

# 程式設計

## Ch06. Arrays

Chuan-Chi Lai 賴傳淇

Department of Communications Engineering  
National Chung Cheng University

Spring Semester, 2024

- 1 陣列簡介 (Introduction of Arrays)
- 2 陣列的宣告 (Declaration of Arrays)
- 3 陣列的存取 (Access of Arrays)
- 4 傳遞陣列給函式 (Passing Array to Functions)
- 5 多維陣列 (Multidimensional Arrays)
- 6 附錄：陣列相關基礎演算法
  - 氣泡排序法 (Bubble Sort)
  - 二分搜尋法 (Binary Search)

## 陣列簡介 Introduction of Arrays

# 陣列簡介 (Introduction of Arrays)

- 在程式設計中，當需要處理大量資料時，可以使用陣列。陣列是一種將數個相同型態的變數按照有序的形式組織起來的一種資料結構。
- 陣列記憶體空間中，每個元素的位址是**連續的**，因此可以實現快速、隨機的存取。反之，於之後會介紹的鏈結串列 (linked list) 則是一種元素位址不連續、只能循序存取的資料結構。
- 註：**隨機存取 (random access)** 或稱直接存取，代表可以在同樣時間存取一組序列中的一個隨意元素。反之，則是循序存取，需要花更多時間進行元素的存取。

## 陣列的宣告

## Declaration of Arrays

## 宣告陣列語法

```
1 TYPE arrayName[CONST_EXPRESSION];
```

- 其中，TYPE 可以是除了 void 以外的任何型態，而 CONST\_EXPRESSION 是選擇性的：
  - ① 由陣列宣告指定元素數目時，CONST\_EXPRESSION 必須是正整數的常數。
  - ② 可由其他地方決定元素數目時，CONST\_EXPRESSION 可省略，但不可省略中括號，此形式**只能**在宣告時同時初始化陣列、將陣列宣告為函式參數，或已宣告其參考時才可以使用。

# 陣列的宣告 (Declaration of Arrays)

- 在宣告時不初始化陣列，陣列元素為 garbage value : `int a[5];`

# 陣列的宣告 (Declaration of Arrays)

- 在宣告時初始化陣列 (給定陣列初始值串列) :

```
int a[5] = {12, 56, 7, 18, -3};
```

a[0]	a[1]	a[2]	a[3]	a[4]
12	56	7	18	-3

- 陣列大小大於初始化的數值個數，未指定的元素會初始化為 0 :

```
int a[5] = {12, 56, 7};
```

a[0]	a[1]	a[2]	a[3]	a[4]
12	56	7	0	0



# 陣列的宣告 (Declaration of Arrays)

- 將陣列初始化為 0：

**int a[5] = {0};**

a[0]	a[1]	a[2]	a[3]	a[4]
0	0	0	0	0

- 在 C99 版本以後的 C 語言，初始值串列可以跳著指定第 N 個元素要被初始化的值：

**int a[5] = {12, [2] = 7, [4] = -3};**

a[0]	a[1]	a[2]	a[3]	a[4]
12	0	7	0	-3

# 陣列的宣告 (Declaration of Arrays)

- 如果省略陣列大小，則此陣列大小等於初始值串列的元素個數：  
**int a[] = {12, 56, 7, 18, -3};**

a[0]	a[1]	a[2]	a[3]	a[4]
12	56	7	18	-3

- **int a[] = {12, [2] = 7, [4] = -3};**

a[0]	a[1]	a[2]	a[3]	a[4]
12	0	7	0	-3

# 陣列的宣告 (Declaration of Arrays)

- 注意，只有在宣告陣列時，才能使用大括號初始化陣列。且陣列不能互相指派。

```
1 int main() {  
2     int foo[10] = {1, 2, 3, 4, 5};  
3     int bar[10];  
4     bar = foo; //ERROR  
5     bar = {1, 2, 3, 4, 5}; //ERROR  
6 }
```

## 陣列的存取 Access of Arrays

# 陣列的存取 (Access of Arrays)

- 陣列是一群具有相同型別的連續記憶體位置。若要引用陣列的某個位置或元素，需指定陣列名稱以及此元素在陣列中的位置編號 (position number，或稱 index、subscript)。
- 每個陣列最開頭的元素均是第零個元素 (zeroth element)。若有一個陣列 a 的大小為 10，代表它有 10 個元素，我們可以使用中括號 ([]) 來引用陣列裡的元素 (編號 0 到 9)，例如陣列 a 編號為 5 的元素寫成 a[5]。

