陣列 (Array)

陣列 (Array)

- 陣列就是一連串相同型別的元素，放置在連續的位置
- 陣列的宣告必須告訴 compiler 這個陣列的組成元素的型別以及總共包含多少元素
- 如此一來 compiler 才能知道如何處理我們需要的陣列，例如下面的宣告：

陣列 (Array)

```
int main(void)
{
    float signal[100];
    char name[20];
    int histogram[128];
    ...
    return 0;
}
```

- 方括號 **[ ]** 用來表示 **signal** 是個陣列，括號裡的數字代表陣列的大小，而每個元素的型別都是 **float**
- 陣列的每個元素可以被單獨存取，編號從 **0** 開始，所以在上面的例子裡 **signal[0]** 是第一個元素，而 **signal[99]** 是最後一個元素

範例E10_01.c

```c
#include <stdio.h>
#define NBIN 10
#define STARS "******************************"
int main(void)
{
   int hist[NBIN] = {3,5,2,1,8,3,1,5,4,3};
   int index;
   for (index = 0; index < NBIN; index++) {
      printf("[%d] %2d ", index, hist[index]);
      printf("%.*s\n", hist[index], STARS);
   }
   return 0;
}
```

範例E10_02.c

```c
#include <stdio.h>
#define STARS "*******************************"
int main(void)
{
    int hist[] = {3,5,2,1,8,3,1,5,4,3};
    int index;
    for (index = 0; index < (sizeof hist) / (sizeof
hist[0]); index++) {
        printf("[%d] %2d ", index, hist[index]);
        printf("%.*s\n", hist[index], STARS);
    }
    return 0;
}
```

陣列 (Array)

- 除了用設定初始值的方式之外，真正比較常用的作法應該是配合迴圈，逐一把適當的值填入陣列中。
- 範例 E10_03.c

陣列 (Array)

- C 語言本身並沒有提供把一個陣列整個設給另一個陣列的功能，所以不能用像下面這樣的寫法來複製陣列的內容：

```
int hist1[NBIN];
int hist2[NBIN];
hist1 = hist2;   /* wrong */
```

- 只能透過迴圈，對每個元素以逐一存取的方式，把一個陣列的內容複製到另一個陣列，或是用 C Standard Library 提供的 function 來複製陣列
- 範例 E10_04.c

設定陣列的大小

- 合法的宣告

```
float a1[5];
float a2[5*2 + 1];
float a3[(int)2.3];
```

- 錯誤的宣告

```
float a4[-4];
float a5[0];
float a6[2.3];
```

- C99 支援的宣告形式

```
int n=5, m=6;
float a7[n];   /* C99 */
float a8[m*n]; /* C99 */
```

二維陣列

- 數位影像或矩陣，本身就具有二維特性，如果要對影像或矩陣做處理，使用二維陣列的方式來操作在思考上會比較直接
- 在 C 程式裡要產生一個二維陣列的寫法是

```
int a[3][4];
```

a[0][0]	a[0][1]	a[0][2]	a[0][3]
a[1][0]	a[1][1]	a[1][2]	a[1][3]
a[2][0]	a[2][1]	a[2][2]	a[2][3]

- 初值設定

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

或

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

二維陣列

- 接著來看看不同的初始化造成的結果
- Demo