ประเภทตัวแปรขั้นสูง 1

Advanced Variable Type

เอกสารประกอบการสอนวิชา CP111

มหาวิทยาลัยศรีนครินทรวิโรฒ

เนื้อหา

Agenda

- Array
- String
- Pointer

อาเรย์

Array

รู้จักกับอาเรย์

The Introduction to Array

- ใช้สำหรับการประกาศตัวแปรเป็นชุด โดยแต่ละตัวในชุดนั้นเป็นตัวแปรประเภทเดียวกัน
- ตัวอย่างเช่นถ้าเราต้องการตัวแปรประเภท **int** จำนวน 100 ตัว เราสามารถประกาศว่า **int y[100]** โดยที่เราสามารถเข้าถึงตัวแปรแต่ละตัวได้ด้วย **y[i]** โดยที่ **i** มีค่าตั้งแต่ 0 ถึง 99

รู้จักกับอาเรย์

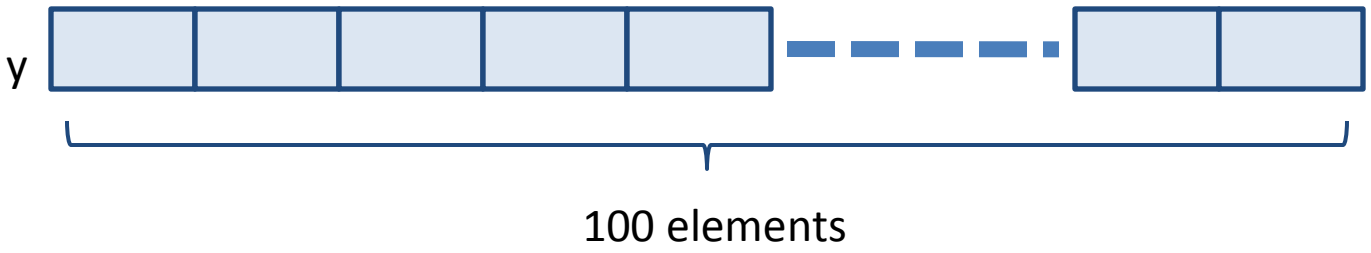
The Introduction to Array

- ตัวแปรแบบทั่วๆ ไป

```
int x1, x2, x3;
```

- ตัวแปรแบบ array

```
int y[100];
```

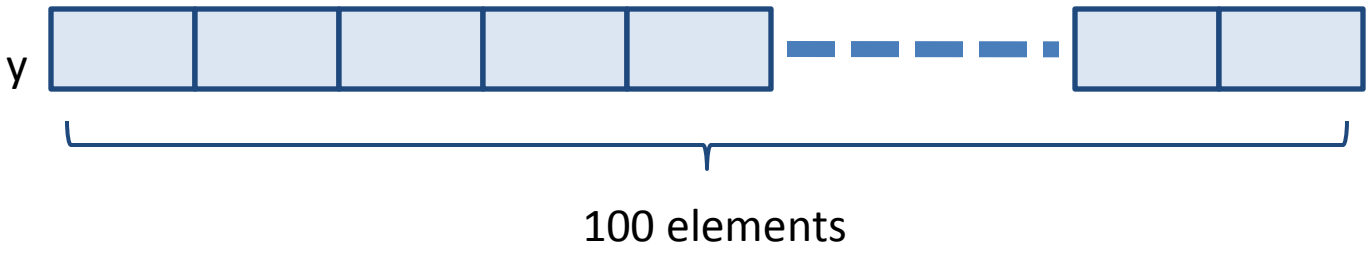


การประกาศตัวแปรแบบอาเรย์

The Declaration of an Array Variable Type

- ลักษณะคำสั่งเหมือนกับการประกาศตัวแปรแบบทั่วๆ ไป แต่จะมี [size] ต่อท้าย โดย size แทนจำนวนตัวแปรที่ต้องการ
- ตัวอย่าง
  - การประกาศตัวแปรประเภท int จำนวน 100 ตัว

```
int y[100];
```



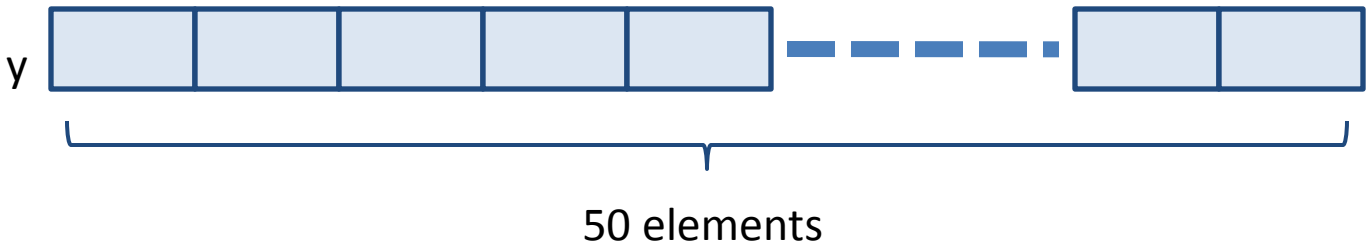
การประกาศตัวแปรแบบอาเรย์

The Declaration of an Array Variable Type

❑ ตัวอย่าง

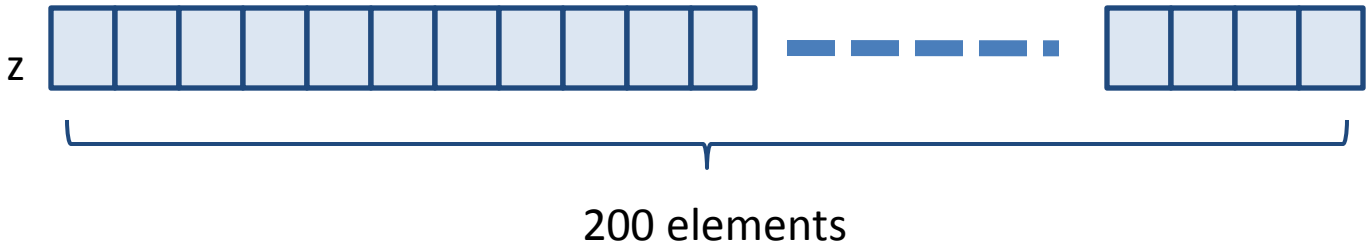
- **ตัวแปรประเภท int จำนวน 50 ตัว**

```
int y[50];
```