# 陣列的存取 (Access of Arrays)

- 中括號內的 index 可以是整數或整數變數或運算式。

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


C:\Projects\ch06\_code\array.i

```
a[1] = 2
a[1] = 4
a[1] = 5
```

# 陣列的存取 (Access of Arrays)

- 陣列輸入輸出範例：

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



```
C:\Projects\ch06_code\array_i x + v
Enter 5 integers: 3 9 -4 3 0
The array is: 3 9 -4 3 0
```

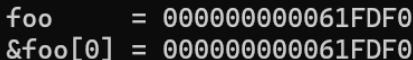
## 傳遞陣列給函式 Passing Array to Functions



# 傳遞陣列給函式 (Passing Array to Functions)

- 在傳遞陣列給函式之前，須了解陣列的名稱代表陣列的第一個元素的位址。

```
1  #include <stdio.h>
2  #include <stdlib.h>
3
4  int main() {
5      int foo[10];
6      printf("foo      = %p\n", foo);
7      printf("&foo[0] = %p\n", &foo[0]);
8  }
```



```
foo      = 000000000061FDF0
&foo[0] = 000000000061FDF0
```

# 傳遞陣列給函式 (Passing Array to Functions)

- 陣列可作為函式的參數，傳遞陣列時只需在引數列寫上陣列名稱，不須寫上中括號。

```
1  #include <stdio.h>
2  #include <stdlib.h>
3
4  int arraySum(int arr[], int n) {
5      int i, sum = 0;
6      for (i = 0; i < n; i++) sum += arr[i];
7      return sum;
8  }
9
10 int main() {
11     int foo[10] = {1, 2, 3, 4, 5, 6, 7, 8, 9, 10};
12     printf("The sum of the array is %d\n", arraySum(foo, 10));
13 }
```

C:\Projects\ch06\_code\array\_1 X + v

The sum of the array is 55

# 傳遞陣列給函式 (Passing Array to Functions)

- 傳遞陣列是傳址呼叫，函式參數的陣列與引數的陣列是相同的位址，在函式更動參數陣列的內容時，也等同變更原本呼叫時的陣列。

```
1  #include <stdio.h>
2  #include <stdlib.h>
3
4  void arrayAddInt(int arr[], int n, int val) {
5      int i;
6      for (i = 0; i < n; i++) arr[i] += val;
7  }
8
9  int main() {
10     int foo[10] = {1, 2, 3, 4, 5, 6, 7, 8, 9, 10};
11     arrayAddInt(foo, 10, 5);
12     for (int i = 0; i < 10; i++) printf("%d ", foo[i]);
13     printf("\n");
14 }
```

C:\Projects\ch06\_code\array\_1

6 7 8 9 10 11 12 13 14 15

## 多維陣列 Multidimensional Arrays

## 二維陣列

- 多維陣列能夠擁有兩個以上的 index。
- 使用兩個 index 的陣列，稱為二維陣列 (two dimensional arrays，或 double-subscripted arrays)，經常會用來表示表格 (tables)，其下標與對應的元素可模擬表格中的行 (columns) 列 (rows)。

# 多維陣列 (Multidimensional Arrays)

- 二維陣列的第一個 index 代表列，第二個 index 代表行。

	第 0 行	第 1 行	第 2 行	第 3 行
第 0 列	a[0][0]	a[0][1]	a[0][2]	a[0][3]
第 1 列	a[1][0]	a[1][1]	a[1][2]	a[1][3]
第 2 列	a[2][0]	a[2][1]	a[2][2]	a[2][3]

## 二維陣列的宣告與初始化

- 由巢狀大括號表示其結構

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

1	2	3
4	5	6

- 未指定數值的元素會初始化為 0

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

1	2	0
4	0	0

# 多維陣列 (Multidimensional Arrays)

- 由一維陣列形式，數值會依序填入元素

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

1	2	3
4	0	0

- 用一維陣列形式初始化為 0

`int a[2][3] = {0};`

0	0	0
0	0	0



# 多維陣列 (Multidimensional Arrays)

- 若宣告時即初始化，或是作為函式參數，陣列的第一個維度大小可以省略，但是陣列的其他維度大小均不能省略。

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

1	2	0
4	0	0

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

1	2	3
4	0	0

- `int a[2][ ] = {{1, 2, 3}, {4, 5, 6}}; // ERROR`
- `int a[ ][ ] = {{1, 2, 3}, {4, 5, 6}}; // ERROR`

## 多維陣列

- 由二維陣列的初始化敘述可以觀察到，多維陣列可視為元素為“數組”的陣列。如 `int a[2][3];` 即可視為元素為 `int[3]` 的大小為 2 的陣列。
- 由此可推知，若有三維陣列 `int b[4][5][6];` 即可視為元素為 `int[5][6]` 的大小為 4 的陣列，而其中，`int[5][6]` 又是元素為 `int[6]` 的大小為 5 的陣列。

