

由片語學習C程式設計

台灣大學資訊工程系劉邦鋒著

台灣大學劉邦鋒老師講授

August 19, 2016

第六單元

陣列

陣列

- 如果我們想要使用大量具同性質的變數，重複宣告變數需要使用不同的變數名稱，非常麻煩。
- 我們可以使用陣列一次宣告許多同一性質的變數。

片語 1: 使用陣列一次宣告10個整數變數

```
1 int a[10];
```

- 一次宣告 10 個整數變數。
- a 後面的方括號 [10] 即表示 a 是一個有 10 個元素的整數陣列。

特殊字元

中括號 [] 用來代表陣列。

屬性

- 一個陣列和一個變數一樣，有類別、名字、值、位址等屬性，但陣列還多了一個屬性，就是陣列中有幾個元素。
- 因為陣列中有數個元素，我們必須用一個數字代表我們使用的是哪一個。這個數字就稱為註標 (index)。
- 可以把陣列想像成數學中的向量，而註標就是指定要從哪一個維度取出向量中的元素。
- 與一般的數學向量慣例不同，C 程式語言 陣列的註標是由 0 開始。`a[0]` 是陣列 `a` 的第一個元素，`a[9]` 是最後一個元素。

```
int a[10];
```

a[0]	
a[1]	
a[2]	
a[3]	
a[4]	
a[5]	
a[6]	
a[7]	
a[8]	
a[9]	

```
int a[10];
```

- 類別 – 整數 `int`
- 名字 – `a`
- 值 – 個別元素有個別的值。
- 元素個數 – 10
- 位址 – 第一個元素 `a[0]` 的位址。

學習要點

陣列的註標由 0 開始，所以陣列的第一個元素是 [0]。

一維陣列

片語 2: 使用一個 for 迴圈處理一個陣列 a

```
1  int i;  
2  int a[10];  
3  ...  
4  for (i = 0; i < 10; i++)  
5      processing a[i];
```

- 使用一個 for 迴圈將陣列中的每一個元素做處理。
- 必須宣告另一個變數 i 作為註標，方便從陣列 a 取元素。i 的範圍為 0 到 9。

範例程式 3: (print-array.c) 印出陣列中元素的值

```
1  #include <stdio.h>
2
3  main()
4  {
5      int a[10];
6      int i;
7      for (i = 0; i < 10; i++)
8          scanf("%d", &(a[i]));
9      for (i = 0; i < 10; i++)
10         printf("%d\n", a[i]);
11 }
```

- 使用片語 1 將一個陣列初始化為註標對應的偶數，並印出陣列 `a` 中元素的值。
- 我們使用一個 `for` 迴圈將 `a` 中元素初始化，再用另一個 `for` 迴圈印出陣列 `a` 中各元素的值。
- 使用一個變數 `i` 作為註標，方便從陣列 `a` 取元素。

輸入

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

輸出

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

範例程式 4: (inner-product.c) 計算內積

```
1  #include <stdio.h>
2  main()
3  {
4      int A[5], B[5], C = 0;
5      int i, j;
6      for (i = 0; i < 5; i++)
7          scanf("%d", &(A[i]));
8      for (i = 0; i < 5; i++)
9          scanf("%d", &(B[i]));
10     for (i = 0; i < 5; i++)
11         C += A[i] * B[i];
12     printf("%d\n", C);
13 }
```

計算內積

- 計算向量 **A** 及 **B** 的內積，並將結果存入變數 **C** 中。
- 首先自鍵盤讀入向量 **A** 及 向量 **B** 中各元素的值。
- 再用另一個 **for** 迴圈來完成這個內積。
- 最後我們將結果 **C** 的值印出。

輸入

1	1	2	3	4	5
2	5	4	3	2	1

輸出

1	35
---	----

- 輸入檔並非一個數字一行，而是一個長度為 5 的向量一行，而數字之間用一個空白隔開。
- `scanf` 會在輸入檔中持續找數字，一行沒有找下一行，直到找到為止。所以對 `scanf` 完全沒有影響。
- 在準備輸入檔時，我們可以清楚的理解，第一行五個數字是給 A 向量，第二行五個數字是給 B 向量，這樣就不容易弄錯。

學習要點

輸入檔的格式有助於了解輸入資料與變數的歸屬關係。

費伯納西數列

$$fib(i) = \begin{cases} 0 & i = 0 \\ 1 & i = 1 \\ fib(i-1) + fib(i-2) & i \geq 2 \end{cases} \quad (1)$$