- **ตัวแปรประเภท short จำนวน 200 ตัว**

```
short z[200];
```



การกำหนดค่าเริ่มต้นตัวแปรแบบอาเรย์

The Declaration of an Array Variable Type

- การกำหนดค่าขณะประกาศตัวแปรเรียกว่าการกำหนดค่าเริ่มต้น (variable initialization)
  - ตัวอย่าง

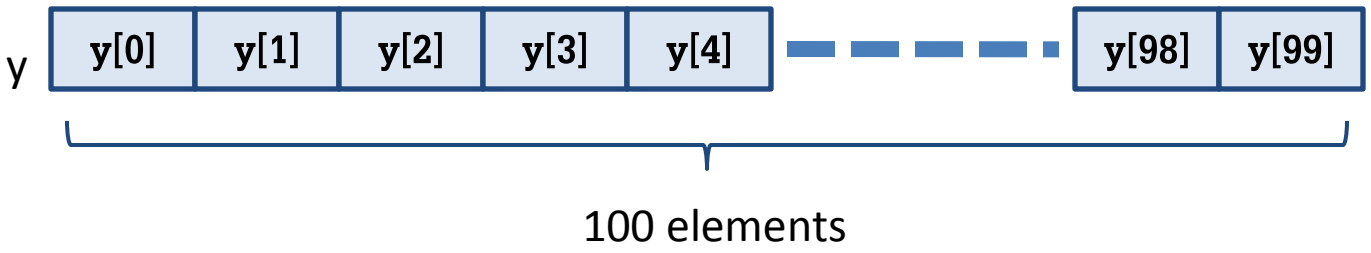
```
int a[5] = {1,2,3,4,5};
```


```
int b[5] = {1,2,3,4};
```

การเข้าถึงตัวแปรแบบอาเรย์

Accessing an Array Variable Type

- ตัวแปรแต่ละตัวใน **Array** สามารถเข้าถึงได้โดยใช้สัญลักษณ์ [i] โดยที่ **i** เป็นตัวที่ต้องการเข้าถึง (เราเรียก **i** ว่า **index**)
- ตัวอย่างการอ้างอิงถึงตัวแปร ณ ตำแหน่งต่างๆ จากการประกาศตัวแปรว่า **int y[100];**

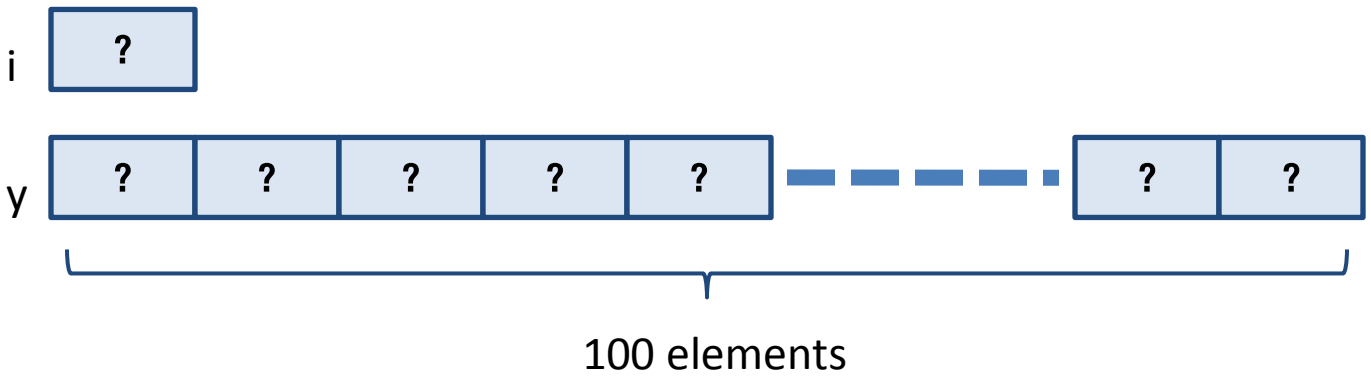


การเข้าถึงตัวแปรแบบอาเรย์

Accessing an Array Variable Type

❑ ตัวอย่างการกำหนดค่าให้กับสมาชิกใน **Array**

```
int y[100], i;

for (i = 0; i < 100; i++)

    y[i] = i;
```

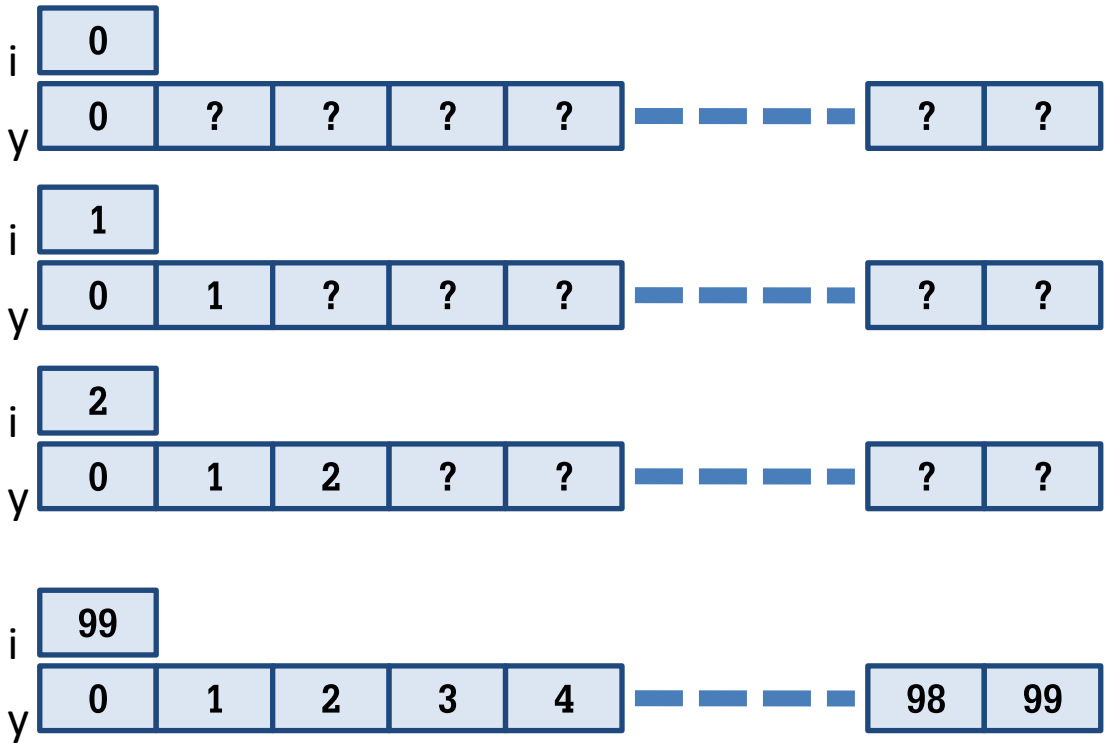


การเข้าถึงตัวแปรแบบอาเรย์

Accessing an Array Variable Type

❑ ค่าของตัวแปร y ณ i ต่างๆ

```
int y[100], i;
for (i = 0; i < 100; i++)
   y[i] = i;
```

