第六章講授重點

陣列

講授重點

1. 陣列之定義與使用
2. 傳陣列給函式
3. 陣列之氣泡排序
4. 陣列之二元搜尋

1. 陣列之定義與使用

- 陣列是一群具有相同名稱以及相同型別的記憶體位置。
- 若要引用陣列的某個位置或元素，我們必須指定陣列名稱，以及此元素在陣列中的位置編號 (position number)。
- 圖6.1表示一個稱爲c的整數陣列。
- 這個陣列含有12個元素 (elements)。
- 我們可以使用陣列名稱，並且在後面接著一個放有位置編號的中括號([])，來引用任何一個元素。

1. 陣列之定義與使用(續)

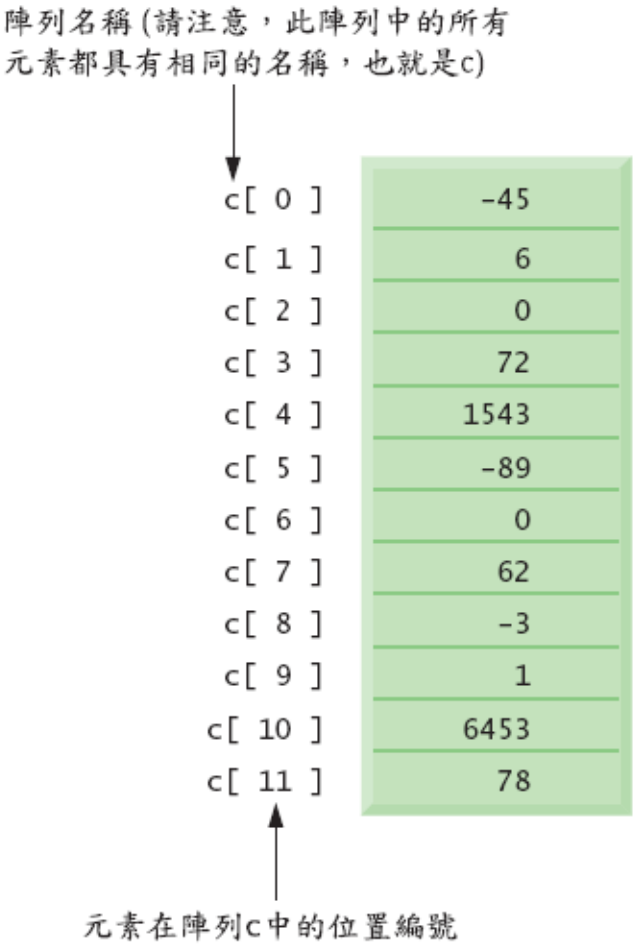


圖 6.1　含有 12 個元素的陣列

1. 陣列之定義與使用(續)

- 此陣列的名稱 (name)是 c
- 它的12個元素分別是c[0], c[1], c[2],...,c[11]。
- c[0]的**值 (value)** 是-45，c[1]的值是6，c[2]的值是0，c[7]的值是62，c[11]的值是78。
- 印出陣列c前三個元素所含之值的總和，可以撰寫以下的敘述式：
  - `printf( "%d", c[ 0 ] + c[ 1 ] + c[ 2 ] );`
- 若想將陣列c的第七個元素除以2，並將結果設給變數x，寫成 x = c[6] / 2

1. 陣列之定義與使用(續)

- 陣列c的第一個元素是c[0]，第二個元素稱是c[1]，第七個元素稱爲c[6]，因此陣列c當中的第i個元素是 `c[i - 1]`.
- 陣列名稱如同其它變數一樣，只能夠使用字母、數字和底線，
- 但是陣列名稱的第一個字母不能夠以數字開頭。

1. 陣列之定義與使用(續)

- 中括號中的位置編號正式名稱爲**下標 (subscript)** 或**索引 (index)**。
- **索引**必須是整數或是整數運算式。
- 例如，假設a=5, b=6底下的敘述式