範例程式 5: (fib-array.c) 計算費伯納西數列到第 n 項。

```
1  #include <stdio.h>
2  main()
3  {
4      int i;
5      int fab[100];
6      int n;
7      scanf("%d", &n);
8      fab[0] = 0;
9      fab[1] = 1;
10     for (i = 2; i < n; i++)
11         fab[i] = fab[i - 1] + fab[i - 2];
12     for (i = 0; i < n; i++)
13         printf("%d\n", fab[i]);
14 }
```

費伯納西數列

- 我們使用一個陣列 `fib` 存費伯納西數列。
- `fib[i]` 存費伯納西數列的第 i 項 $fib(i)$ 。
- 然後計算到費伯納西數列的第 n 項。

輸入

1 10

輸出

1	0
2	1
3	1
4	2
5	3
6	5
7	8
8	13
9	21
10	34

片語 6: 在 array 中尋找第一個不為 1 的元素

```
1  int array[10];  
2  ...  
3  i = 0;  
4  while (i < 10 && array[i] == 1)  
5      i++;
```

- 在陣列中尋找第一個符合某種性質的元素。
- 必須限定 i 不能超過 9，因為陣列只有 10 個元素。
- 迴圈結束後如果 i 為 10，則陣列 `array` 所有元素均為 1，否則 `array[i]` 即為第一個不為 1 的元素。

範例程式 7: (prime-array.c) 印出 n 之內的質數

```
1  #include <stdio.h>
2  main()
3  {
4      int composite[101];
5      int i, n, j = 2;
6      scanf("%d", &n);
7      for (i = 2; i <= n; i++)
8          composite[i] = 0;
9      while (j * j <= n) {
10         while (composite[j] == 1)
11             j++;
12         for (i = 2 * j; i <= n; i += j)
13             composite[i] = 1;
14         j++;
15     }
16     for (i = 2; i <= n; i++)
17         if (composite[i] == 0)
18             printf("%d\n", i);
19 }
```


- 利用一個陣列 `composite` 作是否為合成數的旗標 (flag)。
如果 `j` 是一個合成數，則對應的 `composite[j]` 為 1，否則 `j` 是一個質數，而對應的 `composite[j]` 為 0。
- 假設所有由 2 到 n 的整數皆為質數，
- 由 2 開始，找第一個還未被確認為合成數的整數 `j`，並認定為質數。
- 設定 `j` 的倍數皆為合成數。
- 將 `j` 加 1，測試下一個數。
- 只需測試到 $j * j \leq n$ 即可。

輸入

1

50

輸出

1 2
2 3
3 5
4 7
5 11
6 13
7 17
8 19
9 23
10 29
11 31
12 37
13 41
14 43
15 47

學習要點

我們可以利用旗標陣列紀錄某個整數是否具有某種性質。

泡沫排序法

- 由左到右比較兩個相鄰元素，如果註標比較小的元素比較大，則交換元素值。
- 由註標比較小的元素兩兩交換到註標比較大的元素，就能使大的元素向註標比較大的方向移動，而小的元素向註標比較小的方向移動。
- 用兩層 `for` 迴圈實作。
 - 第一層迴圈決定兩兩交換的範圍。
 - 第二層則實際作兩兩交換，

範例程式 8: (bubble-sort.c) 泡沫排序法

```
1  #include <stdio.h>
2  int main()
3  {
4      int m, n[100];
5      int i, j, temp;
6      scanf("%d", &m);
7      for (i = 0; i < m; i++)
8          scanf("%d", &(n[i]));
9      for (i = m - 2; i >= 0; i--)
10         for (j = 0; j <= i; j++)
11             if (n[j] > n[j + 1]) {
12                 temp = n[j];
13                 n[j] = n[j + 1];
14                 n[j + 1] = temp;
15             }
16         for (i = 0; i < m; i++)
17             printf("%d\n", n[i]);
18     return 0;
19 }
```

