Chapter 6 – Arrays (陣 列)

Outline

Objectives

- In this chapter, you will learn:
  - To introduce the array data structure.
  - To understand the use of arrays to store, sort and search lists and tables of values.
  - To understand how to
    - define an array,
    - initialize an array and
    - refer to individual elements of an array.
  - To be able to pass arrays to functions.
  - To understand basic sorting techniques.
  - To be able to define and manipulate multiple subscript arrays.

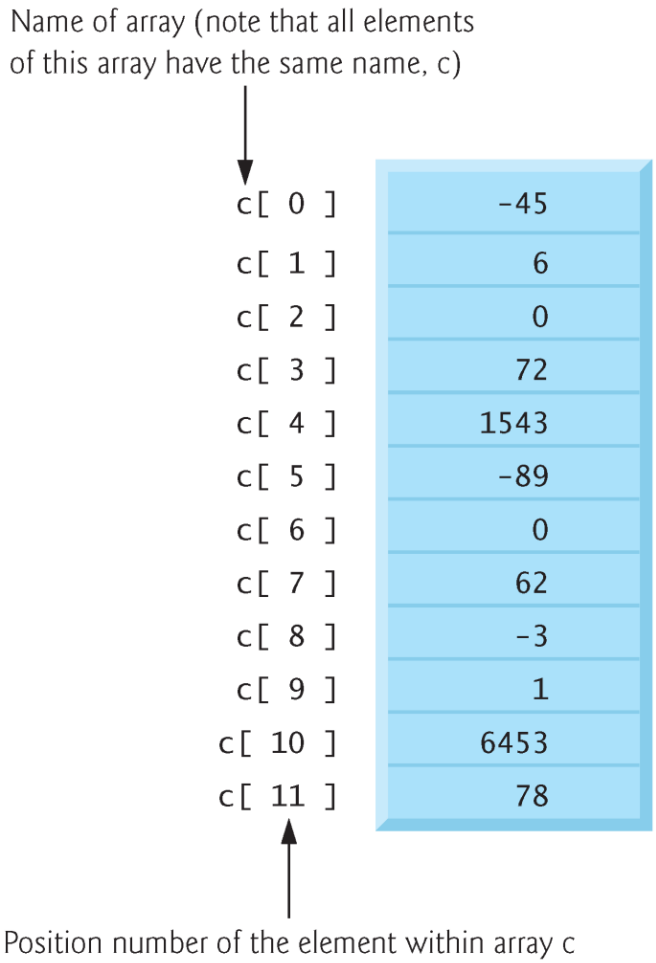
6.1 Introduction

- Arrays
  - Structures of related data items
  - Static entity – same size throughout program
  - Dynamic data structures discussed in Chapter 12

6.2 Arrays

- Array
  - Group of consecutive memory locations
  - Same name and type
- To refer to an *array element*, need to specify
  - **Array name**（陣列名稱）
  - **Position number** (位置編號), or the subscript
- Format:

  arrayname[*position number*]
  - First element at position 0 (編號從 0 開始)
  - n element array named c:
    - `c[ 0 ], c[ 1 ]...c[ n - 1 ]`



6.2 Arrays

Array elements are like **normal variables**

```
c[ 0 ] =  3;
printf( "%d", c[ 0 ] );
```

Can perform operations in subscript.

If x equals 3, then `c[ 5 - 2 ]`, `c[ 3 ]` and `c[ x ]` are the same

If a = 5 and b = 6, then the statement

```
c[ a + b ] += 2;
```

adds 2 to array element c[11].

6.2 Arrays

Operators						Associativity	Type
[]	()					left to right	highest
++	--	!	(*type*)			right to left	unary
*	/	%				left to right	multiplicative
+	-					left to right	additive
<	<=	>	>=			left to right	relational
==	!=					left to right	equality
&&						left to right	logical AND
||						left to right	logical OR
?:						right to left	conditional
=	+=	-=	*=	/=	%=	right to left	assignment
,						left to right	comma

Fig. 6.2 | Operator precedence and associativity.

6.3 Defining Arrays

- When defining arrays, specify
  - Name
  - Type of array
  - Number of elements

```
arrayType arrayName[ numberOfElements ];
```

  - Examples:

```
int c[ 10 ];
float myArray[ 3284 ];
```

- Defining multiple arrays of same type
  - Format similar to regular variables
  - Example:

```
int b[ 100 ], x[ 27 ];
```

Initializers

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

If not enough initializers, **rightmost elements become 0**

```
int n[ 5 ] = { 0 };/* { 0, 0, 0, 0, 0 } */
int n[ 5 ] = { 1 };/* { 1, 0, 0, 0, 0 } */
```

If too many, a syntax error is produced

- `int n[ 5 ] = { 1, 2, 3, 4, 5, 6 };`
  `/*error!*/`

C arrays have no bounds checking

If size omitted, initializers determine it

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

- 5 initializers, therefore 5 element array

Defining an Array and Using a Loop to Initialize the Array's Elements

fig06_03.c

```
1  /* Fig. 6.3: fig06_03.c
2     initializing an array */
3  #include <stdio.h>
4
5  /* function main begins program execution */
6  int main()
7  {
8     int n[ 10 ]; /* n is an array of 10 integers */
9     int i;       /* counter */
10
11    /* initialize elements of array n to 0 */
12    for ( i = 0; i < 10; i++ ) {
13       n[ i ] = 0; /* set element at location i to 0 */
14    } /* end for */
15
16    printf( "%s%13s\n", "Element", "Value" );
17
18    /* output contents of array n in tabular format */
19    for ( i = 0; i < 10; i++ ) {
20       printf( "%7d%13d\n", i, n[ i ] );
21    } /* end for */
22
23    return 0; /* indicates successful termination */
24
25 } /* end main */
```

宣告含十個元素的整數陣列

若改為 for (i = 0 ; i <= 10; i++) 會如何？

此處 “Element” 與 “Value” 為兩個字串，在 printf 中利用 %s 把字串輸出到螢幕；注意第二個 %13s 中 13 表示給 13 個空位列印 “Value” 字串。

```
Element         Value
      0             0
      1             0
      2             0
      3             0
      4             0
      5             0
      6             0
      7             0
      8             0
      9             0
```

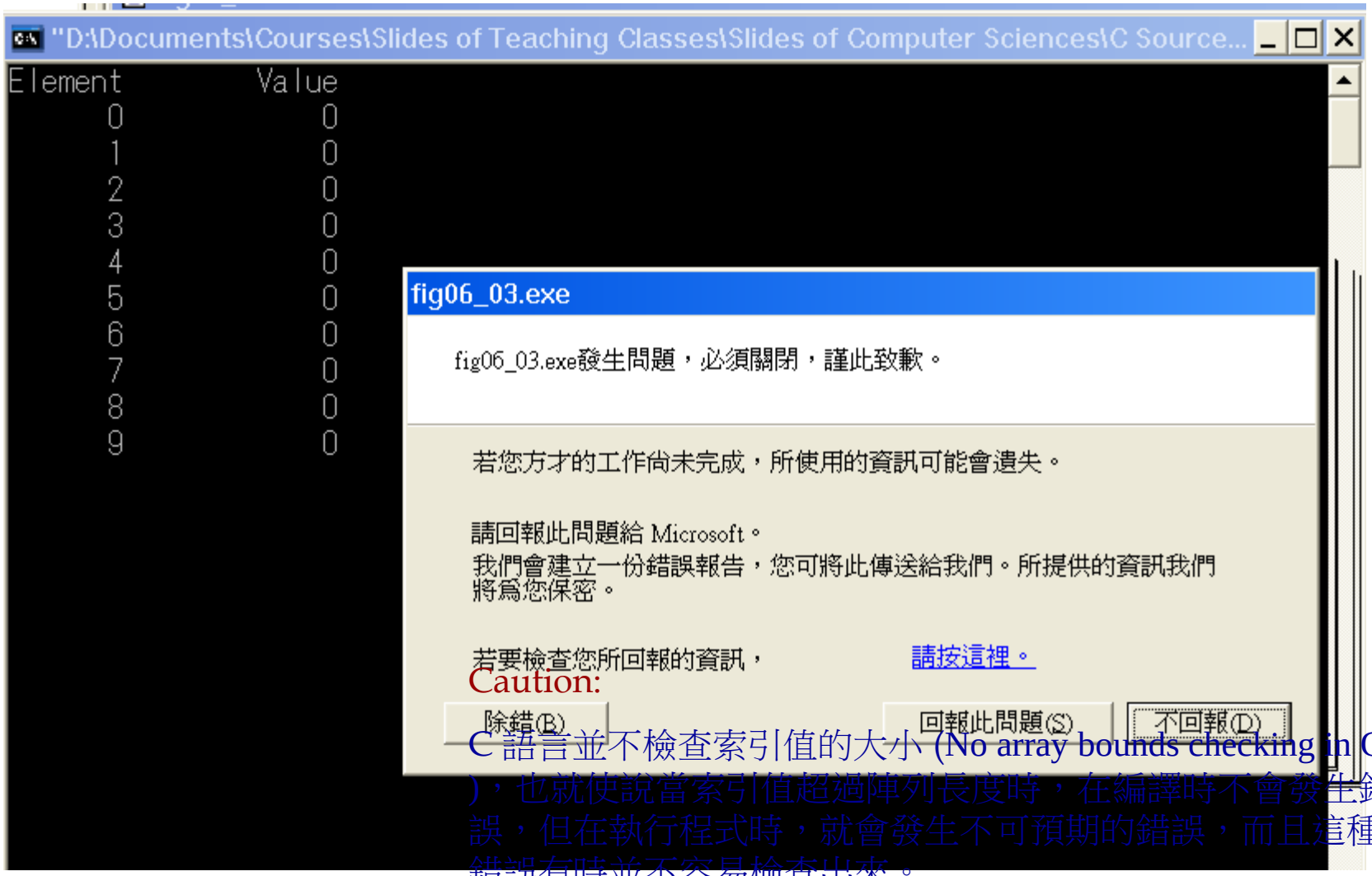
Program Output

若改為 for (i = 0 ; i <= 10; i++) 會如何？

```
/* initialize elements of array n to 0 */
for ( i = 0; i <= 10; i++ ) {
   n[ i ] = 0;
}
```