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

- 爲陣列中的元素c[11]加上2。

1. 陣列之定義與使用(續)

- 陣列會佔用記憶體空間。
- 每個陣列所需要的元素型別和元素個數，電腦會依此來預留適當大小的記憶體空間。
- 12個元素的整數陣列c預留空間，使用以下的定義：
  - `int c[ 12 ];`

1. 陣列之定義與使用(續)

- 以下的定義
  - `int b[ 100 ], x[ 27 ];`

  爲整數陣列b預留100個元素的位置，以及爲整數陣列x預留27個元素的位置。
- 陣列亦可儲存別種資料型別。
- 字元字串與陣列之間關係，將於第八章討論。
- 指標與陣列之間關係，在第七章中討論。

例題一

```
 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( void )
 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  } /* end main */
```

圖 6.4　以初始值串列將陣列的元素初始化

例題一(續)

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

圖 6.4　以初始值串列將陣列的元素初始化

1. 陣列之定義與使用(續)

- 給定的初始值個數若小於陣列的元素個數，則剩下的元素將自動指定初始值爲零。
- 例如，圖6.3中陣列n的元素在下列的宣告後也會全部指定初始值爲零：
  - `int n[ 10 ] = { 0 };`

1. 陣列之定義與使用(續)

- 初始值陣列的定義上省略陣列的大小，則此陣列的元素個數將等於初始值串列上的元素個數。
- 例如，
  - `int n[] = { 1, 2, 3, 4, 5 };`

  將會產生一個具有5個元素的陣列。

例題二

```c
/* Fig. 6.6: fig06_06.c
   Compute the sum of the elements of the array */
#include <stdio.h>
#define SIZE 12

/* function main begins program execution */
int main( void )
{
   /* use initializer list to initialize array */
   int a[ SIZE ] = { 1, 3, 5, 4, 7, 2, 99, 16, 45, 67, 89, 45 };
   int i; /* counter */
   int total = 0; /* sum of array */

   /* sum contents of array a */
   for ( i = 0; i < SIZE; i++ ) {
      total += a[ i ];
   } /* end for */

   printf( "Total of array element values is %d\n", total );
   return 0; /* indicates successful termination */
} /* end main */
```

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

圖 6.6　計算陣列中所有元素的總和

例題三

```
 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( void )
10  {
11     int face; /* random die value 1 - 6 */
12     int roll; /* roll counter 1-6000 */
13     int frequency[ SIZE ] = { 0 }; /* clear counts */
14
15     srand( time( NULL ) ); /* seed random-number generator */
16
17     /* roll die 6000 times */
18     for ( roll = 1; roll <= 6000; roll++ ) {
19        face = 1 + rand() % 6;
20        ++frequency[ face ]; /* replaces 26-line switch of Fig. 5.8 */
21     } /* end for */
22
23     printf( "%s%17s\n", "Face", "Frequency" );
```

圖 6.9　使用陣列而不用 switch 的擲骰子程式

例題三(續)

```
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   } /* end main */
```

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

圖 6.9　使用陣列而不用 switch 的擲骰子程式

2. 傳陣列給函式

- 以傳參考的方式，來將陣列名稱傳給函式。被呼叫的函式能修改呼叫者內的原始陣列。
- 陣列名稱實際上是此陣列第一個元素的位址。
- 傳遞陣列名稱，被呼叫函式即可得知儲存這個陣列第一個元素的位置。

例題四

```
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( void )
7  {
8     char array[ 5 ]; /* define an array of size 5 */
9
10    printf( "    array = %p\n&array[0] = %p\n   &array = %p\n",
11       array, &array[ 0 ], &array );
12    return 0; /* indicates successful termination */
13 } /* end main */
```

