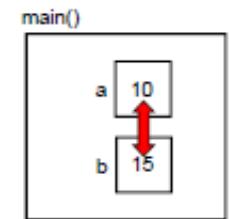
```
int main(void)
{
    int a, b;

    a = 10;
    b = 15;

    Swap(a, b);
}

void Swap(...)

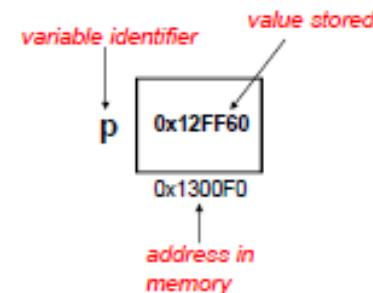
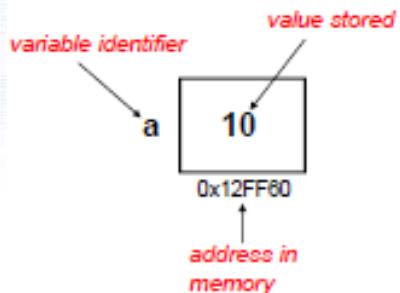
{...}
```



6

What is a pointer?

A *pointer* is a variable that stores an *address* in memory

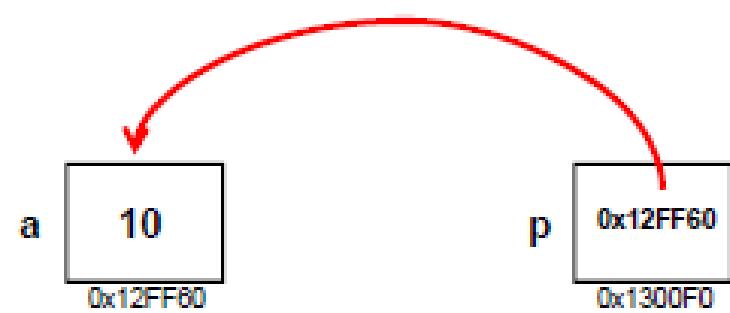


CS112



7

Visualising a pointer



CS112



8

- อีกวิธีที่จะเข้าถึงตัวแปร คือเราจะอ้างถึงตำแหน่งที่เก็บข้อมูล
- พอยน์เตอร์ เป็นชนิดข้อมูลชนิดหนึ่งของภาษาซี
- แตกต่างจากชนิดข้อมูลพื้นฐานอื่น ๆ
- ตัวแปรพอยน์เตอร์** เป็นตัวแปรที่ใช้ เก็บค่าแอดเดรสของตัวแปรอื่น ๆ

การประกาศตัวแปรพอยน์เตอร์

- ใช้การดำเนินการชนิดเอกสาร (Unary Operator) *
- ชื่อเรียกเป็นภาษาอังกฤษว่า Indirection
- หรือ Dereferencing Operator
- รูปแบบคำสั่ง**
- Type `*Variable-name;`
- Type ชนิดของตัวแปร
- * เป็นเครื่องหมายที่แสดงว่า ตัวแปรที่ตามหลังเครื่องหมายนี้เป็นตัวแปรชนิดพอยน์เตอร์
- `Variable-name` ชื่อตัวแปรที่เป็นตัวแปรพอยน์เตอร์

```
int x = 88;
```

```
char c = 'A';
```

x

88

100

204

104

100

p

c

'A'

110

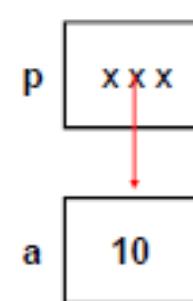
สมมติให้ตัวแปร `p` เป็นตัวแปรพอยน์เตอร์ ซึ่ง เก็บค่าแอดเดรสของตัวแปร `x` (หรือ `p` ซึ่งเป็นตัวแปร `x`)

Declaring a pointer

```
int *p;
```

```
int a = 10;
```

```
p = &a;
```



- ตัวแปรพอยน์เตอร์ เป็นตัวชี้ไปยังตัวแปรชนิดอื่นๆ

- การประกาศชนิดของตัวแปรพอยน์เตอร์ ต้องสอดคล้องกับชนิดของตัวแปรนั้นๆ

▪ เช่น `char *prt;` ประกาศตัวแปร prt ให้เป็นตัวแปรพอยน์เตอร์ ที่ชี้ไปยังตัวแปรชนิด char