Caution:

C 語言並不檢查索引值的大小 (No array bounds checking in C!)，也就使說當索引值超過陣列長度時，在編譯時不會發生錯誤，但在執行程式時，就會發生不可預期的錯誤，而且這種錯誤有時並不容易檢查出來。



Caution:

C 語言並不檢查索引值的大小 (No array bounds checking in C!)，也就使說當索引值超過陣列長度時，在編譯時不會發生錯誤，但在執行程式時，就會發生不可預期的錯誤，而且這種錯誤有時並不容易檢查出來。

Initializing an Array in a Definition with an Initializer List

```
1  /* Fig. 6.4: fig06_04.c
2     Initializing an array with an initializer list */
3  #include <stdio.h>
4
5  /* function main begins program execution */
6  int main()
7  {
8     /* use initializer list to initialize array n */
9     int n[ 10 ] = { 32, 27, 64, 18, 95, 14, 90, 70, 60, 37 };
10    int i; /* counter */
11
12    printf( "%s%13s\n", "Element", "Value" );
13
14    /* output contents of array in tabular format */
15    for ( i = 0; i < 10; i++ ) {
16       printf( "%7d%13d\n", i, n[ i ] );
17    } /* end for */
18
19    return 0; /* indicates successful termination */
20
21 } /* end main */
```

fig06_04.c

宣告含十個元素整數陣列，同時設定各元素的值

```
Element         Value
      0            32
      1            27
      2            64
      3            18
      4            95
      5            14
      6            90
      7            70
      8            60
      9            37
```

Program Output

Specifying an Array's Size with a **Symbolic Constant** and Initializing Array Elements with Calculations

fig06_05.c

```
1  /* Fig. 6.5: fig06_05.c
2     Initialize the elements of array s to the even integers from 2 to 20 */
3  #include <stdio.h>
4  #define SIZE 10
5
6  /* function main begins program execution */
7  int main()
8  {
9     /* symbolic constant SIZE can be used to spe
10    int s[ SIZE ]; /* array s has 10 elements *
11    int j;         /* counter */
12
13    for ( j = 0; j < SIZE; j++ ) { /* set the v
14       s[ j ] = 2 + 2 * j;
15    } /* end for */
16
17    printf( "%s%13s\n", "Element", "Value" );
18
19    /* output contents of array s in tabular fo
20    for ( j = 0; j < SIZE; j++ ) {
21       printf( "%7d%13d\n", j, s[ j ] );
22    } /* end for */
23
24    return 0; /* indicates successful termination */
25
26 } /* end main */
```

`#define SIZE 10`

前置處理器指令 (preprocessor directive)

Define a symbolic constant (符號常數) SIZE whose value is 10

Symbolic constant SIZE in the code is replaced by text 10 before compilation

Makes programs more scalable (將來如果要調整陣列的大小，只需重設 SIZE 的值即可)

習慣上 symbolic constant names 使用大寫字母

下面兩種是錯誤的寫法 (不可加「分號」；及「等號」=)

```
#define SIZE = 10
#define SIZE 10;
```

```
Element         Value
      0             2
      1             4
      2             6
      3             8
      4            10
      5            12
      6            14
      7            16
      8            18
      9            20
```

Program Output

Summing the Elements of an Array

fig06_06.c

```
1  /* Fig. 6.6: fig06_06.c
2     Compute the sum of the elements of the array */
3  #include <stdio.h>
4  #define SIZE 12
5
6  /* function main begins program execution */
7  int main()
8  {
9     /* use initializer list to initialize array */
10    int a[ SIZE ] = { 1, 3, 5, 4, 7, 2, 99, 16, 45, 67, 89, 45 };
11    int i;          /* counter */
12    int total = 0; /* sum of array */
13
14    /* sum contents of array a */
15    for ( i = 0; i < SIZE; i++ ) {
16       total += a[ i ];
17    } /* end for */
18
19    printf( "Total of array element values is %d\n", total );
20
21    return 0; /* indicates successful termination */
22
23 } /* end main */
```

在此計算各元素之和；

```
total = total + a [ i ];
```

```
Total of array element values is 383
```

Using Arrays to Summarize Survey Results

Forty students were asked to rate the quality of the food in the student cafeteria on a scale of 1 to 10. Place the 40 responses in an integer array and summarize the results of the poll.

```
1  /* Fig. 6.7: fig06_07.c
2     Student poll program */
3  #include <stdio.h>
4  #define RESPONSE_SIZE 40 /* define array sizes */
5  #define FREQUENCY_SIZE 11
6
7  /* function main begins program execution */
8  int main()
9  {
10    int answer; /* counter */
11    int rating; /* counter */
12
13    /* initialize frequency counters to 0 */
14    int frequency[ FREQUENCY_SIZE ] = { 0 };
15
16    /* place survey responses in array responses */
17    int responses[ RESPONSE_SIZE ] = { 1, 2, 6, 4, 8, 5, 9, 7, 8, 10,
18         1, 6, 3, 8, 6, 10, 3, 8, 2, 7, 6, 5, 7, 6, 8, 6, 7, 5, 6, 6,
19         5, 6, 7, 5, 6, 4, 8, 6, 8, 10 };
20
```

本範例中有兩個分別為 **responses** 及 **frequency** 的陣列；

responses 為個別學生所打的分數所組成的陣列；**frequency** 則為出現某分數之次數所組成的。

例如 **frequency[1]** 為將分數打為 1 的學生人數 (次數) 。由於共有 10 種分數，故將 **frequency** 的陣列定為 11 ；如此就有

frequency[1]
frequency[2]
. . .
frequency[9]
frequency[10]

十種分數的個別次數。

在這裡 **frequency[0]** 並未使用。

fig06_07.c (Part 1 of 2)

```
21    /* for each answer, select value of an element of array responses
22       and use that value as subscript in array frequency to
23       determine element to increment */
24    for ( answer = 0; answer < RESPONSE_SIZE; answer++ ) {
25       ++frequency[ responses [ answer ] ];
26    } /* end for */
27
28    /* display results */
29    printf( "%s%17s\n", "Rating", "Frequency" );
30
31    /* output frequencies in tabular format */
32    for ( rating = 1; rating < FREQUENCY_SIZE; rating++ ) {
33       printf( "%6d%17d\n", rating, frequency[ rating ] );
34    } /* end for */
35
36    return 0; /* indicates successful termination */
37
38 } /* end main */
```

fig06_07.c (Part 2 of 2)

Key Statement:

```
++frequency[ responses[ answer ] ];
```

```
Rating        Frequency
     1                2
     2                2
     3                2
     4                2
     5                5
     6               11
     7                5
     8                7
     9                1
    10                3
```

Caution:

C 語言並不檢查索引值的大小 (No array bounds checking in C!)，也就使說當索引值超過陣列長度時，在編譯時不會發生錯誤，但在執行程式時，就會發生不可預期的錯誤，而且這種錯誤有時並不容易檢查出來。

What if `responses[23] = 13` ?

Graphing Array Element Values with Histograms

Reads numbers from an array and graphs the information in the form of a bar chart or histogram

```
Element         Value      Histogram
      0            19      *******************
      1             3      ***
      2            15      ***************
      3             7      *******
      4            11      ***********
      5             9      *********
      6            13      *************
      7             5      *****
      8            17      *****************
      9             1      *
```

印一顆星星用
例如要印出

3　　7　　*******

n[3]　　星星數

就可用此段迴圈。

```
1  /* Fig. 6.8: fig06_08.c
2     Histogram printing program */
3  #include <stdio.h>
4  #define SIZE 10
5
6  /* function main begins program execution */
7  int main()
8  {
9     /* use initializer list to initialize array n */
10    int n[ SIZE ] = { 19, 3, 15, 7, 11, 9, 13, 5, 17, 1 };
11    int i; /* outer counter */
12    int j; /* inner counter */
13
14    printf( "%s%13s%17s\n", "Element", "Value", "Histogram" );
15
16    /* for each element of array n, output a bar in histogram */
17    for ( i = 0; i < SIZE; i++ ) {
18       printf( "%7d%13d        ", i, n[ i ]) ;
19
20       for ( j = 1; j <= n[ i ]; j++ ) { /* print one bar */
21          printf( "%c", '*' );
22       } /* end inner for */
23
```

8 spaces

```
24       printf( "\n" ); /* start next line of output */
25    } /* end outer for */
26
27    return 0; /* indicates successful termination */
28
29 } /* end main */
```

```
Element         Value         Histogram
      0            19         ********************
      1             3         ***
      2            15         ***************
      3             7         *******
      4            11         ***********
      5             9         *********
      6            13         *************
      7             5         *****
      8            17         *****************
      9             1         *
```

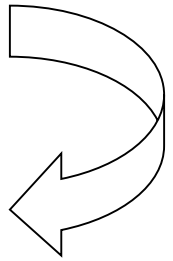
fig06_08.c (Part 2 of 2)