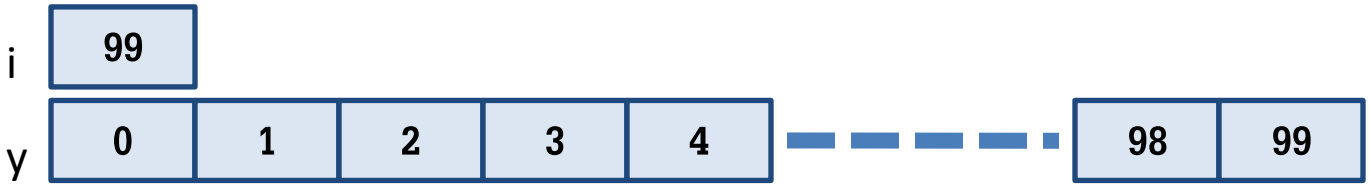


การเข้าถึงตัวแปรแบบอาเรย์

Accessing an Array Variable Type

- ตัวอย่างการพิมพ์ค่าของสมาชิกใน Array

```
...
for (i = 0; i < 100; i++)
    printf("%d ", y[i]);
```



output

```
0 1 2 3 4 5 6 7 8 9 10 11 12 13 ... 99
```

ตัวอย่าง

An example

ex06_01.c

```
...

void main() {
   int data[11];
   int total, a, i;

   for (i = 0; i < 11; i++)
      data[i] = 0;

   printf("The number of data : ");
   scanf("%d", &total);

   printf("Data (range 0 - 10)\n");
   for (i = 0; i < total; i++) {
      printf("%d : ", i + 1);
      scanf("%d", &a);
      data[a]++;
   }

   for (i = 0; i < 11; i++)
      printf("data[%d] : %d\n", i, data[i]);
}
```

กำหนดค่าเริ่มต้น

รับค่าเข้ามาเก็บไว้ในตัวแปรอาเรย์ที่ใช้นับความถี่

อาเรย์หลายมิติ

Multi-dimensional Array

- เราสามารถประกาศอาเรย์ขนาดหลายมิติได้เช่น

  int mat[3][4];

- โดยสมาชิกในอาเรย์สามารถถูกอ้างอิงได้ตามแผนภาพดังนี้

mat[0][0]	mat[0][1]	mat[0][2]	mat[0][3]
mat[1][0]	mat[1][1]	mat[1][2]	mat[1][3]
mat[2][0]	mat[2][1]	mat[2][2]	mat[2][3]

อาเรย์หลายมิติ

Multi-dimensional Array

- การกำหนดค่าเริ่มต้นก็ทำในทำนองเดียวกันคือ

```
int mat[3][4] = { {1,2,3,4}, {5,6,7,8}, {9,10,11,12} };
```

1	2	3	4
5	6	7	8
9	10	11	12

String

รู้จักกับ String

The Introduction to String

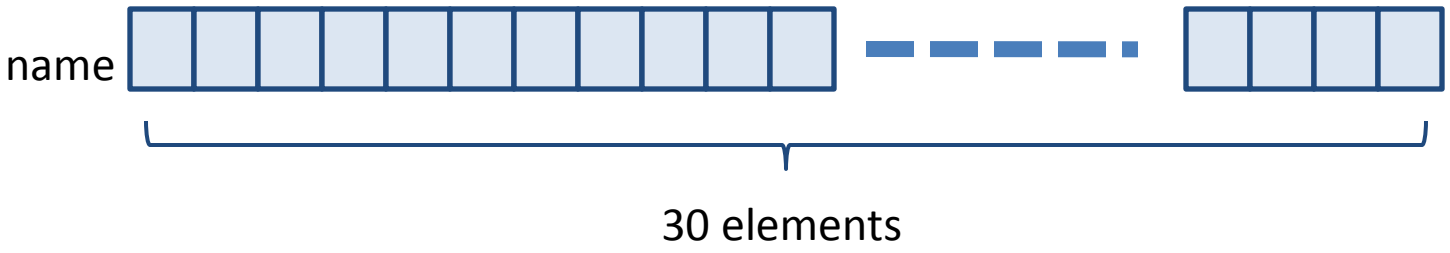
- String คือ Array ของตัวแปรประเภท char
- Null Terminated String คือ String ที่สิ้นสุดด้วย Null ซึ่งสามารถเขียนแทนได้ด้วย '\0' (ASCII code 0x00)
- String ที่ใช้กันทั่วๆ ไปในภาษาซีคือ null terminated string

การประกาศตัวแปรประเภท String

The Declaration of a String Variable Type

- **String คือ Array ของตัวแปรประเภท char ฉะนั้นสามารถประกาศได้ดังนี้**

  char name[30];



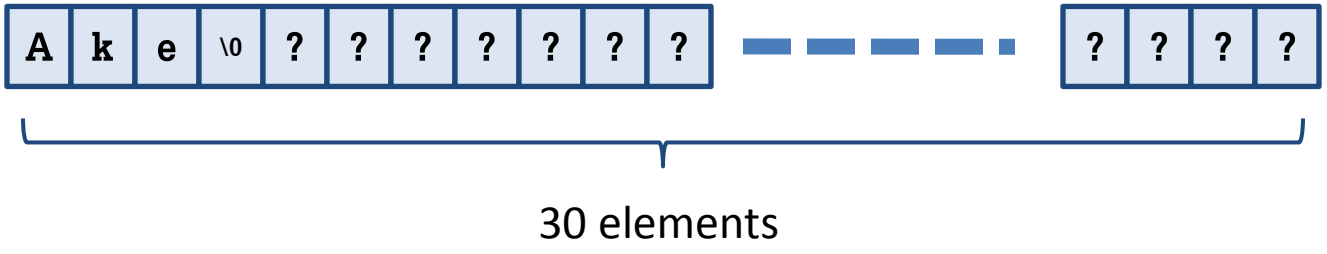
- **ตัวแปร name ที่ประกาศนี้ จะสามารถเก็บตัวอักษรได้ 29 ตัวอักษร (เพราะต้องมีตัวสุดท้ายคือ '\0')**

