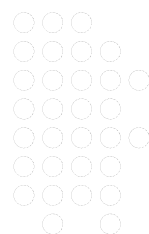


陣列與指標的關聯

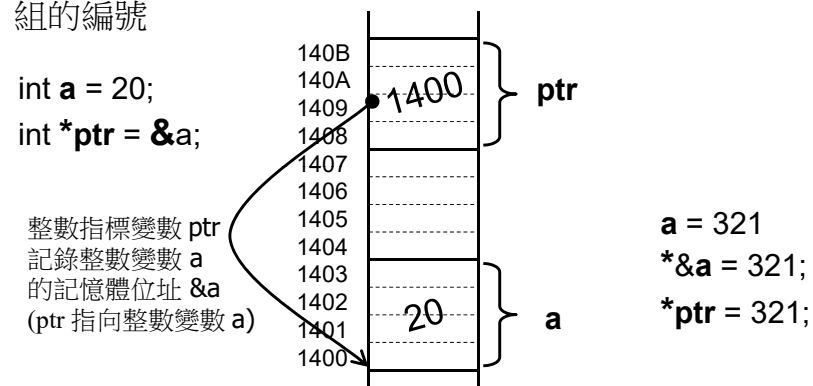
C 語言中陣列是以指標實作的
陣列與指標在定義時, 使用記憶體不一
樣大, 但是在存取資料時幾乎是等效的



1

變數的位址

- 變數的位址是它所使用的記憶體中, 第一個位元組的編號



- 拿到一個位址常數 **&a** 或是指標變數 **ptr**, 主要目標是運用間接存取運算子 ***** 來存取安排在該位址的變數 a

2

一維陣列

- int x[3]; 一次定義了
 - 存放在連續記憶體裡面的三個整數變數 x[0], x[1], 與 x[2]
 - x 是 3 個整數的陣列, 型態是 int[3], 也是一個整數指標常數 (記憶體位址常數), 型態是 int *const, 值是 &x[0]

int x[3] = {2,3,4};

int *ptr = x; // &x[0]

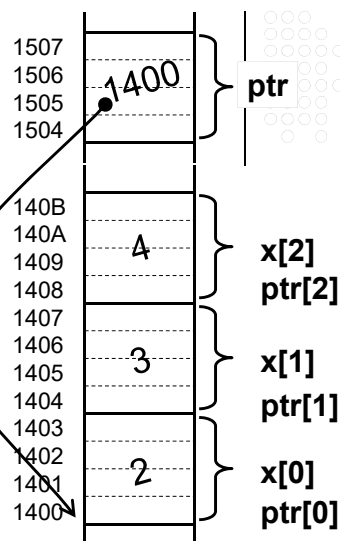
少部分程式裡, 例如: sizeof(x) 和 &x,

x 代表 int[3] 這種型態的變數, &x 的型態是 int (*)[3],

代表「3 個元素的整數陣列」的位址, &x 不是位址常數的位址(型態是 int **)

差異表現在 ptr=&x 時 ① ptr 的型態、② sizeof(*ptr)、③ ptr+k、④ ptr[k]

需要記憶體位址的地方, x 代表陣列第一個元素的位址 &x[0], 型態是 int *const



二維陣列

- int x[5][3]; 一次定義了
 - 存放在連續記憶體裡面的十五個整數變數, 依序是 x[0][0], x[0][1], x[0][2], x[1][0], ..., x[4][2]
 - 一個指向整數陣列 int[3] 變數的指標常數 x, 型態是 int (*const)[3] 或是 int[5][3]
 - 五個指向整數變數的指標常數 x[0], x[1], ..., x[4], 型態是 int *const 或是 int[3]

int x[5][3];

int (*ptr)[3] = x; // 或 &x[0]

int *ptr = x[1]; // 或 &x[1][0]

完全不會出現 int ** 的型態

4

一維指標陣列

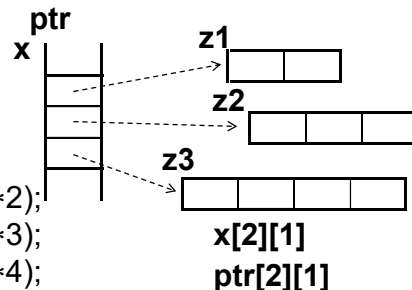
- `int *x[3]`; 一次定義了

- 存放在連續記憶體裡面的三個整數指標變數 `x[0]`, `x[1]`, 與 `x[2]`
- 還有一個整數**雙重指標**常數 `x`, 型態是 `int **const`, 值是 `&x[0]`, 或是 `int*[3]`

```
int z1[2], z2[3], z3[4];
int *x[3] = {z1, z2, z3};
int **ptr = x; // &x[0]
```

比較常看到用來組織動態配置的記憶體

```
int *z1=(int*) malloc(sizeof(int)*2);
int *z2=(int*) malloc(sizeof(int)*3);
int *z3=(int*) malloc(sizeof(int)*4);
```