Program Output

Rolling a Die 6000 Times and Summarizing the Results in an Array (in Chapter 5)

```
23    /* determine face value and increment appropriate counter */
24    switch ( face ) {
25
26       case 1:          /* rolled 1 */
27          ++frequency1;
28          break;
29
30       case 2:          /* rolled 2 */
31          ++frequency2;
32          break;
33
34       case 3:          /* rolled 3 */
35          ++frequency3;
36          break;
37
38       case 4:          /* rolled 4 */
39          ++frequency4;
40          break;
41
42       case 5:          /* rolled 5 */
43          ++frequency5;
44          break;
45
46       case 6:          /* rolled 6 */
47          ++frequency6;
48          break;
49    } /* end switch */
```

Recall that in Chapter 5 (Fig. 5.8):



擲骰子後，每一個點數就有一個出現的次數：

Six Variables:

- frequency1
- frequency2
- frequency3
- frequency4
- frequency5
- frequency6

Rolling a Die 6000 Times and Summarizing the Results in an Array

```
1  /* Fig. 6.9: fig06_09.c
2     Roll a six-sided die 6000 times */
3  #include <stdio.h>
4  #include <stdlib.h>
5  #include <time.h>
6  #define SIZE 7
7
8  /* function main begins program execution */
9  int main()
10 {
11    int face;                    /* random number
12    int roll;                    /* roll counter
13    int frequency[ SIZE ] = { 0 }; /* initialize
14
15    srand( time( NULL ) ); /* seed random-numbe
16
17    /* roll die 6000 times */
18    for ( roll = 1; roll <= 6000; roll++ ) {
19       face = rand() % 6 + 1;
20       ++frequency[ face ]; /* replaces 26-line
21    } /* end for */
22
```

NOTE: SIZE 7,

```
24    switch ( face ) {
25
26       case 1:          /* rolled 1 */
27          ++frequency1;
28          break;
29
30       case 2:          /* rolled 2 */
31          ++frequency2;
32          break;
33
34       case 3:          /* rolled 3 */
35          ++frequency3;
36          break;
37
38       case 4:          /* rolled 4 */
39          ++frequency4;
40          break;
41
42       case 5:          /* rolled 5 */
43          ++frequency5;
44          break;
45
46       case 6:          /* rolled 6 */
47          ++frequency6;
48          break;
49    } /* end switch */
```

1

在
f
f
.
、

此處用 ++frequency[face];

取代上一頁的 switch 程式部分。

```
23    printf( "%s%17s\n", "Face", "Frequency" );
24
25    /* output frequency elements 1-6 in tabular format */
26    for ( face = 1; face < SIZE; face++ ) {
27       printf( "%4d%17d\n", face, frequency[ face ] );
28    } /* end for */
29
30    return 0; /* indicates successful termination */
31
32 } /* end main */
```

```
Face        Frequency
   1             1029
   2              951
   3              987
   4             1033
   5             1010
   6              990
```

fig06_09.c (Part 2 of 2)

Character and String

- 字元型態 (**char**)
  - char c = 'A'; /* A 為一字元常數，用**單引號**包圍 */
- 字串(string)
  - "Sweet Home" , "ChE NCKU" , "first"
  - 字串要用**雙引號**包住，如 "ChE NCKU" 。
- **String** “first” is really **a static array of characters**,字串是由**字元陣列**所組成，所以處理字串就是在處理陣列中之字元

Character Array (字元陣列)

- ***Character arrays*** can be initialized using string literals, *e.g.*,

```
char string1[ ] = "first";
```

- ***Null character*** **'\0'** terminates strings ('\0' 可視為字串結束符號). string1 actually has 6 elements. It is equivalent to:

```
char string1[ ] = { 'f', 'i', 'r', 's', 't', '\0' };
```

- ***Individual characters*** can be accessed, i.e., string1[3] is character 's'

Character Array (字元陣列)

Array name is ***address*** of array, so & is not needed for scanf

```
scanf( "%s", string2 );
```

- Reads characters until *whitespace* (such as tabs, newlines, spaces) encountered；(因此用 scanf 讀字串時無法讀取字串中之空白)
- 如果要輸入含空白的字串，可用 gets(string2); 來讀取字串。
- 輸入與輸出字串(字元陣列)皆用 %s；注意：輸入及輸出字元則用 %c ！

Using Character Arrays to Store and Manipulate Strings

fig06_10.c (Part 1 of 2)

```
1  /* Fig. 6.10: fig06_10.c
2     Treating character arrays as strings */
3  #include <stdio.h>
4
5  /* function main begins program execution */
6  int main()
7  {
8     char string1[ 20 ];                  /* reserves 20 characters */
9     char string2[] = "string literal"; /* reserves 15 characters */
10    int i;                               /* counter */
11
12    /* read string from user into array string
13    printf("Enter a string: ");
14    scanf( "%s", string1 );
15
16    /* output strings */
17    printf( "string1 is: %s\nstring2 is: %s\n"
18            "string1 with spaces between characters is:\n",
19            string1, string2 );
20
21    /* output characters until null character is reached */
22    for ( i = 0; string1[ i ] != '\0'; i++ ) {
23       printf( "%c ", string1[ i ] );
24    } /* end for */
25
```

算算看 "string literal" 有幾個字元？

如果輸入 "Hello there" ，string1 會有哪些字元？

輸入與輸出字串(字元陣列)皆用 %s；注意：輸入及輸出字元則用 %c ！

A space!

```
26    printf( "\n" );
27
28    return 0; /* indicates successful termination */
29
30 } /* end main */
```

```
Enter a string: Hello there
string1 is: Hello
string2 is: string literal
string1 with spaces between characters is:
H e l l o
```

fig06_10.c (Part 2 of 2)

Static Local Arrays and Automatic Local Arrays

```
1  /* Fig. 6.11: fig06_11.c
2     Static arrays are initialized to zero */
3  #include <stdio.h>
4
5  void staticArrayInit( void );    /* function prototype */
6  void automaticArrayInit( void ); /* function prototype */
7
8  /* function main begins program execution */
9  int main()
10 {
11    printf( "First call to each function:\n" );
12    staticArrayInit();
13    automaticArrayInit();
14
15    printf( "\n\nSecond call to each function:\n" );
16    staticArrayInit();
17    automaticArrayInit();
18
19    return 0; /* indicates successful termination */
20
21 } /* end main */
22
```

就像變數分靜態變數與區域變數一般，陣列也有 static local array 及 automatic local arrays 的區別。

fig06_11.c (Part 1 of 3)

```
23 /* function to demonstrate a static local array */
24 void staticArrayInit( void )
25 {
26    /* initializes elements to 0 first time function is called */
27    static int array1[ 3 ];
28    int i; /* counter */
29
30    printf( "\nValues on entering staticArrayInit:\n" );
31
32    /* output contents of array1 */
33    for ( i = 0; i <= 2; i++ ) {
34       printf( "array1[ %d ] = %d  ", i, array1[ i ] );
35    } /* end for */
36
37    printf( "\nValues on exiting staticArrayInit:\n" );
38
39    /* modify and output contents of array1 */
40    for ( i = 0; i <= 2; i++ ) {
41       printf( "array1[ %d ] = %d  ", i, array1[ i ] += 5 );
42    } /* end for */
43
44 } /* end function staticArrayInit */
45
```

fig06_11.c (Part 2 of

宣告時若沒有給初始值時，會自動設為 0。
注意此處 array1 為靜態陣列，因此第一次進入時會自動設為 0。但離開後再進入時，就 ???

在此處陣列的內容改變了！

```
46 /* function to demonstrate an automatic local array */
47 void automaticArrayInit( void )
48 {
49    /* initializes elements each time function is called */
50    int array2[ 3 ] = { 1, 2, 3 };
51    int i; /* counter */
52
53    printf( "\n\nValues on entering automaticArrayInit:\n" );
54
55    /* output contents of array2 */
56    for ( i = 0; i <= 2; i++ ) {
57       printf("array2[ %d ] = %d  ", i, array2[ i ] );
58    } /* end for */
59
60    printf( "\nValues on exiting automaticArrayInit:\n" );
61
62    /* modify and output contents of array2 */
63    for ( i = 0; i <= 2; i++ ) {
64       printf( "array2[ %d ] = %d  ", i, array2[ i ] += 5 );
65    } /* end for */
66
67 } /* end function automaticArrayInit */
```

此處 array2 為區域陣列

在此處陣列的內容改變了！

fig06_11.c (Part 3 of 3)

Program Output

```
First call to each function:

Values on entering staticArrayInit:
array1[ 0 ] = 0  array1[ 1 ] = 0  array1[ 2 ] = 0
Values on exiting staticArrayInit:
array1[ 0 ] = 5  array1[ 1 ] = 5  array1[ 2 ] = 5

Values on entering automaticArrayInit:
array2[ 0 ] = 1  array2[ 1 ] = 2  array2[ 2 ] = 3
Values on exiting automaticArrayInit:
array2[ 0 ] = 6  array2[ 1 ] = 7  array2[ 2 ] = 8

Second call to each function:

Values on entering staticArrayInit:
array1[ 0 ] = 5  array1[ 1 ] = 5  array1[ 2 ] = 5
Values on exiting staticArrayInit:
array1[ 0 ] = 10  array1[ 1 ] = 10  array1[ 2 ] = 10

Values on entering automaticArrayInit:
array2[ 0 ] = 1  array2[ 1 ] = 2  array2[ 2 ] = 3
Values on exiting automaticArrayInit:
array2[ 0 ] = 6  array2[ 1 ] = 7  array2[ 2 ] = 8
```