การกำหนดค่าเริ่มต้นตัวแปรประเภท String

Accessing a String Variable Type

- การกำหนดค่าเริ่มต้นสามารถทำได้หลายวิธี ซึ่งให้ผลเหมือนกัน

```
char namea[30] = "Ake";
char nameb[30] = {'A', 'k', 'e', '\0'};
// The following is not the initialize of a string, it is a normal assignment
char namec[30];
namec[0] = 'A'; namec[1] = 'k'; namec[2] = 'e'; namec[3] = '\0';
```



การเข้าถึงตัวแปรประเภท String

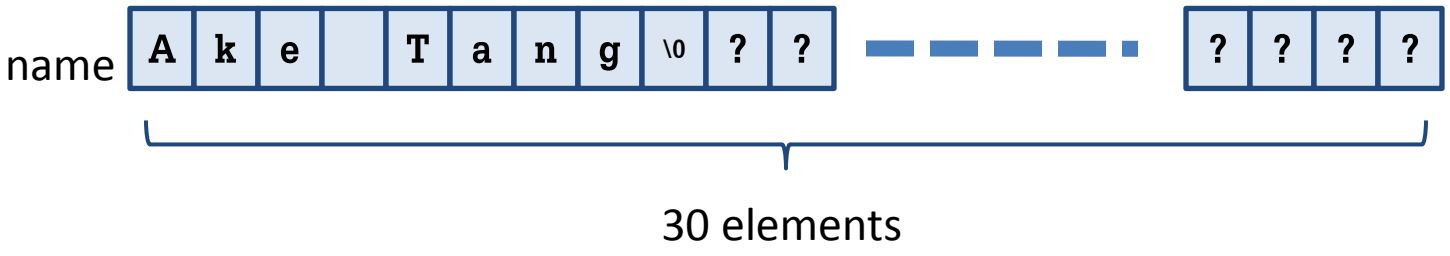
Accessing a String Variable Type

- สามารถอ้างอิงทำได้เหมือนกับการเข้าถึง Array ตามปกติ

```
char name[30] = "Ake Tang";
int i;
for (i = 0; i < 30; i++) {
    if (name[i] == '\0')
        break;
    printf("name[%d] : %c\n", i, name[i]);
}
```

output

```
name[0] : A
name[1] : k
name[2] : e
name[3] :
name[4] : T
name[5] : a
name[6] : n
name[7] : g
```



การเข้าถึงตัวแปรประเภท String

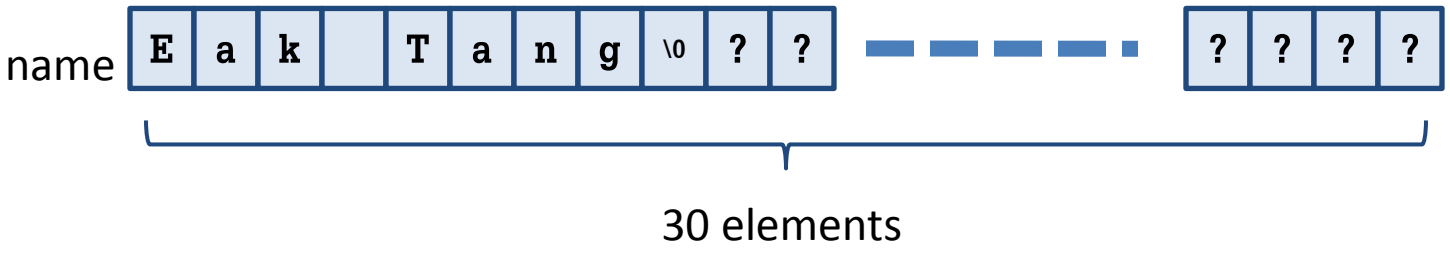
Accessing a String Variable Type

- การแก้ไขค่าก็ทำได้ตามปกติเช่นกัน

```
char name[30] = "Ake Tang";
int i;
printf("%s\n", name);
name[0] = 'E';
name[1] = 'a';
name[2] = 'k';
printf("%s\n", name);
```

output

```
Ake Tang
Eak Tang
```



ข้อควรระวัง

Common mistakes

- อย่าสับสนระหว่างการกำหนดค่าเริ่มต้น (initialize) กับการกำหนดค่า (assign)
  - การกำหนดค่าเริ่มต้น คือการกำหนดค่าให้กับตัวแปรทันทีที่มีการประกาศตัวแปร
  - การกำหนดค่าทั่วๆ ไปนั้น คือการกำหนดค่าหลังจากที่ตัวแปรได้ถูกประกาศแล้ว

ข้อควรระวัง

Common mistakes

- ลักษณะของการเขียนดังต่อไปนี้ เป็นรูปแบบของการกำหนดค่าเริ่มต้น ไม่สามารถใช้กำหนดค่าแบบธรรมดาได้
  - int a[5] = {1, 2, 3, 4, 5}
  - char name[20] = {'A', 'k', 'e', '\0'}
  - char surname[20] = "Tangkananond"

ข้อควรระวัง

Common mistakes

- การเขียนดังต่อไปนี้จะคอมไพล์ไม่ผ่าน
  - char surname[20];

    surname = "Tangkananond";
  - char name[20];

    name = {'A', 'k', 'e', '\0'};

Pointer

รู้จักกับ Pointer

The Introduction to Pointer

- เป็นตัวแปรที่เก็บตำแหน่งของตัวแปรอีกตัวหนึ่ง
- ในอีกความหมายหนึ่งคือเป็นตัวแปรที่เอาไว้ชี้ตัวแปรอีกตัวหนึ่ง
- ในการประกาศ อาจจะบอกประเภทของตัวแปรที่จะชี้ไปหาด้วย

```
int * pInt; // A pointer to int variables

float * pFloat; // A pointer to float variables

int ** ppInt; // A pointer to pointers of int variables
```