▪ `int *ip , *temp;`

ประกาศตัวแปร ip และ temp เป็นตัวแปรพอยน์เตอร์ที่ชี้ไปยังตัวแปรชนิด int

▪ `double *dp;` ประกาศตัวแปร dp

เป็นตัวแปรพอยน์เตอร์ที่ชี้ไปยังข้อมูลแบบ double

CS112



13

การกำหนดค่าและการอ่านค่าตัวแปรพอยน์เตอร์

- การกำหนดค่าให้กับตัวแปรพอยน์เตอร์ เป็นการกำหนด แอดเดรส ของตัวแปรที่มีชนิดข้อมูลสอดคล้องกับชนิดข้อมูลของตัวแปรพอยน์เตอร์
- ใช้ ตัวดำเนินการชนิดเดียว (Unary Operator) & เป็นตัวดำเนินการที่อ้างถึงแอดเดรส
 - ตัวดำเนินการ `&` เป็นเครื่องหมายที่ใช้เมื่อต้องการให้เอกสาร่าดำเนินการที่อยู่ (address) ของตัวแปรที่เก็บในหน่วยความจำอุปกรณ์มาใช้
 - ตัวดำเนินการ `*` เป็นเครื่องหมายที่ใช้เมื่อต้องการให้นำค่าที่อยู่ในตำแหน่งที่ตัวแปรอยู่นั้นข้อมูลอุปกรณ์มาแสดง

CS112

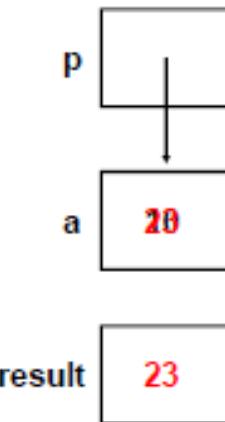


15

Pointer dereferencing

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

p = &a;
*p = 23;
result = *p;
```



CS112

14

ตัวอย่าง

```
int x=10, *y;
y = &x;
```

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

CS112



16

Warning

```
int a = 1;  
int b = 2;  
int c = 3;
```

Example

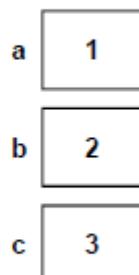
The most common pointer bug:

```
int *p;  
*p = 23;
```

never dereference
a bad pointer!



```
int *p;  
int *s;  
  
p = &b;  
s = p;  
  
a = *p + *s;  
  
p = &c;  
*s = *p;  
  
*p = 2;
```



CS112



17

การดำเนินการกับพอยน์เตอร์

- ตัวแปรพอยน์เตอร์สามารถใช้เครื่องหมายทางคณิตศาสตร์มากระทำได้ เช่นเดียวกับตัวแปรทั่ว ๆ ไป แต่ไม่สามารถใช้ได้ทั้งหมด เครื่องหมายทางคณิตศาสตร์ที่ใช้ได้ คือ
 - เครื่องหมาย + คือ การบวก
 - เครื่องหมาย - คือ การลบ
 - เครื่องหมาย ++ คือ การเพิ่มค่าครั้งละ 1 หน่วย
 - เครื่องหมาย -- คือ การลดค่าครั้งละ 1 หน่วย

CS112



19

Null pointers

A null pointer is a pointer which doesn't point anywhere

```
int *p;
```

```
p = NULL;
```



Don't try to dereference a null pointer

CS112



18

ตัวอย่างชุดคำสั่ง

```
char c = 'A', *pc = &c;  
int x=100, *pi=&x;
```

```
printf("c=%c,\t&c=%X, size of memory is %d byte\n", c, &c, sizeof(c));  
printf("*pc=%c,\tpc=%X, &pc=%X, size of memory is %d byte\n", *pc, pc,  
    &pc, sizeof(pc));  
printf("++c = %c, ++pc=%X\n", ++c, ++pc);  
printf("x=%d,\t&x=%X, size of memory is %d byte\n", x, &x, sizeof(x));  
printf("*pi=%d,\tpi=%X, &pi=%X, size of memory is %d byte\n", *pi, pi,  
    &pi, sizeof(pi));  
printf("++x = %d, ++pi=%X\n", ++x, ++pi);  
printf("pi = %X, *pi++ = %d, pi = %X\n", pi, *pi++, pi);
```

CS112

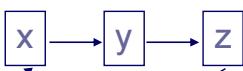


20

- จะเขียนชุดคำสั่ง เพื่อทำการสลับค่าของตัวแปร 3 ตัว x, y และ z ซึ่งมีชนิดข้อมูล เป็นเลขจำนวนเต็ม โดยใช้ตัวแปรพอยน์เตอร์

Hint : char x, y, z, *px, *py, *pz;

โดยให้การสลับค่าดังนี้



ทดลองโปรแกรมต่อไปนี้