Call by Value – Passing Variables to Functions

```
#include <stdio.h>

void add10(int,int);          /* add10()的原型 */

int main()
{
   int a = 3, b = 5;          /* 宣告區域變數a與b */

   printf ( "呼叫函數add10()之前: " );
   printf ( "a = %2d, b = %2d\n", a, b ); /* 印出a、b的值 */

   add10(a,b);

   printf ( "呼叫函數add10()之後: " );
   printf ( "a = %2d, b = %2d\n", a, b ); /* 印出a、b的值 */

   return 0;
}
```

```
呼叫函數add10()之前: a =  3, b =  5
函數 add10 中 :      a = 13, b = 15
呼叫函數add10()之後: a =  3, b =  5
```

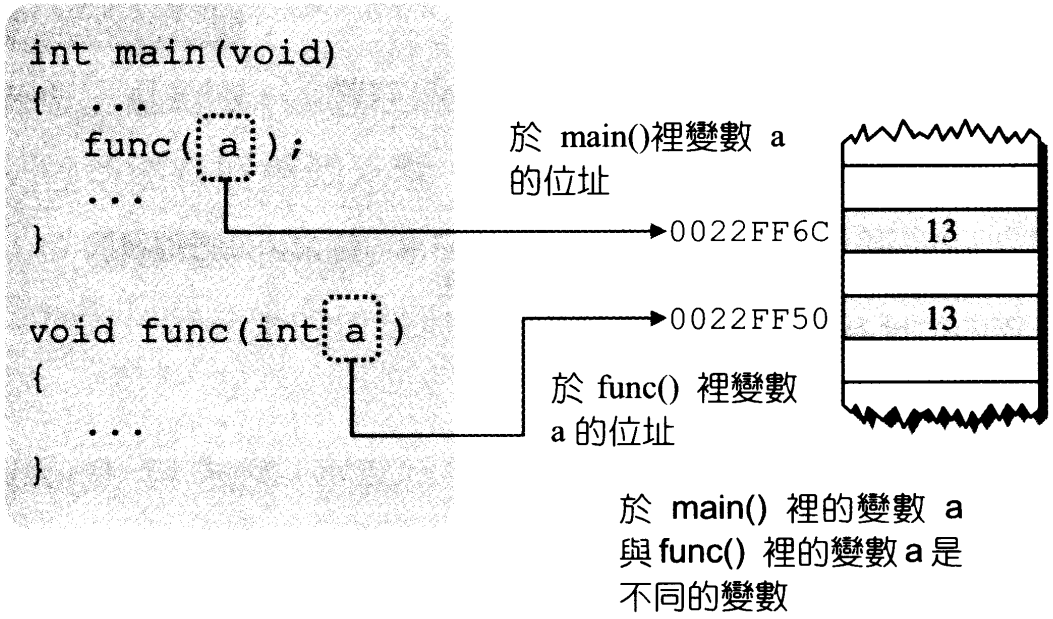
```
void add10(int a,int b)
{
   a = a + 10;                /* 將變數a的值加10之後，設回給a */
   b = b + 10;                /* 將變數b的值加10之後，設回給b */

   printf ( "函數 add10 中 :      " );
   printf ( "a = %2d, b = %2d\n", a, b ); /* 印出a、b的值
}
```

在 add10 函式中a及b這兩個變數與主程式中a,b所佔的記憶體位置不同。

Passing Variables to Functions

當我們傳遞一般變數名稱到函數時，接收的函數會將參數的內容**複製**一份到函數中變數所使用的記憶體位置中，就像函數中區域變數一樣；

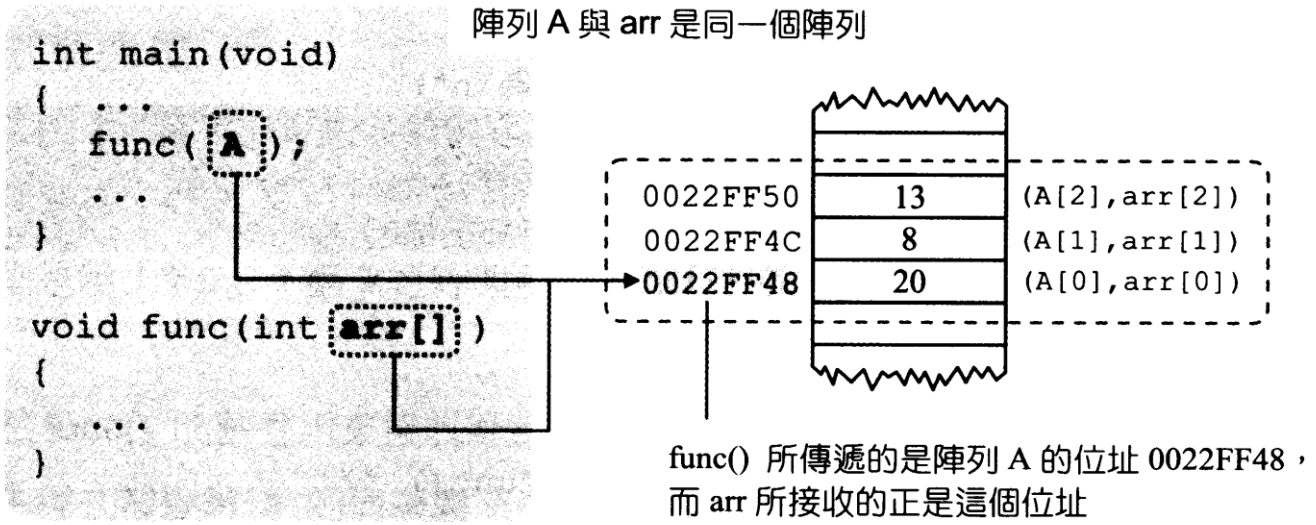
當函數結束後，原先在其他區段之變數的值並不會改變。所以叫做「**傳值呼叫 (call-by-value)**」



Passing Arrays to Functions

但當傳遞的引數為陣列名稱時，由於陣列的長度可能很大，為了考量執行效率，因此傳遞到函數中的只是該陣列存放在記憶體位置的地址，而不是將變數的內容傳到函數中，所以叫做「傳址呼叫 (call-by-reference)」

由於傳遞的是陣列的地址，被傳遞的陣列與函數中接收的陣列有相同的記憶體位置(地址)，所以兩個陣列的內容一樣；當函數結束後，原先在其他區段之陣列內容值已經改變了。



Passing Arrays and Array Elements to Functions

Passing *Arrays*: To pass an array argument to a function, specify **the name of the array without any brackets**,

```
void myFunction( int b[], int arraySize )
```

主程式中：

```
. . .
int myArray[ 24 ];
myFunction( myArray, 24 );
. . .
```

- Array size usually passed to function also

– Arrays passed *call-by-reference*

– **Name of array is address of first element**

– Function knows where the array is stored

- Modifies original memory locations

Passing *Array Elements*

– Passed by call-by-value

– Pass subscripted name (i.e., `myArray[ 3 ]`) to function

Passing Arrays to Functions

- Function Prototype

```
void modifyArray( int b[], int arraySize );
```

  - Parameter names *optional* in prototype
    - `int b[]` could be written `int []`
    - `int arraySize` could be simply `int`
    - i.e. can be written as

```
void modifyArray( int [], int );
```

Array Name = Address of the Array's First Element

fig06_12.c

```
1  /* Fig. 6.12: fig06_12.c
2     The name of an array is the same as &array[ 0 ] */
3  #include <stdio.h>
4
5  /* function main begins program execution */
6  int main()
7  {
8     char array[ 5 ]; /* define an array of size 5 */
9
10    printf( "    array = %p\n&array[0] = %p\n"
11       "   &array = %p\n",
12       array, &array[ 0 ], &array );
13
14    return 0; /* indicates successful termination */
15
16 } /* end main */
```

用 %p 列印變數或陣列在記憶體的地址

```
    array = 0012FF78
&array[0] = 0012FF78
   &array = 0012FF78
```

array 、 &array[0] 和 &array 三者的地址一樣

Passing Arrays and Individual Array Elements to Functions

fig06_13.c (Part 1 of 3)

```
1  /* Fig. 6.13: fig06_13.c
2     Passing arrays and individual array elements to functions */
3  #include <stdio.h>
4  #define SIZE 5
5
6  /* function prototypes */
7  void modifyArray( int b[], int size );
8  void modifyElement( int e );
9
10 /* function main begins program execution */
11 int main()
12 {
13    int a[ SIZE ] = { 0, 1, 2, 3, 4 }; /* initialize a */
14    int i; /* counter */
15
16    printf( "Effects of passing entire array by reference:\n\nThe "
17            "values of the original array are:\n" );
18
19    /* output original array */
20    for ( i = 0; i < SIZE; i++ ) {
21       printf( "%3d", a[ i ] );
22    } /* end for */
23
24    printf( "\n" );
25
```

此處也可寫為

```
void modifyArray( int [], int );
void modifyElement( int );
```

```
26    /* pass array a to modifyArray by refere
27    modifyArray( a, SIZE );
28
29    printf( "The values of the modified array are:\n" );
30
31    /* output modified array */
32    for ( i = 0; i < SIZE; i++ ) {
33       printf( "%3d", a[ i ] );
34    } /* end for */
35
36    /* output value of a[ 3 ] */
37    printf( "\n\n\nEffects of passing array element "
38            "by value:\n\nThe value of a[3] is %d\n", a[ 3 ] );
39
40    modifyElement( a[ 3 ] ); /* pass array element a[ 3 ] by value */
41
42    /* output value of a[ 3 ] */
43    printf( "The value of a[ 3 ] is %d\n", a[ 3 ] );
44
45    return 0; /* indicates successful termination */
46
47 } /* end main */
48
```