5

動態配置 `m x n` 二維陣列

- 在 Heap 區域配置 `m` 列 `n` 行的二維陣列
- 範例

```
int i, j, m=5, n=3;
int **x;
```

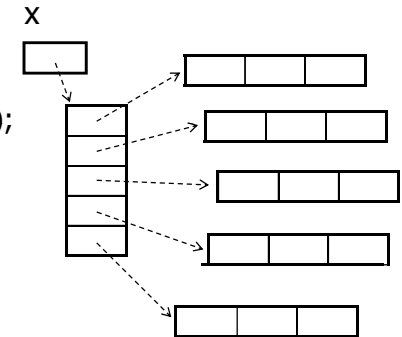
Conceptual layout

```
x = (int **) malloc(sizeof(int *)*m);
for (i=0; i<m; i++)
    x[i] = (int *) malloc(sizeof(int)*n);

for (i=0; i<m; i++)
    for (j=0; j<n; j++)
        x[i][j] = 0;

fun(x);

for (i=0; i<m; i++)
    free(x[i]);
free(x);
```



`void fun(int **const intarray) { ... }` or `void fun(int *intarray[]) { ... }`₆

C/C++ 的陣列都是一維的

- C/C++ **array** syntax is **one dimensional**

`int x[3];` 3-element array of int

- What? How about `int x[4][3]`; or `int y[5][4][3]`;
- Actually compiler treats them as:

`int x[4][3];` 4-element array of `int[3]`

`int y[5][4][3];` 5-element array of `int[4][3]`

- Another way to define array `int x[4][3];`

```
typedef int INT3[3];
INT3 x[4];
int y[5][4][3];
```

```
typedef int INT4x3[4][3];
INT4x3 y[5];
INT3 y[5][4];
```

7

sizeof()

`int x[4], y[5][4];`

<code>sizeof(int)</code>	4	<code>sizeof(*x)</code>	4	<code>int</code>
<code>sizeof(int*)</code>	8	<code>sizeof(*(int*)x)</code>	16	<code>int[4]</code>
<code>sizeof(int[4])</code>	16	<code>sizeof(*&y[0])</code>	16	<code>int[4]</code>
<code>sizeof(int*[4])</code>	32	<code>sizeof(y)</code>	80	<code>int[5][4]</code>
<code>sizeof(x)</code>	16	<code>sizeof(*y[0])</code>	4	<code>int</code>
<code>sizeof((int*)x)</code>	8	<code>sizeof(*(int*)y[0])</code>	16	<code>int[4]</code>

8

指標的型態

```
int x[4], y[5][4], a, z[6][5][4];
int *ptrI, (*ptrI4)[4], (*ptrI5x4)[5][4];

ptrI4 = &x;    // type matched  int (*)[4] => int (*)[4]
ptrI4 = y;     // type matched  int (*)[4] => int (*)[4]
ptrI = x;      // automatic type conversion int[4] => int *
ptrI = y[0];   // automatic type conversion int[4] => int *
//ptrI4 = &a;  // error int * => int (*)[4]
//ptrI = &x;   // error int (*)[4] => int *
ptrI5x4 = &y;  // type matched int (*)[5][4] => int (*)[5][4]
ptrI5x4 = z;   // type matched int (*)[5][4] => int (*)[5][4]
```

9

函式參數的陣列宣告

```
int x[7]={1,2,3,4,5,6,7};
int result = sum(x);
printf("%d\n", result);
printf("%d\n", sizeof(x));
```

概念上最簡單的寫法

28

仔細比較一下

```
int sum(int x[7]) {
    int i, y=0;
    for (i=0; i<7; i++)
        y += x[i];
    printf("%d\n", sizeof(x));
    return y;
}
```

```
int sum(int x[7]) {
    int i, y=0, z[3];
    for (i=0; i<7; i++)
        etu printf("%d %d\n", sizeof(x), sizeof(z));
    func(&x, &z);
    return y;
}
```

噢! 這個陣列語法是假的

12

```
int sum(int *x) {
    int i, y=0;
    for (i=0; i<7; i++)
        y += x[i];
    printf("%d\n", sizeof(x));
    return y;
}
```

```
void func(int **x, int(*z)[3])
{
    ...
}
```

其實是指標耶!?

10

範例

```
void func(int x[4], int *const xptr,
          int y[5][4], int (*const yptr)[4]) {
    printf("%d\n", sizeof(xptr));    8
    printf("%d\n", sizeof(*xptr));   4
    printf("%d\n", sizeof(x));       8
    printf("%d\n", sizeof(*x));       4

    printf("%d\n", sizeof(yptr));     8
    printf("%d\n", sizeof(*yptr));    16
    printf("%d\n", sizeof(yptr[0]));  16

    printf("%d\n", sizeof(y));        8
    printf("%d\n", sizeof(*y));       16
    printf("%d\n", sizeof(y[0]));     16
}
```

```
int main() {
    int x[4], y[5][4];
    func(x, x, y, y);
}
```

int*, not int[4]
int

int[4]
int[4]

int(*)[4], not int[5][4]
int[4]
int[4]

11