輸入

1	10
2	7 6 9 0 8 4 5 3 2 1

輸出

1	0
2	1
3	2
4	3
5	4
6	5
7	6
8	7
9	8
10	9

片語 9: 以十六進位印出變數所在的記憶體位址

```
1 printf ("%p\n", &i);
```

- 使用 `&i` 印出變數 `i` 的 **記憶體位址**。
- 格式 `"%p\n"` 是以一般記憶體位址常用的**十六進位** 印出並換行。

範例程式 10: (print-array-address.c) 印出陣列 a 中的元素的大小及位址

```
1  #include <stdio.h>
2  main()
3  {
4      int a[10];
5      int i;
6
7      printf("%d\n", sizeof(a[0]));
8      printf("%d\n", sizeof(a));
9      for (i = 0; i < 10; i++)
10         printf("%p\n", &(a[i]));
11     printf("%p\n", &a);
12     printf("%p\n", a);
13 }
```


輸出

```
1 4
2 40
3 0x7fff8afdf920
4 0x7fff8afdf924
5 0x7fff8afdf928
6 0x7fff8afdf92c
7 0x7fff8afdf930
8 0x7fff8afdf934
9 0x7fff8afdf938
10 0x7fff8afdf93c
11 0x7fff8afdf940
12 0x7fff8afdf944
13 0x7fff8afdf920
14 0x7fff8afdf920
```

- `a[0]` 是一個 32 位元的整數，所以會印出 4，
- `a` 是 10 個 32 位元的整數所組成的陣列，所以會印出 40。
- 陣列 `a` 元素的記憶體位址是連續的，而且一個元素和下一個元素的位址剛好差 4。
- 陣列元素在記憶體中是由小排到大，而且每個元素佔 4 個位元組。

$$a + (i \times L) \quad (2)$$

- 元素 $a[i]$ 的記憶體位址可用上式表示。
- a 為陣列 a 的起始位址，而 L 為每一元素所佔的位元組數。
- a 的位址 和 $a[0]$ 的位址一樣。

0028FEF4	a[0]
0028FEF8	a[1]
0028FEFC	a[2]
0028FF00	a[3]
0028FF04	a[4]
0028FF08	a[5]
0028FF0C	a[6]
0028FF10	a[7]
0028FF14	a[8]
0028FF18	a[9]

address of $a[i]$ is $0028FEF4 + (i \times 4)$ (3)

a 的意義

- 觀察執行結果可發現 `a` 的位址 和 `a[0]` 的位址是一樣的。
- 在C 程式語言中，`a` 的值並非代表陣列中所有的元素的值，而就是代表陣列 `a` 的 位址，這是C 程式語言中關於陣列最需要注意的地方。

學習要點

在 C 程式語言中，陣列的值就是陣列的位址。如果要取陣列中元素的值必須用方括號 `[]` 再加上註標變數。

片語 11: 陣列的初始化

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

- 陣列可以使用類似一般變數的方法加以初始化。
- 想給的初始值必須以逗號分開，再用大括號括起來。

片語 12: 陣列的初始化

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

- 如果陣列有初始化，但沒有宣告宣告長度，編譯器會自行決定陣列長度。

片語 13: 陣列的初始化

```
1 int array[5] = {1, 2, 3};  
2 int array[5] = {1, 2, 3, 0, 0};
```

- 如果陣列有宣告長度，也有初始化，但初始化所給的元素數目不足，則其餘的元素會被初始為 0。
- 兩種寫法是一樣的。

片語 14: 陣列的初始化

```
1 int array[10000] = {0};
```

- 利用這個補 0 的特性，我們可以很容易將整個陣列初始為 0。

多維陣列

片語 15: 宣告一個 3 乘 4 的整數陣列。

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

- 多維陣列至少兩個個註標變數加上方括號 [] 才能指定陣列中的元素。

片語 16: 使用兩層 for 迴圈處理一個二維陣列 a

```
1  int i;  
2  int j;  
3  int a[3][4];  
4  ...  
5  for (i = 0; i < 3; i++)  
6      for (j = 0; j < 4; j++)  
7          processing a[i][j];
```

- 使用兩層 for 迴圈處理一個二維陣列 a。
- 陣列有二維，所以我們必須宣告兩個變數 i, j 作為註標變數。
- 註標 i 的範圍為 0 到 2，註標 j 的範圍為 0 到 3。

矩陣相乘

- 計算矩陣 A 及 B 的乘積，將結果存入矩陣 C 中。
- 自鍵盤讀入矩陣 A 及 矩陣 B。
- 將矩陣 C 的各元素初始化為 0。
- 矩陣 C 的第 i 列 第 j 行的元素 $C[i][j]$ 是矩陣 A 的第 i 列 及 於矩陣 B 的第 j 行的內積，所以用另一個 for 迴圈及註標變數 k 來完成內積。
- 印出 C。