將陣列名稱(事實上是地址)及大小傳到函數中，此時為傳址呼叫！

在這裡，傳遞的是陣列中某一個元素，此時為傳值呼叫！

fig06_13.c (Part 2 of 3)

```
/* in function modifyArray, "b" points to the original array "a"
   in memory */
void modifyArray( int b[], int size )
{
   int j; /* counter */

   /* multiply each array element by 2 */
   for ( j = 0; j < size; j++ ) {
      b[ j ] *= 2;
   } /* end for */

} /* end function modifyArray */

/* in function modifyElement, "e" is a local copy of array element
   a[ 3 ] passed from main */
void modifyElement( int e )
{
   /* multiply parameter by 2 */
   printf( "Value in modifyElement is %d\n", e *= 2 );
} /* end function modifyElement */
```

改變陣列的內容

改變整數區域變數 e 的內容

fig06_13.c (Part 3 of 3)

```
Effects of passing entire array by reference:

The values of the original array are:
  0  1  2  3  4
The values of the modified array are:
  0  2  4  6  8


Effects of passing array element by value:

The value of a[3] is 6
Value in modifyElement is 12
The value of a[ 3 ] is 6
```

Program Output

Type Qualifier - const

fig06_14.c (Part 1 of 2)

```
1  /* Fig. 6.14: fig06_14.c
2     Demonstrating the const type qualifier with arrays */
3  #include <stdio.h>
4
5  void tryToModifyArray( const int b[] ); /* function prototype */
6
7  /* function main begins program execution */
8  int main()
9  {
10    int a[] = { 10, 20, 30 }; /* initialize a */
11
12    tryToModifyArray( a );
13
14    printf("%d %d %d\n", a[ 0 ], a[ 1 ], a[ 2 ] );
15
16    return 0; /* indicates successful termination */
17
18 } /* end main */
19
```

當宣告變數或陣列時使用 const 這個**修飾詞**後，該變數或陣列就宣告成為常數或常數陣列，它們的內容在程式執行時就不容更改！

當主程式呼叫函式時，如果函式的引數(arguments)是利用陣列傳遞時，只要在函式中陣列的值改變的話，主程式中對應的陣列值也會跟著改變。如果要避免主程式的陣列值改變，這時在函式中可宣告該陣列為常數陣列。

```
20 /* in function tryToModifyArray, array b is const, so it cannot be
21    used to modify the original array a in main. */
22 void tryToModifyArray( const int b[] )
23 {
24    b[ 0 ] /= 2;     /* error */
25    b[ 1 ] /= 2;     /* error */
26    b[ 2 ] /= 2;     /* error */
27 } /* end function tryToModifyArray */
```

fig06_14.c (Part 2 of 2)

Program Output

```
Compiling...
FIG06_14.C
fig06_14.c(24) : error C2166: l-value specifies const object
fig06_14.c(25) : error C2166: l-value specifies const object
fig06_14.c(26) : error C2166: l-value specifies const object
```

在本函數中， b 已經宣告型態為 const int 的常數型整數陣列，但在程式中又試圖改變 b的內容，因此產生語法錯誤！

Sorting Arrays (排序)

- Sorting Data
  - Important computing application
  - Virtually every organization must sort some data
- Bubble Sort (氣泡排序法、Sinking Sort)
  - **Several passes** through the array
  - Successive pairs (鄰近的兩個) of elements are compared
    - If increasing order (or identical), no change
    - If decreasing order, elements exchanged
  - Repeat
- Example:
  - original: 3 4 2 6 7
  - pass 1: 3 2 4 6 7
  - pass 2: 2 3 4 6 7
  - Small elements "bubble to the top"

Sorting an Array with **Bubble Sort**

```
1  /* Fig. 6.15: fig06_15.c
2     This program sorts an array's values into ascending order */
3  #include <stdio.h>
4  #define SIZE 10
5
6  /* function main begins program execution */
7  int main()
8  {
9     /* initialize a */
10    int a[ SIZE ] = { 2, 6, 4, 8, 10, 12, 89, 68, 45, 37 };
11    int i;    /* inner counter */
12    int pass; /* outer counter */
13    int hold; /* temporary location used to swap array elements */
14
15    printf( "Data items in original order\n" );
16
17    /* output original array */
18    for ( i = 0; i < SIZE; i++ ) {
19       printf( "%4d", a[ i ] );
20    } /* end for */
21
```

fig06_15.c (Part 1 of 3)

要將本陣列之值由小至大排序

首先，列印原始陣列

```
22    /* bubble sort */
23    /* loop to control number of passes */
24    for ( pass = 1; pass < SIZE; pass++ ) {
25
26       /* loop to control number of comparisons per pass */
27       for ( i = 0; i < SIZE - 1; i++ ) {
28
29          /* compare adjacent elements and swap them if
30          element is greater than second element */
31          if ( a[ i ] > a[ i + 1 ] ) {
32             hold = a[ i ];
33             a[ i ] = a[ i + 1 ];
34             a[ i + 1 ] = hold;
35          } /* end if */
36
37       } /* end inner for */
38
39    } /* end outer for */
40
41    printf( "\nData items in ascending order\n" );
42
```

fig06_15.c (Part 2 of 3)

若只寫
`a[ i ] = a[ i + 1 ];`
`a[ i + 1 ] = a [ i ];`
會發生什麼結果？

能不能提高本程式的執行效率？

```
43    /* output sorted array */
44    for ( i = 0; i < SIZE; i++ ) {
45       printf( "%4d", a[ i ] );
46    } /* end for */
47
48    printf( "\n" );
49
50    return 0; /* indicates successful termination */
51
```

最後列印排序完成後的陣列

fig06_15.c (Part 3 of 3)

Program Output

```
Data items in original order
   2   6   4   8  10  12  89  68  45  37
Data items in ascending order
   2   4   6   8  10  12  37  45  68  89
```

Case Study: Computing Mean, Median and Mode Using Arrays

- Mean (算術平均値) – average
- Median (中間數) – number in middle of sorted list
  - 1, 2, 3, 4, 5 ⇒ 3 is the median
  - 1, 1, 1, 4, 5, 5, 10 ⇒ 4 is the median
- Mode (最常出現之値) – number that occurs most often
  - 1, 1, 1, 2, 3, 3, 4, 5 ⇒ 1 is the mode

Computing Mean, Median and Mode Using Arrays

```
1  /* Fig. 6.16: fig06_16.c
2     This program introduces the topic of survey data analysis.
3     It computes the mean, median, and  mode of the data */
4  #include <stdio.h>
5  #define SIZE 99
6
7  /* function prototypes */
8  void mean( const int answer[] );
9  void median( int answer[] );
10 void mode( int freq[], const int answer[] ) ;
11 void bubbleSort( int a[] );
12 void printArray( const int a[] );
13
14 /* function main begins program execution */
15 int main()
16 {
17    int frequency[ 10 ] = { 0 }; /* initialize array frequency */
18
```

有 99 個數值介於 1 ~ 9 的原始數據

各個函數之原型 (prototype)。

mean	平均值
median	中間數
mode	最多數
bubbleSort	陣列排序
printArray	列印陣列

fig06_16.c (Part 1 of 8)

```
19    /* initialize array response */
20    int response[ SIZE ] =
21       { 6, 7, 8, 9, 8, 7, 8, 9, 8, 9,
22         7, 8, 9, 5, 9, 8, 7, 8, 7, 8,
23         6, 7, 8, 9, 3, 9, 8, 7, 8, 7,
24         7, 8, 9, 8, 9, 8, 9, 7, 8, 9,
25         6, 7, 8, 7, 8, 7, 9, 8, 9, 2,
26         7, 8, 9, 8, 9, 8, 9, 7, 5, 3,
27         5, 6, 7, 2, 5, 3, 9, 4, 6, 4,
28         7, 8, 9, 6, 8, 7, 8, 9, 7, 8,
29         7, 4, 4, 2, 5, 3, 8, 7, 5, 6,
30         4, 5, 6, 1, 6, 5, 7, 8, 7 };
31
32    /* process responses */
33    mean( response );
34    median( response );
35    mode( frequency, response );
36
37    return 0; /* indicates successful termination */
38
39 } /* end main */
40
```

有 99 個數值介於 1 ~ 9 的原始數據

分別呼叫計算 mean, median 及 mode 的函數

fig06_16.c (Part 2 of 8)

```
41 /* calculate average of all response values */
42 void mean( const int answer[] )
43 {
44    int j;          /* counter */
45    int total = 0; /* variable to hold sum of array elements */
46
47    printf( "%s\n%s\n%s\n", "********", "  Mean", "********" );
48
49    /* total response values */
50    for ( j = 0; j < SIZE; j++ ) {
51       total += answer[  j  ];
52    } /* end for */
53
54    printf( "The mean is the average value of the data\n"
55            "items. The mean is equal to the total of\n"
56            "all the data items divided by the number\n"
57            "of data items ( %d ). The mean value for\n"
58            "this run is: %d / %d = %.4f\n\n",
59            SIZE, total, SIZE, (  double  ) total / SIZE );
60 } /* end function mean */
61
```

本函數中陣列 answer 的內容不會改變，故可用 const 修飾字

fig06_16.c (Part 3 of 8)