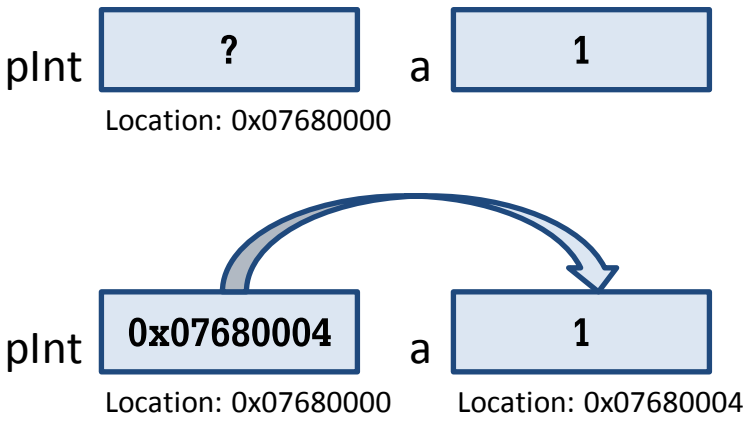
ตัวดำเนินการของ Pointer

Pointer operators

- **address operator, reference operator (&)**
  - ให้ค่าตำแหน่งของตัวแปรปัจจุบัน
- **indirection operator, dereference operator (*)**
  - ให้ตัวแปรที่ pointer ชี้ไปหา
- **operator +, -, ++, --**
  - คำนวนตำแหน่งใหม่ที่ pointer ควรชี้ไป
- **operator [*index*]**
  - ให้ตัวแปรที่ pointer + ***index*** ชี้ไปหา (คล้ายๆ *)

Address operator (&)

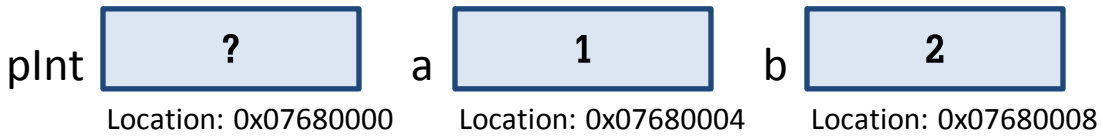
- เมื่อใช้นำหน้าตัวแปรจะให้ค่าตำแหน่งของตัวแปรนั้นออกมา
- การทำงานของคำสั่ง **pInt = &a**



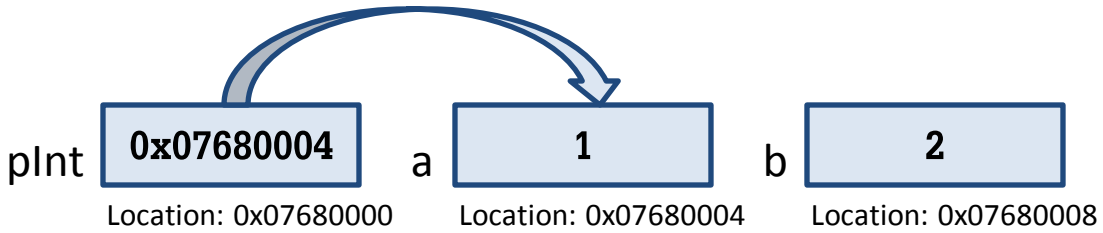
Address operator (&)

Pointer Syntax

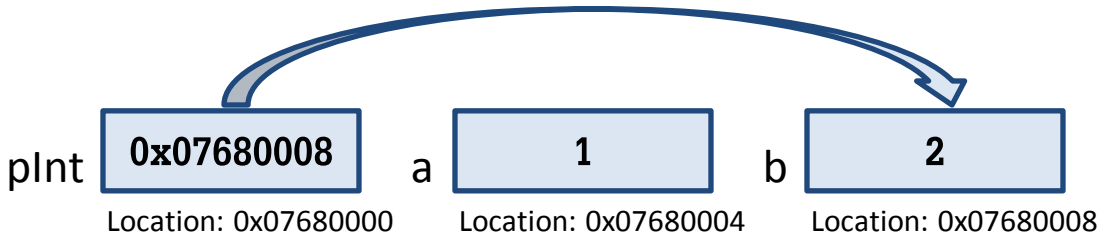
```
int * pInt;

int a = 1, b = 2;
```




```
pInt = &a;
```

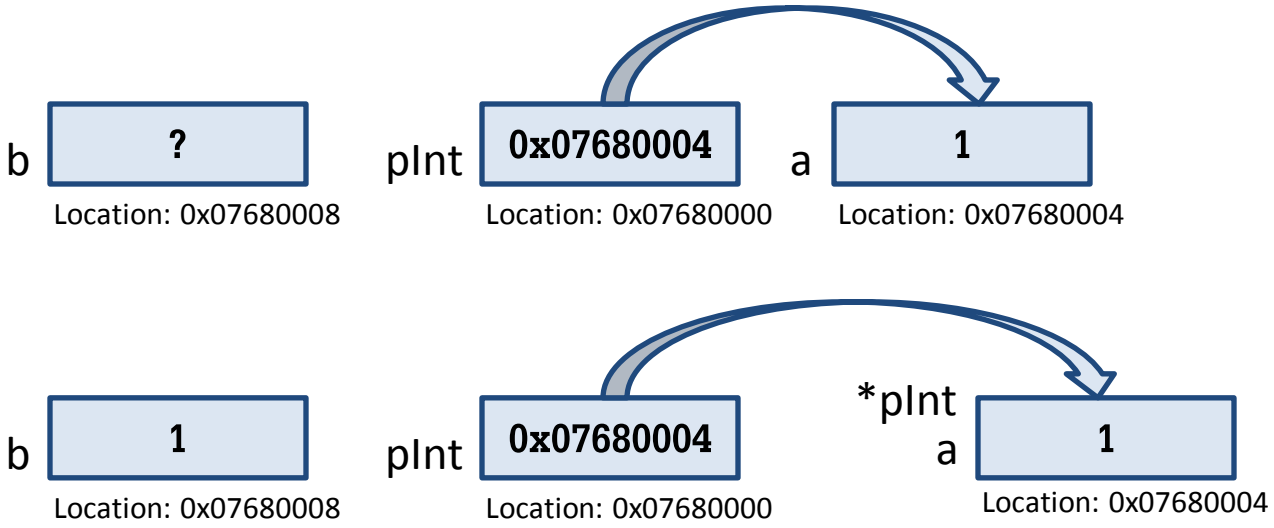



```
pInt = &b;
```



Indirection operator (*)