範例程式 17: (matrix-multiply.c) 矩陣相乘

```
4  int A[2][3], B[3][4], C[2][4];
5  int i, j, k;
6
7  for (i = 0; i < 2; i++)
8      for (j = 0; j < 3; j++)
9          scanf("%d", &(A[i][j]));
10 for (i = 0; i < 3; i++)
11     for (j = 0; j < 4; j++)
12         scanf("%d", &(B[i][j]));
13 for (i = 0; i < 2; i++)
14     for (j = 0; j < 4; j++)
15         C[i][j] = 0;
```

```
17  for (i = 0; i < 2; i++)
18      for (j = 0; j < 4; j++)
19          for (k = 0; k < 3; k++)
20              C[i][j] += A[i][k] * B[k][j];
21
22  for (i = 0; i < 2; i++)
23      for (j = 0; j < 4; j++)
24          printf("%d\n", C[i][j]);
```

輸入

1	4	6	2	
2	7	8	3	
3	5	8	2	5
4	6	7	4	2
5	7	6	5	4

輸出

1	70
2	86
3	42
4	40
5	104
6	130
7	61
8	63

片語 18: 換行

```
1 printf("\n");
```

- 如果輸出一行一個數字就不容易理解，因為沒辦法和矩陣的形狀連結起來。
- 如果我們在輸出矩陣 **c** 的時候不換行，執行結果會是所有的數字都連在一起。
- 為了避免這個問題，我們可以在格式字串中加入空白，例如 `"%d "` 使數字不會連在一起，請注意 `%d` 之後的空白。
- 然後在適當的地方換行，將輸出結果與矩陣的形狀結合。換行的方法就是在 `printf` 的格式字串單獨使用 `\n`。

範例程式 19: (matrix-multiply-lines.c) 矩陣相乘

```
19  for (i = 0; i < 2; i++)
20      for (j = 0; j < 4; j++)
21          for (k = 0; k < 3; k++)
22              C[i][j] += A[i][k] * B[k][j];
23
24  for (i = 0; i < 2; i++) {
25      for (j = 0; j < 4; j++)
26          printf("%4d ", C[i][j]);
27      printf("\n");
28  }
```

輸入

1	4	6	2	
2	7	8	3	
3	5	8	2	5
4	6	7	4	2
5	7	6	5	4

輸出

1	70	86	42	40
2	104	130	61	63

輸出一行

- 如果輸出都是一行一個數字，在處理大量迴圈資料時有些不便。
- 以下範例會將輸出放進一行。

範例程式 20: (for-print-one-line.c) 在 for 迴圈中列印 i 的值

```
1  #include <stdio.h>
2  main()
3  {
4      int i;
5      int start;
6      int end;
7      scanf("%d", &start);
8      scanf("%d", &end);
9      for (i = start; i <= end; i++)
10         printf("%d ", i);
11 }
```

輸入

```
1 10
2 20
```

輸出

```
1 10 11 12 13 14 15 16 17 18 19 20
```

- 將輸出放進一行。 請注意 %d 之後的空白。

範例程式 21: (prime-array-line.c) 印出 n 之內的質數

```
4  int composite[1000];
5  int i;
6  int j = 2;
7  int count = 0;
8  int column;
9  int n;
10 scanf("%d", &n);
11 scanf("%d", &column);
12 for (i = 2; i <= n; i++)
13     composite[i] = 0;
14 while (j * j <= n) {
15     while (composite[j] == 1)
16         j++;
17     for (i = 2 * j; i <= n; i += j)
18         composite[i] = 1;
19     j++;
20 }
```

```
22  for (i = 2; i <= n; i++)
23      if (composite[i] == 0) {
24          if (count % column == (column - 1))
25              printf("%3d\n", i);
26          else
27              printf("%3d ", i);
28          count++;
29      }
```


輸入

1	100
2	8

輸出

1	2	3	5	7	11	13	17	19
2	23	29	31	37	41	43	47	53
3	59	61	67	71	73	79	83	89
4	97							

- 以一行 8 個質數的方式輸出。
- 以一個計數器 `count` 紀錄目前的質數個數。如果 `count` 除以 8 餘 7，則代表處於行末，需換行。

多維陣列記憶體擺放

- 說明多維陣列在記憶體中的擺放方法。
- 使用輸出技術，讓位址的輸出和陣列的形狀相結合，方便使用者檢視。
- 使用片語 18 換行，將輸出分塊，同樣方便使用者檢視。