計算總和

計算平均值，注意此處用 (double) 把整數型態的數據轉成倍精準浮點數。

在函數中，輸入陣列的大小要如何決定？

```
62 /* sort array and determine median
63 void median( int answer[] )
64 {
65    printf( "\n%s\n%s\n%s\n%s",
66            "********", " Median", "********",
67            "The unsorted array of responses is" );
68
69    printArray( answer ); /* output unsorted array */
70
71    bubbleSort( answer ); /* sort array */
72
73    printf( "\n\nThe sorted array is" );
74    printArray( answer ); /* output sorted array */
75
76    /* display median element */
77    printf( "\n\nThe median is element %d of\n"
78            "the sorted %d element array.\n"
79            "For this run the median is %d\n\n",
80            SIZE / 2, SIZE, answer[ SIZE / 2 ] );
81 } /* end function median */
82
```

利用本函數排序並決定中間數，所以在本函數中，陣列 answer 的內容會被排序而更改，不可用 const 修飾字

fig06_16.c (Part 4 of 8)

利用 bubbleSort 函數排序

排序後找中間值

Note 99/2=49

如果陣列中元素的個數為偶數時，
本程式找出的中間值是哪個位置？

```
83 /* determine most frequent response */
84 void mode( int freq[], const int answer[] )
85 {
86    int rating;         /* counter */
87    int j;              /* counter */
88    int h;              /* counter */
89    int largest = 0;    /* represents largest frequency */
90    int modeValue = 0; /* respesents most frequent response */
91
92    printf( "\n%s\n%s\n%s\n",
93            "*********", "  Mode", "*********" );
94
95    /* initialize frequencies to 0 */
96    for ( rating = 1; rating <= 9; rating++ ) {
97       freq[ rating ] = 0;
98    } /* end for */
99
100   /* summarize frequencies */
101   for ( j = 0; j < SIZE; j++ ) {
102      ++freq[ answer[ j ] ];
103   } /* end for */
104
```

找最常出現的值的函數

先將計數器設為 0

freq[1], freq[2], . . .freq[9] 為分別出現 1, 2, . . . 9 的次數；此處即是分別計數的部份。

fig06_16.c (Part 5 of 8)

```
105   /* output headers for result columns */
106   printf( "%s%11s%19s\n\n",
107           "Response", "Frequency", "Histogram");
108
109
110   /* output results */
111   for ( rating = 1; rating <= 9; rating++ ) {
112      printf( "%8d%11d          ", rating, freq[ rating ] );
113
114      /* keep track of mode value and largest frequency value */
115      if ( freq[ rating ] > largest ) {
116         largest = freq[ rating ];
117         modeValue = rating;
118      } /* end if */
119
120      /* output histogram bar representing frequency value */
121      for ( h = 1; h <= freq[ rating ]; h++ ) {
122         printf( "*" );
123      } /* end inner for */
124
125      printf( "\n" ); /* being new line of output */
126   } /* end outer for */
127
```

fig06_16.c (Part 6 of 8)

rating = 1 到 9 的迴圈；在此迴圈中，尋找出現最多次數之值以及繪長條圖

(1) 找 freq[1], freq[2], . . .freq[9]最大值

(2) 分別畫 histogram

```
128   /* display the mode value */
129   printf( "The mode is the most frequent value.\n"
130           "For this run the mode is %d which occurred"
131           " %d times.\n", modeValue, largest );
132} /* end function mode */
133
134/* function that sorts an array with bubble sort alg
135void bubbleSort( int a[] )
136{
137   int pass; /* counter */
138   int j;    /* counter */
139   int hold; /* temporary location used to swap elements */
140
141   /* loop to control number of passes */
142   for ( pass = 1; pass < SIZE; pass++ ) {
143
144      /* loop to control number of comparisons per pass */
145      for ( j = 0; j < SIZE - 1; j++ ) {
146
147         /* swap elements if out of order */
148         if ( a[ j ] > a[ j + 1 ] ) {
149            hold = a[ j ];
150            a[ j ] = a[ j + 1 ];
151            a[ j + 1 ] = hold;
152         } /* end if */
153
```

再列印出現最多次的值及次數

Bubble Sort 部分與前面的例子相同，此處只是將它寫成函數

fig06_16.c (Part 7 of 8)

```
154      } /* end inner for */
155
156   } /* end outer for */
157
158} /* end function bubbleSort */
159
160/* output array contents (20 values per row) */
161void printArray( const int a[] )
162{
163   int j; /* counter */
164
165   /* output array contents */
166   for ( j = 0; j < SIZE; j++ ) {
167
168      if ( j % 20 == 0 ) { /* begin new line every 20 values */
169         printf( "\n" );
170      } /* end if */
171
172      printf( "%2d", a[ j ] );
173   } /* end for */
174
175} /* end function printArray */
```

利用這個函數是把陣列 a 列印出來。每行列印 20 個元素。

fig06_16.c (Part 8 of 8)

Program Output

```
********
  Mean
********
The mean is the average value of the data
items. The mean is equal to the total of
all the data items divided by the number
of data items ( 99 ). The mean value for
this run is: 681 / 99 = 6.8788


********
 Median
********
The unsorted array of responses is
 6 7 8 9 8 7 8 9 8 9 7 8 9 5 9 8 7 8 7 8
 6 7 8 9 3 9 8 7 8 7 7 8 9 8 9 8 9 7 8 9
 6 7 8 7 8 7 9 8 9 2 7 8 9 8 9 8 9 7 5 3
 5 6 7 2 5 3 9 4 6 4 7 8 9 6 8 7 8 9 7 8
 7 4 4 2 5 3 8 7 5 6 4 5 6 1 6 5 7 8 7

The sorted array is
 1 2 2 2 3 3 3 3 4 4 4 4 4 5 5 5 5 5 5 5
 5 6 6 6 6 6 6 6 6 6 7 7 7 7 7 7 7 7 7 7
 7 7 7 7 7 7 7 7 7 7 7 7 7 8 8 8 8 8 8 8
 8 8 8 8 8 8 8 8 8 8 8 8 8 8 8 8 8 8 8 8
 9 9 9 9 9 9 9 9 9 9 9 9 9 9 9 9 9 9 9 9
```

Program Output (continued)

```
The median is element 49 of
the sorted 99 element array.
For this run the median is 7


********
  Mode
********
Response  Frequency          Histogram


       1          1          *
       2          3          ***
       3          4          ****
       4          5          *****
       5          8          ********
       6          9          *********
       7         23          ***********************
       8         27          ***************************
       9         19          *******************
The mode is the most frequent value.
For this run the mode is 8 which occurred 27 times.
```

Searching Arrays: Linear Search and Binary Search

- Search an array for a *key value*
- Linear search (線性搜尋法)
  - Simple
  - Compare each element of array with key value
  - Useful for small and unsorted arrays
- Binary search (二分搜尋法)
  - For sorted arrays
  - Compares **`middle`** element with **`key`**
    - If equal, match found
    - If **`key < middle`**, looks in first half of array
    - If **`key > middle`**, looks in last half
    - Repeat
  - Very fast; at most n steps, where

    $2^n >$ number of elements
    - 30 element array takes at most 5 steps
      - $2^5 > 30$ so at most 5 steps
    - 1,000,000,000 element array takes at most 30 steps (2^{30} =1,073,741,824)
    - On the other hand, 1,000,000,000 element array searched by linear search method takes in average 500,000,000 comparisons.

Linear Search of an Array

```
1  /* Fig. 6.18: fig06_18.c
2     Linear search of an array */
3  #include <stdio.h>
4  #define SIZE 100
5
6  /* function prototype */
7  int linearSearch( const int array[], int key, int size );
8
9  /* function main begins program execution */
10 int main()
11 {
12    int a[ SIZE ]; /* create array a */
13    int x;         /* counter */
14    int searchKey; /* value to locate in a */
15    int element;   /* variable to hold location of searchKey or -1 */
16
17    /* create data */
18    for ( x = 0; x < SIZE; x++ ) {
19       a[ x ] = 2 * x;
20    } /* end for */
21
22    printf( "Enter integer search key:\n" );
23    scanf( "%d", &searchKey );
24
```

fig06_18.c (Part 1 of 3)

製造一組一百個元素{0, 2, 4, 6, . . . , 198}的陣列

由鍵盤輸入一個要搜尋的數字

```
25    /* attempt to locate searchKey in array a */
26    element = linearSearch( a, searchKey, SIZE );
27
28    /* display results */
29    if ( element != -1 ) {
30       printf( "Found value in element %d\n", element );
31    } /* end if */
32    else {
33       printf( "Value not found\n" );
34    } /* end else */
35
36    return 0; /* indicates successful termination */
37
38 } /* end main */
39
```

呼叫 linearSearch 函數尋找 searchKey

列印結果

fig06_18.c (Part 2 of 3)

```
40 /* compare key to every element of array until the location is found
41    or until the end of array is reached; return subscript of element
42    if key or -1 if key is not found */
43 int linearSearch( const int array[], int key, int size )
44 {
45    int n; /* counter */
46
47    /* loop through array */
48    for ( n = 0; n < size; ++n ) {
49
50       if ( array[ n ] == key ) {
51          return n; /* return location of key */
52       } /* end if */
53
54    } /* end for */
55
56    return -1; /* key not found */
57
58 } /* end function linearSearch */
```

```
Enter integer search key:
36
Found value in element 18


Enter integer search key:
37
Value not found
```

如果陣列中有多筆相同資料時，...

fig06_18.c (Part 3 of 3)