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

圖 6.12　陣列的名稱和陣列第一個元素的位址是相同的

例題五

t0(call by refer).c

```c
// t0(call by refer).c

#include <stdio.h>

void change(int a[5]);

int main( void )
{
   int A[5] = {0,1, 2,3,4}, i;
   change(A);
   printf("A = ");
    for(i=0; i<5; i++) printf("%d ", A[i]);

   return 0; /* indicates successful termination */
} /* end main */

void change(int a[5])
{
    int i;

    printf("a = ");
    for(i=0; i<5; i++){
      printf("%d ", a[i]);
      a[i] *= 2;
    }
     printf( "\n" );
}
```

```
G:\Documents and Settings\hu\桌面\C\1\ch06\t0(call by refer).ex
a = 0 1 2 3 4
A = 0 2 4 6 8
--------------------------------
Process exited with return value 0
Press any key to continue . . .
```

作業

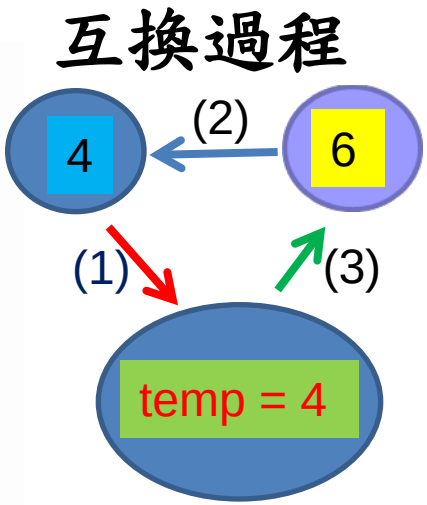
- 請重做作業5-3，請使用陣列與函式方式。
- 宣告 double X[2] ={0}
- 將X傳入函式root，第一、二根分別爲X[0]及X[1]，計算後再傳回主程式 。

3. 陣列之氣泡排序

- 氣泡排序法
- 因爲較小的數值將會如氣泡浮出水面一樣，慢慢地上升至陣列的頂點，而較大的數值則會沉到陣列的尾端。
- 在每一回合當中，將會比較相鄰的一對元素。
- 如果此對元素爲遞增順序 (或相等) 的話，則將他們維持現狀。
- 如果這對元素爲遞減順序的話，便將他們的值對調過來。

氣泡排序法演算流程

- 由小到大排序：



- 第一次掃瞄會先拿第一個元素6和第二個元素4作比較，如果第二個元素小於第一個元素，則作交換的動作。
- 接著拿6和9作比較，就這樣一直比較並交換，到第4次比較完後即可確定最大值在陣列的最後面。

第二次掃瞄:

- 第二次掃瞄亦從頭比較起，但因最後一個元素在第一次掃瞄就已確定是陣列最大值，故只需比較3次即可把剩餘陣列元素的最大值排到剩餘陣列的最後面。

第三次掃瞄：

- 第三次掃瞄完，完成三個值的排序

- 第四次掃瞄完，即可完成所有排序
- 由此可知5個元素的氣泡排序法必須執行5-1次掃瞄，第一次掃瞄需比較5-1次，共比較4+3+2+1=10次

核心程式碼

- 原理：兩兩互相比較

```
for(i=1;i<=max-1;i++)    //外迴圈數
  for(j=0;j<=max-2;j++)     //內迴圈數,兩兩比較
    if(arr[j]>arr[j+1])          //由小至大排,交換判斷
    {
      temp=arr[j];
      arr[j]=arr[j+1];
      arr[j+1]=temp;
    }
```

例題六

```
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( void )
8  {
9     /* initialize a */
10    int a[ SIZE ] = { 2, 6, 4, 8, 10, 12, 89, 68, 45, 37 };
11    int pass; /* passes counter */
12    int i; /* comparisons 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
22    /* bubble sort */
23    /* loop to control number of passes */
24    for ( pass = 1; pass < SIZE; pass++ ) {
```