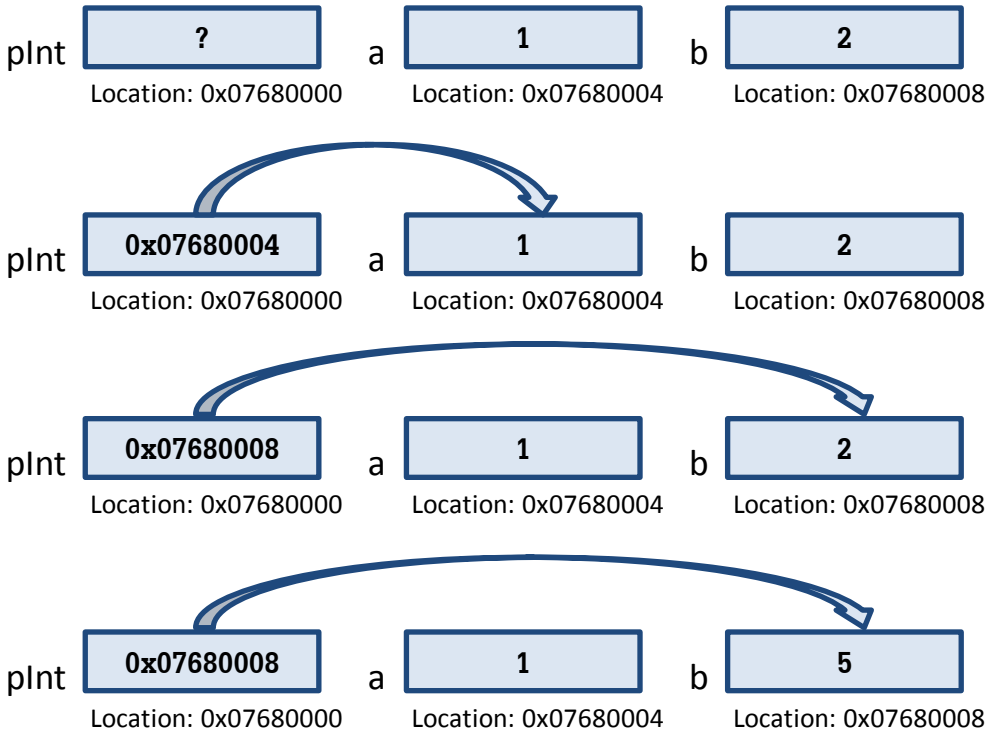
- เมื่อใช้นำหน้าตัวแปรประเภท pointer จะให้ตัวแปรที่ชี้ไป
- การทำงานของคำสั่ง b = *pInt;



Indirection operator (*)

Pointer Syntax

```
int * pInt;
int a = 1, b = 2;
pInt = &a;
printf("%d", *pInt);
pInt = &b;
printf("%d", *pInt);
b = 5;
printf("%d", *pInt);
```



output

```
1
2
5
```

การจองหน่วยความจำ

- เราสามารถจองหน่วยความจำได้ด้วยคำสั่ง malloc และ calloc
- ตรวจสอบ include directive ด้วย (stdlib.h, malloc.h)
- ฟังก์ชั่น
  - void *malloc(size_t *size*);
  - void *calloc(size_t *num*, size_t *size*);

การจองหน่วยความจำ

- ตัวอย่าง
  - `int * list1 = malloc(100 * sizeof(int));`

    `...`

    `free(list1);`
  - `int * list2 = calloc(100, sizeof(int));`

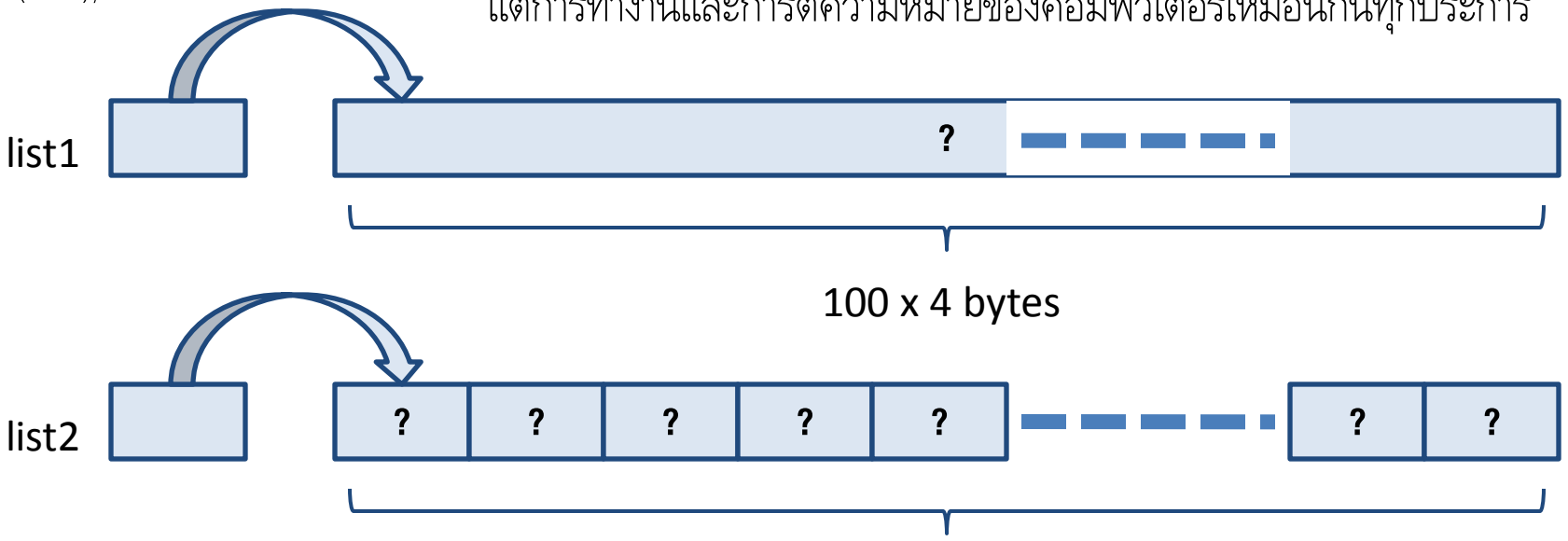
    `...`

    `free(list2);`

การจองหน่วยความจำ

```
int * list1 = malloc(100 * sizeof(int));
...
free(list1);
int * list2 = calloc(100, sizeof(int));
...
free(list2);
```