Binary Search of a *Sorted Array*

fig06_19.c (Part 1 of 5)

```
1  /* Fig. 6.19: fig06_19.c
2     Binary search of an array */
3  #include <stdio.h>
4  #define SIZE 15
5
6  /* function prototypes */
7  int binarySearch( const int b[], int searchKey, int low, int high );
8  void printHeader( void );
9  void printRow( const int b[], int low, int mid, int high );
10
11 /* function main begins program execution */
12 int main()
13 {
14    int a[ SIZE ]; /* create array a */
15    int i;         /* counter */
16    int key;       /* value to locate in array a */
17    int result;    /* variable to hold locati
18
19    /* create data */
20    for ( i = 0; i < SIZE; i++ ) {
21       a[ i ] = 2 * i;
22    } /* end for */
23
24    printf( "Enter a number between 0 and 28: " );
25    scanf( "%d", &key );
26
```

製造一組 { 0 , 2 , 4 , 6 , . . . , 28 } 的陣列，注意本陣列已經按大小排序，

如果要用二分法對陣列搜尋，必須要把該陣列按大小先排序後再搜尋。因此，利用二分法搜尋未排序陣列最花時間的地方就是先將該陣列排序的時間。

```
27    printHeader();
28
29    /* search for key in array a */
30    result = binarySearch( a, key, 0, SIZE - 1 );
31
32    /* display results */
33    if ( result != -1 ) {
34       printf( "\n%d found in array element %d\n", key, result );
35    } /* end if */
36    else {
37       printf( "\n%d not found\n", key );
38    } /* end else */
39
40    return 0; /* indicates successful termination */
41
42 } /* end main */
43
44 /* function to perform binary search of an array */
45 int binarySearch( const int b[], int searchKey, int low, int high )
46 {
47    int middle; /* variable to hold middle element of array */
48
```

呼叫 binarySearch 函數在陣列 a 中尋找 key 在哪裡

fig06_19.c (Part 2 of 5)

binarySearch 函數在陣列 a 中尋找 searchKey 在哪裡；

low 及 high 為陣列 a 中位置編號 的上下限。

```
49    /* loop until low subscript is greater than high subscript */
50    while ( low <= high ) {
51
52       /* determine middle element of subarray being searched */
53       middle = ( low + high ) / 2;
54
55       /* display subarray used in this loop iteration */
56       printRow( b, low, middle, high );
57
58       /* if searchKey matched middle element, return m
59       if ( searchKey == b[ middle ] ) {
60          return middle;
61       } /* end if */
62
63       /* if searchKey less than middle element, set new high */
64       else if ( searchKey < b[ middle ] ) {
65          high = middle - 1; /* search low end of array */
66       } /* end else if */
67
68       /* if searchKey greater than middle element, set new low */
69       else {
70          low = middle + 1; /* search high end of array */
71       } /* end else */
72
73    } /* end while */
74
```

fig06_19.c (Part 3 of 5)

Compares middle element with key

- If equal, match found
- If key < middle, looks in first half of array
- If key > middle, looks in last half

Repeat

```
75    return -1;   /* searchKey not found */
76
77 } /* end function binarySearch */
78
79 /* Print a header for the output */
80 void printHeader( void )
81 {
82    int i; /* counter */
83
84    printf( "\nSubscripts:\n" );
85
86    /* output column head */
87    for ( i = 0; i < SIZE; i++ ) {
88       printf( "%3d ", i );
89    } /* end for */
90
91    printf( "\n" ); /* start new line of output */
92
93    /* output line of - characters */
94    for ( i = 1; i <= 4 * SIZE; i++ ) {
95       printf( "-" );
96    } /* end for */
97
98    printf( "\n" ); /* start new line of output */
99 } /* end function printHeader */
100
```

A space

fig06_19.c (Part 4 of 5)

```
Subscripts:
  0   1   2   3   4   5   6   7   8   9  10  11  12  13  14
---------------------------------------------------------------
```

```
/* Print one row of output showing the current
   part of the array being processed. */
void printRow( const int b[], int low, int mid, int high )
{
   int i; /* counter */

   /* loop through entire array */
   for ( i = 0; i < SIZE; i++ ) {

      /* display spaces if outside current subarray range */
      if ( i < low || i > high ) {
         printf( "    " );
      } /* end if */
      else if ( i == mid ) { /* display middle element */
         printf( "%3d*", b[ i ] ); /* mark middle value */
      } /* end else if */
      else { /* display other elements in subarray */
         printf( "%3d ", b[ i ] );
      } /* end else */

   } /* end for */

   printf( "\n" ); /* start new line of output */
} /* end function printRow */
```

4 spaces

Note!

fig06_19.c (Part 5 of 5)

Program Output

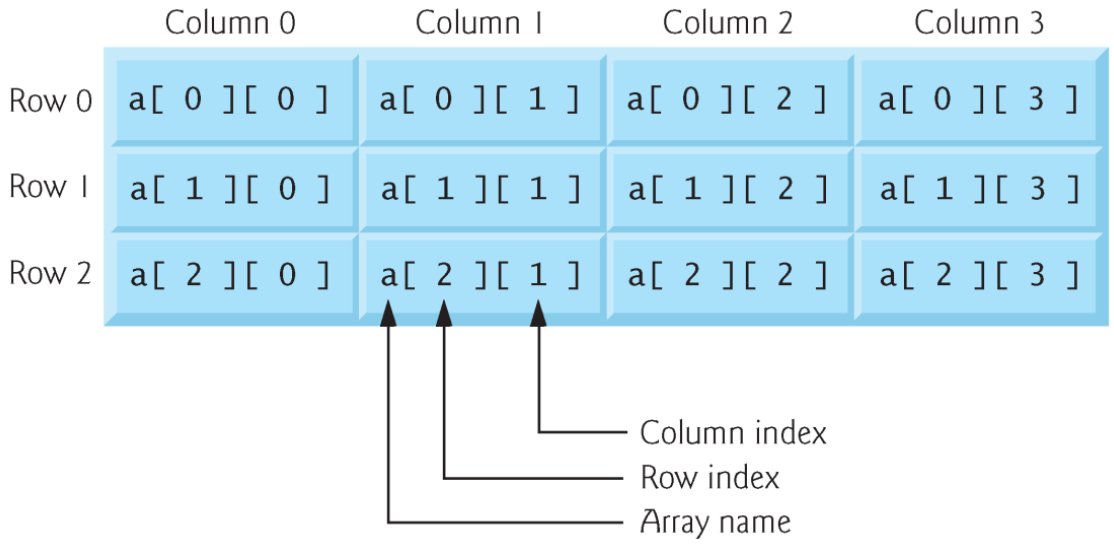
```
Enter a number between 0 and 28: 25

Subscripts:
  0   1   2   3   4   5   6   7   8   9  10  11  12  13  14
---------------------------------------------------------------
  0   2   4   6   8  10  12  14* 16  18  20  22  24  26  28
                              16  18  20  22* 24  26  28
                                          24  26* 28
                                          24*

25 not found

Enter a number between 0 and 28: 8

Subscripts:
  0   1   2   3   4   5   6   7   8   9  10  11  12  13  14
---------------------------------------------------------------
  0   2   4   6   8  10  12  14* 16  18  20  22  24  26  28
  0   2   4   6*  8  10  12
                  8  10* 12
                  8*

8 found in array element 4
```

```
Enter a number between 0 and 28: 6

Subscripts:
  0   1   2   3   4   5   6   7   8   9  10  11  12  13  14
---------------------------------------------------------------
  0   2   4   6   8  10  12  14* 16  18  20  22  24  26  28
  0   2   4   6*  8  10  12

6 found in array element 3
```

Program Output (continued)

Multiple-Subscripted Arrays

- Multiple Subscripted Arrays
  - Tables with rows (列) and columns (行)(m by n array)
  - m 列 n 行
  - Like matrices: specify row, then column
  - In a double subscripted array, **each row is basically a single-subscripted array**.



Multiple-Subscripted Arrays

- Initialization
  - `int b[ 2 ][ 2 ] = { { 1, 2 }, { 3, 4 } };`
  - Initializers grouped by row in braces
  - If not enough, *unspecified elements set to zero*

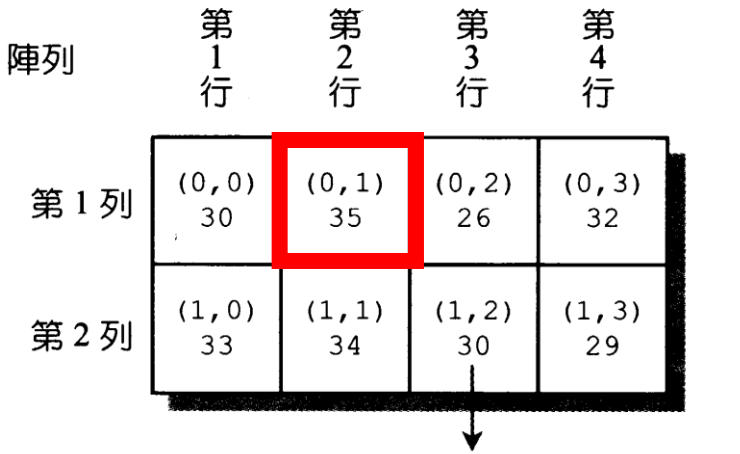
    `int b[ 2 ][ 2 ] = { { 1 }, { 3, 4 } };`

1	2
3	4

1	0
3	4

- Referencing Elements
  - Specify row (列), then column (行)