圖 6.15　以氣泡排序來排序一個陣列

例題六(續)

```
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 first
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          } /* end inner for */
37       } /* end outer for */
38
39       printf( "\nData items in ascending order\n" );
40
41       /* output sorted array */
42       for ( i = 0; i < SIZE; i++ ) {
43          printf( "%4d", a[ i ] );
44       } /* end for */
45
46       printf( "\n" );
47       return 0; /* indicates successful termination */
48    } /* end main */
```

圖 6.15　以氣泡排序來排序一個陣列

例題六(續)

```
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
```

圖 6.15　以氣泡排序來排序一個陣列

例題七

t2(bubble sorting fun).c

```c
//t2.c bubble sorting function
#include <stdio.h>
#define SIZE 10

void bubble_sorting( int b[20]);

int main( void )
{
   /* initialize a */
   int a[ SIZE ] = { 2, 6, 4, 8, 10, 12, 89, 68, 45, 37 };
   int pass; /* passes counter */
   int i; /* comparisons counter */
   int hold; /* temporary location used to swap array elements */

   printf( "Data items in original order\n" );

   /* output original array */
   for ( i = 0; i < SIZE; i++ ) {
      printf( "%4d", a[ i ] );
   }

   bubble_sorting(a);

   printf( "\nData items in ascending order\n" );

   /* output sorted array */
   for ( i = 0; i < SIZE; i++ ) {
      printf( "%4d", a[ i ] );
   } /* end for */
```

例題七(續)

```
31
32      printf( "\n" );
33      return 0;
34  }
35
36  void bubble_sorting( int b[20])
37  {
38
39    int pass,i,hold;
40
41        for ( pass = 1; pass < SIZE; pass++ ) {
42
43        /* loop to control number of comparisons per pass */
44         for ( i = 0; i < SIZE - 1; i++ ) {
45
46           /* compare adjacent elements and swap them if first
47           element is greater than second element */
48           if ( b[ i ] > b[ i + 1 ] ) {
49              hold = b[ i ];
50              b[ i ] = b[ i + 1 ];
51              b[ i + 1 ] = hold;
52           }
53         }
54        }
55  }
```

```
J:\Documents and Settings\tahsiang\桌面\課\C\1\ch06\t2(bubble sorting
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

--------------------------------
Process exited with return value 0
Press any key to continue . . .
```

例題八

t5(name_height).c

```c
//t5.c 鍵入人名與身高
#include <stdio.h>

int intput_data(char N[10][5], int H[]);

int main( void )
{
   char name[10][5]={0}; /* 預定十位名字，每個名字僅有4個字元*/
   int height[10]={0}; /* 預定十個身高 */
   int i, total = 0;

   total = intput_data(name, height);

   printf( "\n" );

   for ( i = total-1; i >= 0 ; i-- ) {
      printf("%s, %d\n", name[i], height[i]);
   }

   printf( "\n" );
   return 0;
}

int intput_data(char N[10][5], int H[])
{
   int i, I = 0;

     for(i=0; i<10; i++){
       printf("Enter name and height: ");
       scanf( "%s %d", N[i], &H[i] ); /* input name and height */
       if ( H[i] <= 0 ) break;
       else I++;
       printf("\n");
   }

   return I;
```

```
J:\Documents and Settings\tahsiang\桌面\課\C
Enter name and height: a 100

Enter name and height: b 200

Enter name and height: c 300

Enter name and height: d 400

Enter name and height: e 500

Enter name and height: f 600

Enter name and height: g 700

Enter name and height: h 800

Enter name and height: i 900

Enter name and height: j 1000


j, 1000
i, 900
h, 800
g, 700
f, 600
e, 500
d, 400
c, 300
b, 200
a, 100


--------------------------------
Process exited with return value 0
Press any key to continue . . .
```

作業

- 請做作業6-2，請使用陣列與函式方式。
- 一組字串陣列與一組整數陣列，分別為 char name[10][6], int h[10]。（最多可放入10位人名，每位長度最多為5字元），整數陣列 h 放入這些人的身高。
- 由鍵盤中輸入人名與身高，一旦身高為0，則結束輸入。最多可輸入10位人名與其身高。