```

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

void main() {
    int x=5, y=10;

    printf("Before: x=%d y=%d\n", x, y);
    swap(x, y);
    printf("x= %d y = %d\n", x, y);

    void swap(int x, int y) {
        int temp;
        temp = x; x = y; y = temp;
    }
  
```

การส่งผ่านข้อมูลแบบ Call by value

- ค่าของอาร์กิวเม้นต์จะถูก copy ให้เป็นค่าของพารามิเตอร์ในฟังก์ชันที่ถูกเรียก
- การเปลี่ยนแปลงค่าของพารามิเตอร์ในฟังก์ชันที่ถูกเรียก จะไม่มีผลต่อ ค่าของอาร์กิวเม้นต์ในส่วนของฟังก์ชันที่เรียกไปยังฟังก์ชันนั้นๆ

Call by reference

- ค่าของอาร์กิวเม้นต์จะมีชนิดข้อมูลเป็นพอยน์เตอร์ (pointer) หรือตำแหน่งในหน่วยความจำ

Call by reference

- ค่าของอาร์กิวเม้นต์จะมีชนิดข้อมูลเป็นพอยน์เตอร์ (**pointer**) หรือตำแหน่งในหน่วยความจำ
- รูปแบบการกำหนดฟังก์ชัน
 - จะคล้ายกับการสร้างฟังก์ชันที่มีการเรียกใช้แบบ **Call by value** แต่จะต่างกันที่ ชนิดของพารามิเตอร์ และ อาร์กิวเม้นต์ จะต้องเป็นพอยน์เตอร์
 - ฟังก์ชันลักษณะนี้ ใช้ในการนี้ที่มีการเปลี่ยนแปลงค่าข้อมูล มากกว่า 1 ค่า
- ตัวอย่าง เช่น การเรียกใช้ฟังก์ชัน **scanf()**

ทดลองโปรแกรมต่อไปนี้

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

int main() {
    int x=5, y=10;

    printf("Before: x=%d y=%d\n", x, y);
    swap(&x, &y);
    printf("x= %d y = %d\n", x, y);
    return 0;
}

void swap(int *x, int *y) {
    int temp;
    temp = *x; *x = *y; *y = temp;
}
```

204112
25

ตัวอย่าง (ต่อ) ปรับให้เป็นฟังก์ชัน

```
void shift (char *pa, char *pb, char *pc) {
    char temp;
    temp = *pa;
    *pa = *pb;
    *pb = *pc;
    *pc = temp;
}

int main () {
    char a='x', b='y', c='z';
    printf("before doing the left-shift operation...");
    printf("a=%c\ tb=%c\ tc=%c\n", a, b, c); /*x y z are printed*/
    shift (&a, &b, &c); /*call shift with address of a, b and c*/
    printf("after that..");
    printf("a=%c\ tb=%c\ tc=%c\n", a, b, c); /*y z x are printed*/
}
```

WJ

CS112
27

ตัวอย่าง ๑ จงเขียนโปรแกรมทำการเลื่อนค่าของตัวแปรเดียวอักระ

3 ตัว โดยใช้ พอยน์เตอร์

```
int main () {
    char a='x', b='y', c='z', *pa=&a, *pb=&b, *pc=&c, temp;
    printf("before doing the left-shift operation...");
    printf("a=%c\ tb=%c\ tc=%c\n", a, b, c); /*x y z are printed*/
    temp = *pa;
    *pa = *pb;
    *pb = *pc;
    *pc = temp;
    printf("after that..");
    printf("a=%c\ tb=%c\ tc=%c\n", a, b, c); /*y z x are printed*/
}
```

เปลี่ยนเป็นการเรียกใช้ฟังก์ชัน shift(&a, &b, &c);

WJ

CS112
26

ตัวอย่าง 3

โปรแกรมต่อไปนี้ทำการเรียงลำดับอักระ 3 ตัว

```
void swap (char *x, char *y)
{
    char tmp;
    tmp = *x; *x = *y; *y = tmp;
}
// Function to order 3 character variables
void order (char *cp1, char *cp2, char *cp3)
{
    if (*cp1 > *cp2)           /*First get the larger of cp1 and cp2 into cp2*/
        swap(cp1, cp2);
    if (*cp2 > *cp3)           /*Next get the largest of the three into cp3*/
        swap(cp2, cp3);
    if (*cp1 > *cp2)           /*Now get the second Largest into cp2*/
        swap(cp1, cp2);
}
int main ()
{
    char c1, c2, c3;
    printf("Sort 3 characters using pointers.\n");
    printf("Enter any 3 characters, or incomplete input to quit.\n");
    while (scanf("%c%c%c", &c1, &c2, &c3) == 3) {
        putchar('>');
        order(&c1, &c2, &c3);
        printf("In ascending order : %c %c %c\n", c1, c2, c3);
    }
    return 0;
}
```

WJ

CS112
28



Pointer to pointer

- <type> ** var [= address of pointer] ;
- int a, *pa = &a, **ppa = &pa;
- *pa = 8; /* equivalent to a = 8 */
- **ppa = *pa + 10;
- printf(" **ppa = %u\n", **ppa);
- printf(" *ppa = %u\n", *ppa);
- printf(" ppa = %u\n", ppa);
- printf(" pa = %u\n", pa);
- printf("*pa = %u\n", *pa);