    `printf( "%d", b[ 0 ][ 1 ] );`



每一格代表一個元素，每個元素皆為 int 型態

Initializing Multi-Dimensional Arrays

```
1  /* Fig. 6.21: fig06_21.c
2     Initializing multidimensional arrays */
3  #include <stdio.h>
4
5  void printArray( const int a[][ 3 ] ); /* function prototype */
6
7  /* function main begins program execution */
8  int main()
9  {
10    /* initialize array1, array2, array3 */
11    int array1[ 2 ][ 3 ] = { { 1, 2, 3 }, { 4, 5, 6 } };
12    int array2[ 2 ][ 3 ] = { 1, 2, 3, 4, 5 };
13    int array3[ 2 ][ 3 ] = { { 1, 2 }, { 4 } };
14
15    printf( "Values in array1 by row are:\n" );
16    printArray( array1 );
17
18    printf( "Values in array2 by row are:\n" );
19    printArray( array2 );
20
21    printf( "Values in array3 by row are:\n" );
22    printArray( array3 );
23
24    return 0; /* indicates successful termination */
25
26 } /* end main */
27
```

fig06_21.c (Part 1 of 2)

$$\begin{bmatrix} 1 & 2 & 3 \\ 4 & 5 & 6 \end{bmatrix}$$

$$\begin{bmatrix} 1 & 2 & 3 \\ 4 & 5 & 0 \end{bmatrix}$$

$$\begin{bmatrix} 1 & 2 & 0 \\ 4 & 0 & 0 \end{bmatrix}$$

```
28 /* function to output array with two rows and three columns */
29 void printArray( const int a[][ 3 ] )
30 {
31    int i; /* counter */
32    int j; /* counter */
33
34    /* loop through rows */
35    for ( i = 0; i <= 1; i++ ) {
36
37       /* output column values */
38       for ( j = 0; j <= 2; j++ ) {
39          printf( "%d ", a[ i ][ j ] );
40       } /* end inner for */
41
42       printf( "\n" ); /* start new line of output */
43    } /* end outer for */
44
45 } /* end function printArray */
```

C 語言中允許二維與二維以上的多維陣列可以省略第一個 (即最左邊)的索引值，但是其他的索引值必須填寫！

列印第 i 列 (row)

列印第i 列中第 j 行 (column)

fig06_21.c (Part 2 of 2)

Program Output

```
Values in array1 by row are:
1 2 3
4 5 6
Values in array2 by row are:
1 2 3
4 5 0
Values in array3 by row are:
1 2 0
4 0 0
```

為什麼最左邊的索引值可忽略，但是其他的索引值必須填寫？

Example of Double-Subscripted Array

```
1  /* Fig. 6.22: fig06_22.c
2     Double-subscripted array example */
3  #include <stdio.h>
4  #define STUDENTS 3
5  #define EXAMS 4
6
7  /* function prototypes */
8  int minimum( const int grades[][ EXAMS ], int pupils, int tests );
9  int maximum( const int grades[][ EXAMS ], int pupils, int tests );
10 double average( const int setOfGrades[], int tests );
11 void printArray( const int grades[][ EXAMS ], int pupils, int tests );
12
13 /* function main begins program execution */
14 int main()
15 {
16    int student; /* counter */
17
18    /* initialize student grades for three students (rows) */
19    const int studentGrades[ STUDENTS ][ EXAMS ] =
20       { { 77, 68, 86, 73 },
21         { 96, 87, 89, 78 },
22         { 70, 90, 86, 81 } };
23
```

三位學生，四次考試；把他們的考試分數用陣列儲存。

宣告並給成績陣列

fig06_22.c (Part 1 of 6)

$$\begin{bmatrix} 77 & 68 & 86 & 73 \\ 96 & 87 & 89 & 78 \\ 70 & 90 & 86 & 81 \end{bmatrix}$$

$$\begin{bmatrix} \text{甲同學的分數} \\ \text{乙同學的分數} \\ \text{丙同學的分數} \end{bmatrix}$$

```
24    /* output array studentGrades */
25    printf( "The array is:\n" );
26    printArray( studentGrades, STUDENTS, EXAMS );
27
28    /* determine smallest and largest grade values */
29    printf( "\n\nLowest grade: %d\nHighest grade: %d\n",
30       minimum( studentGrades, STUDENTS, EXAMS ),
31       maximum( studentGrades, STUDENTS, EXAMS ) );
32
33    /* calculate average grade for each student */
34    for ( student = 0; student <= STUDENTS - 1; student++ ) {
35       printf( "The average grade for student %d is %.2f\n",
36          student, average( studentGrades[ student ], EXAMS ) );
37    } /* end for */
38
39    return 0; /* indicates successful termination */
40
41 } /* end main */
42
```

找所有成績的最高及最低分數

計算個別學生的平均分數

fig06_22.c (Part 2 of 6)

$$\begin{bmatrix} 77 & 68 & 86 & 73 \\ 96 & 87 & 89 & 78 \\ 70 & 90 & 86 & 81 \end{bmatrix}$$

$$\begin{bmatrix} \text{甲同學的分數} \\ \text{乙同學的分數} \\ \text{丙同學的分數} \end{bmatrix}$$

注意在呼叫 average 函數計算個別學生的平均分數時，只用 studentGrades[student] 單維陣列型態； **In a double subscripted array, each row is basically treated as a single-subscripted array.** 此時

studentGrades[0] 表示傳入 studentGrades 陣列中第 0 列第 0 行的地址到函數中；

studentGrades[1] 表示傳入 studentGrades 陣列中第 1 列第 0 行的地址到函數中；

studentGrades[2] 表示傳入 studentGrades 陣列中第 2 列第 0 行的地址到函數中；

Function minimum

```
43 /* Find the minimum grade */
44 int minimum( const int grades[][ EXAMS ], int pupils, int tests )
45 {
46    int i;              /* counter */
47    int j;              /* counter */
48    int lowGrade = 100; /* initialize to highest possible grade */
49
50    /* loop through rows of grades */
51    for ( i = 0; i < pupils; i++ ) {
52
53       /* loop through columns of grades */
54       for ( j = 0; j < tests; j++ ) {
55
56          if ( grades[ i ][ j ] < lowGrade ) {
57             lowGrade = grades[ i ][ j ];
58          } /* end if */
59
60       } /* end inner for */
61
62    } /* end outer for */
63
64    return lowGrade; /* return minimum grade */
65
66 } /* end function minimum */
67
```

fig06_22.c (Part 3 of 6)

Function maximum

```
68 /* Find the maximum grade */
69 int maximum( const int grades[][ EXAMS ], int pupils, int tests )
70 {
71    int i;             /* counter */
72    int j;             /* counter */
73    int highGrade = 0; /* initialize to lowest possible grade */
74
75    /* loop through rows of grades */
76    for ( i = 0; i < pupils; i++ ) {
77
78       /* loop through columns of grades */
79       for ( j = 0; j < tests; j++ ) {
80
81          if ( grades[ i ][ j ] > highGrade ) {
82             highGrade = grades[ i ][ j ];
83          } /* end if */
84
85       } /* end inner for */
86
87    } /* end outer for */
88
89    return highGrade; /* return maximum grade */
90
91 } /* end function maximum */
```

fig06_22.c (Part 4 of 6)

Function average

```
92
93 /* Determine the average grade for a particular student */
94 double average( const int setOfGrades[], int tests )
95 {
96    int i;          /* counter */
97    int total = 0; /* sum of test grades */
98
99    /* total all grades for one student */
100    for ( i = 0; i < tests; i++ ) {
101       total += setOfGrades[ i ];
102    } /* end for */
103
104    return ( double ) total / tests; /* average */
105
106 } /* end function average */
```

fig06_22.c (Part 5 of 6)

Function printArray

```
107
108 /* Print the array */
109 void printArray( const int grades[][ EXAMS ], int pupils, int tests )
110 {
111    int i; /* counter */
112    int j; /* counter */
113
114    /* output column heads */
115    printf( "                [0]  [1]  [2]  [3]" );
116
117    /* output grades in tabular format */
118    for ( i = 0; i < pupils; i++ ) {
119
120       /* output label for row */
121       printf( "\nstudentGrades[%d] ", i );
122
123       /* output grades for one student */
124       for ( j = 0; j < tests; j++ ) {
125          printf( "%-5d", grades[ i ][ j ] );
126       } /* end inner for */
127
128    } /* end outer for */
129
130 } /* end function printArray */
```

fig06_22.c (Part 6 of 6)

%-5d 的意思是預留5個位置顯示整數，並從左邊開始顯示。

```
The array is:
                [0]  [1]  [2]  [3]
studentGrades[0] 77   68   86   73
studentGrades[1] 96   87   89   78
studentGrades[2] 70   90   86   81
```

```
The array is:
                 [0]  [1]  [2]  [3]
studentGrades[0] 77   68   86   73
studentGrades[1] 96   87   89   78
studentGrades[2] 70   90   86   81

Lowest grade: 68
Highest grade: 96
The average grade for student 0 is 76.00
The average grade for student 1 is 87.50
The average grade for student 2 is 81.75
```

Review

- In this chapter, you have learned:
  - The array data structure.
  - The use of arrays to store, sort and search lists and tables of values.
    - Bubble Sort, Exchanger Sort (in exercise)
    - Linear Search, Binary Search
  - How to define an array, initialize an array and refer to individual elements of an array.
    - 足標 (subscript, 索引值) 從 0 開始！
    - Character Arrays (字元陣列) ；
      - `char string1[] = "first";`
      - `char string1[] = { 'f', 'i', 'r', 's', 't', '\0' };`
  - To pass arrays to functions.
    - 傳遞的是陣列的地址；Call-by-Reference！
  - The basic sorting and searching techniques.
  - To define and manipulate multiple subscript arrays.