การประกาศ list1, list2 มีแนวคิดและมุมมองต่างกันเล็กน้อย
แต่การทำงานและการตีความหมายของคอมพิวเตอร์เหมือนกันทุกประการ

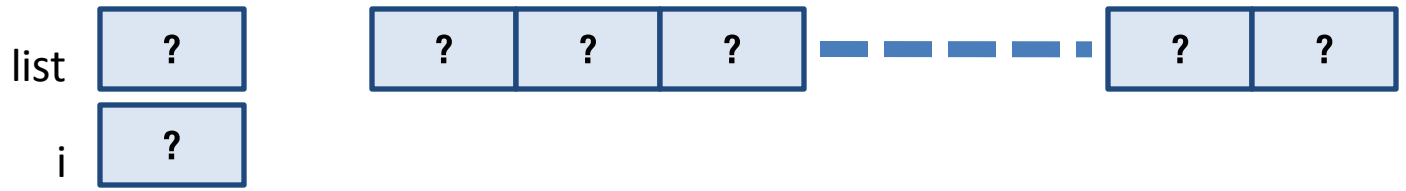


assuming sizeof(int) = 4

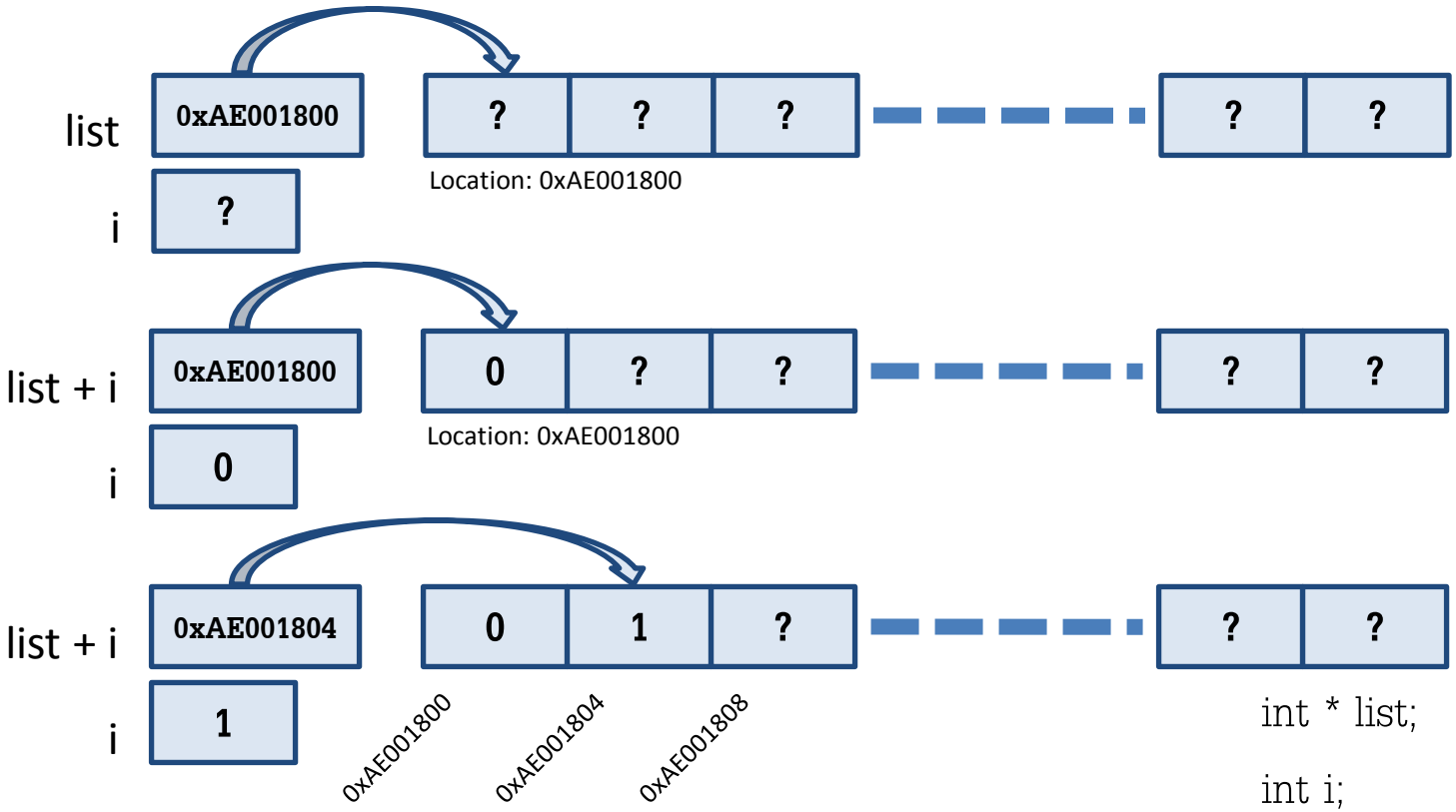
Operator ++, --, +, -

- ตัวอย่าง

```
int * list;
int i;
list = calloc(10, sizeof(int));
for (i = 0; i < 10; i++)
    *(list + i) = i;
```



Operator ++, --, +, -




```
int * list;
int i;
list = calloc(10, sizeof(int));
for (i = 0; i < 10; i++)
    *(list + i) = i;
```

Operator ++, --, +, -, [index]

```
int * list;
int i;
list = calloc(10, sizeof(int));
for (i = 0; i < 10; i++)
    *(list + i) = i;
```

```
int * list;
int i;
list = calloc(10, sizeof(int));
for (i = 0; i < 10; i++)
    list[i] = i;
```

Operator ++, --, +, -, [index]

```
int * list;
int i;
list = calloc(10, sizeof(int));
for (i = 0; i < 10; i++)
    list[i] = i;
```

```
int * list, *p;
int i;
list = calloc(10, sizeof(int));
p = list; i = 0;
while (i < 10) {
    *p = i;
    p++; i++;
}
```

การฟรีหน่วยความจำ

- เราสามารถฟรีหน่วยความจำได้ด้วยคำสั่ง **free**
- ฟังก์ชั่น
  - **void free(void* *memblock*);**
- ภาษาซีไม่มีการจัดการหน่วยความจำให้ ผู้เขียนโปรแกรมต้องรับผิดชอบเอง ถ้าจองหน่วยความจำเรื่อยๆ โดยไม่มีการลบขนาดหน่วยความจำที่เหลือจะลดน้อยลงจนคอมพิวเตอร์ไม่สามารถทำงานได้