範例程式 22: (print-matrix-address.c) 三維陣列大小及位址

```
4  int a[2][3][4];
5  int i, j, k;
6  printf("%d\n", sizeof(a[0][0][0]));
7  printf("%d\n", sizeof(a[0][0]));
8  printf("%d\n", sizeof(a[0]));
9  printf("%d\n", sizeof(a));
10 for (i = 0; i < 2; i++) {
11     for (j = 0; j < 3; j++) {
12         for (k = 0; k < 4; k++)
13             printf("%p ", &(a[i][j][k]));
14         printf("\n");
15     }
16     printf("\n");
17 }
```

```
19  for (i = 0; i < 2; i++)
20      printf("%p\n", &(a[i][1]));
21  printf("\n");
22  for (i = 0; i < 2; i++)
23      printf("%p\n", a[i][1]);
24  printf("\n");
25  for (i = 0; i < 2; i++)
26      printf("%p\n", &(a[i]));
27  printf("\n");
28  for (i = 0; i < 2; i++)
29      printf("%p\n", a[i]);
30  printf("\n");
31  printf("%p\n", &a);
32  printf("%p\n", a);
```

記憶體大小

- `a[0][0][0]` 是一個 4 個位元組的整數。
- `a[0][0]` 是一個 4 個 4 個位元組的整數所組成的陣列。
- `a` 包含兩個矩陣.每個矩陣有三列，每一列都是一個有四個元素的一維陣列。

a

0028FEB4	a[0][0][0]	a[0][0][1]	a[0][0][2]	a[0][0][3]
0028FEC4	a[0][1][0]	a[0][1][1]	a[0][1][2]	a[0][1][3]
0028FED4	a[0][2][0]	a[0][2][1]	a[0][2][2]	a[0][2][3]
0028FEE4	a[1][0][0]	a[1][0][1]	a[1][0][2]	a[1][0][3]
0028FEF4	a[1][1][0]	a[1][1][1]	a[1][1][2]	a[1][1][3]
0028FF04	a[1][2][0]	a[1][2][1]	a[1][2][2]	a[1][2][3]

a[0]

0028FEB4	a[0][0][0]	a[0][0][1]	a[0][0][2]	a[0][0][3]
0028FEC4	a[0][1][0]	a[0][1][1]	a[0][1][2]	a[0][1][3]
0028FED4	a[0][2][0]	a[0][2][1]	a[0][2][2]	a[0][2][3]

a[0][0]

0028FEB4	a[0][0][0]	a[0][0][1]	a[0][0][2]	a[0][0][3]
----------	------------	------------	------------	------------

記憶體位置

- 記憶體中擺放的順序是
`a[0][0][0]`，`a[0][0][1]`，`a[0][0][2]`，`a[0][0][3]`，
`a[0][1][0]`，最後一直排到 `a[1][2][3]`。
- `a` 包含兩個矩陣。每一個矩陣有三列，每一列都是一個有四個元素的一維陣列。
- 擺放的方式就是先放第一個矩陣，再放第一個矩陣。而一個矩陣中先放第一列，再放第二列，最後放第三列。每一列由小到大按註標放。

0028FEB4

a[0][0][0]

a[0][0][1]

a[0][0][2]

a[0][0][3]

0028FEC4

a[0][1][0]

a[0][1][1]

a[0][1][2]

a[0][1][3]

0028FED4

a[0][2][0]

a[0][2][1]

a[0][2][2]

a[0][2][3]

0028FEE4

a[1][0][0]

a[1][0][1]

a[1][0][2]

a[1][0][3]

0028FEF4

a[1][1][0]

a[1][1][1]

a[1][1][2]

a[1][1][3]

0028FF04

a[1][2][0]

a[1][2][1]

a[1][2][2]

a[1][2][3]

$$0028FEB4 + (i \times 3 \times 4 + j \times 4 + k) \times 4 \quad (4)$$

- 如果把 `a` 看成一個一維陣列，`a[i][j][k]` 是 `a` 的第幾個元素可由 $(i \times 3 \times 4) + (j \times 4) + k$ 算出。
- 一個 `int` 是 4 位元組，所以記憶體位置如上式。

位址計算

$$a + \left(\sum_{i=1}^{n-1} (k_i \times \prod_{j=i+1}^n m_j) + k_n \right) \times L \quad (5)$$