# 多維陣列 (Multidimensional Arrays)

- 多維陣列的第一個維度大小代表此陣列中，以“數組”為基本單位時陣列的大小，而第二個以後的維度大小則代表“數組”本身的結構。
- 在宣告多維陣列時，不能省略決定“數組”這個基本單位的維度大小。因此，在宣告時只可以省略多維陣列的第一個維度大小，但是其他維度大小均不能省略。

## 氣泡排序法 Bubble Sort

# 氣泡排序法 (Bubble Sort)

- 排序 (Sorting) 資料 (也就是照特定的順序放置資料，例如遞增或遞減順序) 是電腦最重要的應用之一。
- 氣泡排序法 (bubble sort) 是眾多排序法之中最容易理解的一個。

# 氣泡排序法 (Bubble Sort)

- 氣泡排序法只依序重複比較相鄰的 2 個元素，並使順序錯誤的兩個元素交換位置。
- 假設要排序 9, 6, 2, 5, 3，以下是氣泡排序法的過程。

step 0	9	6	2	5	3
step 1-1	6	9	2	5	3
step 1-2	6	2	9	5	3
step 1-3	6	2	5	9	3
step 1-4	6	2	5	3	9
step 2-1	2	6	5	3	9
step 2-2	2	5	6	3	9
step 2-3	2	5	3	6	9
step 3-1	2	5	3	6	9
step 3-2	2	3	5	6	9
step 4-1	2	3	5	6	9

# 氣泡排序法 (Bubble Sort)

- 氣泡排序法的虛擬碼：

```
1 FUNCTION BubbleSort(array, n):  
2   FOR i := 0 ; i < n - 1; i++: // i = 已經確定不動的元素個數  
3     FOR j := 0 ; j < n - i - 1 ; j++:  
4       IF array[j] > array[j + 1]:  
5         swap array[j] and array[j + 1]  
6 END FUNCTION
```

## 二分搜尋法 Binary Search



# 二分搜尋法 (Binary Search)

- 搜尋演算法的目的是找出數列中是否含有符合某個關鍵值 (key value) 的元素，並能知道該元素的 index。最簡單暴力的搜尋演算法是線性搜尋法 (linear search)，即走訪陣列中的每個元素，並判斷元素的內容是否符合關鍵值。
- 線性搜尋的虛擬碼：

```
1 FUNCTION LinearSearch(array, n, target):  
2     i := 0;  
3     DO {  
4         IF array[i] == target:  
5             RETURN i;  
6         i++;  
7     } WHILE (i < n)  
8     RETURN -1; // -1 means the target is not found  
9 END FUNCTION
```

# 二分搜尋法 (Binary Search)

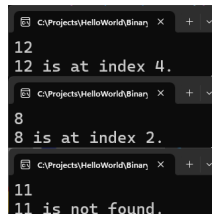
- 對於**已排序**的陣列，可使用比線性搜尋法來的更加快速的二分搜尋法 (binary search)。
- 二分搜尋法每個回合會先將數列的中央數字與目標數字相比，若中央數字不等於目標數字，則保留目標數值可能會存在的那半邊，下一個回合使用上個回合所保留的區間繼續縮小範圍。
- 假設要在 1, 3, 8, 9, 12, 18, 23 中找出 12：

step 0		1	3	8	9	12	18	23		9 < 12
step 1		1	3	8	9	12	18	23		18 > 12
step 2		1	3	8	9	12	18	23		12 = 12

# 二分搜尋法 (Binary Search)

## ● 二分搜尋法遞迴版本範例：

```
1 #include <stdio.h>
2 #include <stdlib.h>
3
4 int binary_search(const int arr[], int start, int end, int key)
5 {
6     if (start > end)
7         return -1;
8
9     int mid = start + (end - start) / 2;    //直接平均可能會溢位，所以用此算法
10    if (arr[mid] > key)
11        return binary_search(arr, start, mid - 1, key);
12    else if (arr[mid] < key)
13        return binary_search(arr, mid + 1, end, key);
14    else
15        return mid;    //最後檢測相等是因為多數搜尋狀況不是大於要不就小於
16 }
17
18 int main()
19 {
20     int a[] = {1, 3, 8, 9, 12, 18, 23}, find;
21     scanf("%d", &find);
22     int findIndex = binary_search(a, 0, 6, find);
23     if (findIndex == -1) printf("%d is not found.\n", find);
24     else printf("%d is at index %d.\n", find, findIndex);
25 }
```



```
C:\Projects\HelloWorld\Binan x + v
12
12 is at index 4.

C:\Projects\HelloWorld\Binan x + v
8
8 is at index 2.

C:\Projects\HelloWorld\Binan x + v
11
11 is not found.
```

# Q & A