ตัวอย่าง

An example

ex06_02.c

```
...

void main() {
   int * pVote;
   int nCandidates, i, a;

   printf("The number of candidates : ");
   scanf("%d", &nCandidates);

   pVote = malloc(nCandidates * sizeof(int));
   memset(pVote, 0, nCandidates * sizeof(int));

   while (1) {
      scanf("%d", &a);
      if (a == -1)
           break;

      pVote[a]++;
   }

   for (i = 0; i < nCandidates; i++)
      printf("%d : %d\n", i, pVote[i]);
   free(pVote);
}
```

รับค่าเข้ามาเก็บไว้ในตัวแปรอาเรย์ที่ใช้นับความถี่

ตัวอย่าง

An example

ex06_03.c

```
...

void main() {
   int * pVote;
   int nCandidates, i, a;

   printf("The number of candidates : ");
   scanf("%d", &nCandidates);

   pVote = malloc(nCandidates * sizeof(int));
   memset(pVote, 0, nCandidates * sizeof(int));

   while (1) {
      scanf("%d", &a);
      if (a == -1)
           break;

      (*(pVote + a))++;
   }

   for (i = 0; i < nCandidates; i++)
      printf("%d : %d\n", i, *(pVote + i));
}
```

pVote[a] มีความหมายเหมือนกับ

***(pVote + a)**

pVote[a]++ มีความหมายเหมือนกับ

(*(pVote + a))++

++ มี precedence สูงกว่า * เลยต้องใส่วงเล็บ

ข้อสังเกต

- ก่อนหน้านี้เราต้องรู้จำนวนสมาชิกของอาเรย์ขณะเขียนโปรแกรม เพราะเวลาประกาศอาเรย์ต้องระบุจำนวนสมาชิก
- เราสามารถใช้ความรู้เรื่อง pointer การจองและฟรีหน่วยความจำในการประกาศอาเรย์ที่ทราบขนาดระหว่างที่โปรแกรมทำงานได้ (dynamic allocation)

สรุป

- เครื่องหมาย *
  - ถ้าอยู่ระหว่างตัวเลขค่าคงที่หรือตัวแปรประเภทตัวเลข จะมีความหมายเป็น arithmetic multiplication operator
  - ถ้าอยู่นำหน้าตัวแปรประเภทพอยเตอร์ จะมีความหมายเป็น indirect operator
  - ถ้าอยู่นำหน้า identifier ในประโยคการประกาศตัวแปร จะมีความหมายเป็นการประกาศตัวแปรแบบพอยเตอร์

ตัวอย่าง

An example

ex06_04.c

```
...

void main() {
  int a, b, c, d;           // *p, *x, *y - pointer declaration
  int *p;                   // int             : a, b, c, d, z
  int *x, *y, z;            // pointer to int : p, x, y

  p = &a;                   // let p point to a

  a = 2 * 5;                // multiplication operator
  b = a * 2;                // multiplication operator
  c = a * b;                // multiplication operator

  x = &b;                   // let x point to b
  y = &c;                   // let y point to c
                            // At this point : *p = a = 10
                            //                 *x = b = 20
                            //                 *y = c = 200
  d = *p + 1;               // * - indirection operator
  z = *x * *y;              // * - multiplication operator
  *p = z**y;
}
```

สรุป

- เครื่องหมาย +, -
  - ถ้าอยู่ระหว่างตัวเลขค่าคงที่หรือตัวแปรประเภทตัวเลข จะมีความหมายเป็น arithmetic addition/subtraction operator
  - ถ้าอยู่ระหว่างตัวเลขค่าคงที่หรือตัวแปรประเภทตัวเลข กับตัวแปรประเภทพอยเตอร์ จะมีความหมายเป็น pointer arithmetic addition/subtraction operator
  - อยู่ระหว่างพอยเตอร์ด้วยกันเองไม่ได้

สรุป

- เครื่องหมาย ++, --
  - ถ้าอยู่ก่อนหรือหลังตัวแปรประเภทตัวเลข จะมีความหมายเป็น arithmetic increment/decrement prefix/postfix operator
  - ถ้าอยู่ก่อนหรือหลังตัวแปรประเภทพอยเตอร์ จะมีความหมายเป็น pointer arithmetic increment/decrement prefix/postfix operator
  - อยู่ก่อนหรือหลังค่าคงที่ไม่ได้

สรุป

- เครื่องหมาย [***index***]
  - เมื่ออยู่ตามหลังตัวแปรประเภทอาเรย์ จะหมายถึงสมาชิกของอาเรย์นั้นลำดับที่ ***index***
  - เมื่ออยู่ตามหลังตัวแปรประเภทพอยเตอร์เช่น **p**[***index***] จะมีความหมายเท่ากับ ***(p + *index**) ซึ่งสามารถมองได้แบบเดียวกับตัวแปรประเภทอาเรย์

ตัวอย่าง

An example

ex06_05.c

```
...

void main() {
  int a, b, c, d;           // *p, *x, *y - pointer declaration
  int *p;                   // int             : a, b, c, d, z
  int *x, *y, z;            // pointer to int : p, x, y

  p = &a;                   // let p point to a

  a = 2 * 5;                // multiplication operator
  b = a * 2;                // multiplication operator
  c = a * b;                // multiplication operator

  x = &b;                   // let x point to b
  y = &c;                   // let y point to c
                            // At this point : *p = a = 10
                            //                 *x = b = 20
                            //                 *y = c = 200
  d = *p + 1;               // * - indirection operator
  z = *x * *y;              // * - multiplication operator
  *p = z**y;
}
```

ตัวอย่าง

An example

ex06_06.c

```
...
void main() {
   int a, *p, *r, *q;
   int z[10];

   p = malloc(10 * sizeof(int));
   for (a = 0; a < 10; a++) {
      z[a] = a; p[a] = a;
   }

   r = p;
   printf("*r = %d\n", *r);
   r++;
   printf("*r = %d\n", *r);
   printf("*r = %d\n", *(r + 2));

   r = z; q = &z[0];
   printf("*r = %d, *q = %d\n", *r, *q);
   r++; q++;
   printf("*r = %d, *q = %d\n", *r, *q);
   *(r + 1) = 100;
   printf("r[1] = %d, *(q + 1) = %d\n", r[1], *(q + 1));
}
```

จบบทที่ 6

The End of Chapter 6

บทที่ 7 ประเภทตัวแปรขั้นสูง 2

...