- 陣列 a 有 n 個維度，而第 i 個維度的元素數為 m_i 。 a 為陣列 a 的起始位址，而 L 是每一陣列元素所佔的位元組數。
- 元素 $a[k_1][k_2] \dots [k_n]$ 的記憶體位址可表為上式。
- C 程式語言 陣列的註標由 0 開始可大幅簡化位址的計算。

- 陣列 $a[i]$ 有三列，分別為 $a[i][0]$ ， $a[i][1]$ ，及 $a[i][2]$ 。
- 每一列的位址就是一維陣列第一個元素的位址。所以 $a[i][0]$ 的位址就是 $a[i][0][0]$ 的位址。
- 一維陣列的值也是一維陣列第一個元素的位址。所以 $a[i][0]$ 的值就是 $a[i][0][0]$ 的位址。
- 同理可推二維陣列 $a[i]$ 的值或是位址都是對應二維陣列的起始位置，也就是 $a[i][0][0]$ 的位址。
- $a[i]$ 的值或是位址都是整個二維陣列的起始位置，也就是 $a[i][0][0]$ 的位址。

0028FEB4

a[0][0][0]

a[0][0][1]

a[0][0][2]

a[0][0][3]

0028FEC4

a[0][1][0]

a[0][1][1]

a[0][1][2]

a[0][1][3]

0028FED4

a[0][2][0]

a[0][2][1]

a[0][2][2]

a[0][2][3]

0028FEE4

a[1][0][0]

a[1][0][1]

a[1][0][2]

a[1][0][3]

0028FEF4

a[1][1][0]

a[1][1][1]

a[1][1][2]

a[1][1][3]

0028FF04

a[1][2][0]

a[1][2][1]

a[1][2][2]

a[1][2][3]

片語 23: 二維陣列的初始化

```
1 int array[2][3] = {{1, 2, 3}, {4, 5, 6}};
```

- 多維陣列也可以使用類似一維陣列的方法加以初始化。
- 對每一列給初始值。以逗號分開，再用大括號括起來。然後再把每一列的初始值以逗號分開，再用大括號括起來。

片語 24: 二維陣列的初始化

```
1 int array[][3] = {{1, 2, 3}, {4, 5, 6}};
```

- 如果二維陣列有初始化，但沒有宣告宣告長度，編譯器會自行決定。

範例程式 25: (2d-no-length.c) 二維陣列的初始化

```
1  #include <stdio.h>
2  main()
3  {
4      int i, j;
5      int array[][3] = {{1, 2, 3}, {4, 5, 6}};
6      for (i = 0; i < 2; i++) {
7          for (j = 0; j < 3; j++)
8              printf("%3d ", array[i][j]);
9          printf("\n");
10     }
11 }
```

- 不能寫成 `array[] []`，這樣編譯器無法決定位址。
- 但可以寫成 `array[] [3]`，因為這對決定位址並無影響。

輸出

1	1	2	3
2	4	5	6

片語 26: 陣列的初始化

```
1 int array[3][3] = {{1, 2}, {4}};  
2 int array[3][3] = {{1, 2, 0},{4, 0, 0},{0, 0, 0}};
```

- 不寫第一維的維度，編譯器會自行決定。
- 如果陣列有宣告長度，也有初始化，但初始化所給的元素數目不足，則其餘的元素會被初始為 0。
- 以上兩種寫法是一樣的。

片語 27: 陣列的初始化

```
1 int array[100][100] = {{0}};
```

- 利用補 0 的特性可以很容易將整個陣列初始為 0。

範例程式 28: (2d-array-fill.c) 陣列補 0 的初始化

```
1  #include <stdio.h>
2  main()
3  {
4      int i, j;
5      int l, m;
6      int array1[3][3] = {{1, 2}, {4}};
7      int array2[3][3] = {{0}};
8      for (i = 0; i < 3; i++) {
9          for (j = 0; j < 3; j++)
10             printf("%3d ", array1[i][j]);
11         printf("\n");
12     }
13     printf("\n");
14     for (i = 0; i < 3; i++) {
15         for (j = 0; j < 3; j++)
16             printf("%3d ", array2[i][j]);
17         printf("\n");
18     }
19 }
```

輸出

1	1	2	0
2	4	0	0
3	0	0	0
4			
5	0	0	0
6	0	0	0
7	0	0	0