作業(續)

- 例如 Mr. a 180; Mr. b 167; Mr. c 175，
- 依據身高排序為 Mr. b 167; Mr. c 175; Mr. a 180。
- 請設計函式 bubble_sort2(char name[10][6], int h[10])，達到上述之要求。

4. 陣列之二元搜尋

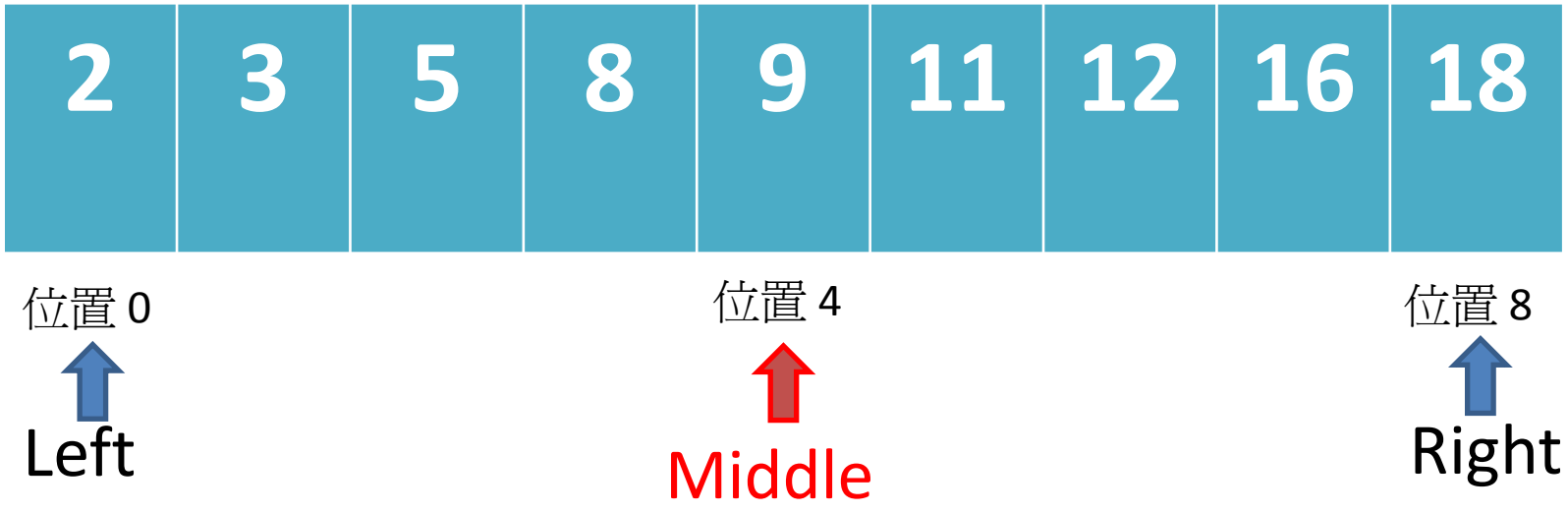
- 利用**已排序**的特性來加快搜尋速度。
- **二分搜尋法(Binary Search)**資料需依大小**先排序好**
- Middle = [(Left + Right)/2]
- 將鍵值key與**搜尋範圍**的中間資料data[Middle]作比對
  - key = data[Middle]：找到
  - key < data[Middle]：縮小**搜尋範圍** ⇒ Right = Middle-1
  - key > data[Middle]：縮小**搜尋範圍** ⇒ Left = Middle+1
- 重複上步驟，直到找到資料或搜尋範圍交叉(找不到)

範例一

考慮排序數列 2, 3, 5, 8, 9, 11, 12, 16, 18 ，搜尋值爲 11 時：

Step1：與中間值(位置4，第五個數值)9 比較, 因爲11 > 9, 所以往後半部移動.

Middle = ⌊(Left + Right)/2⌋= ⌊(0 + 8)/2⌋=4

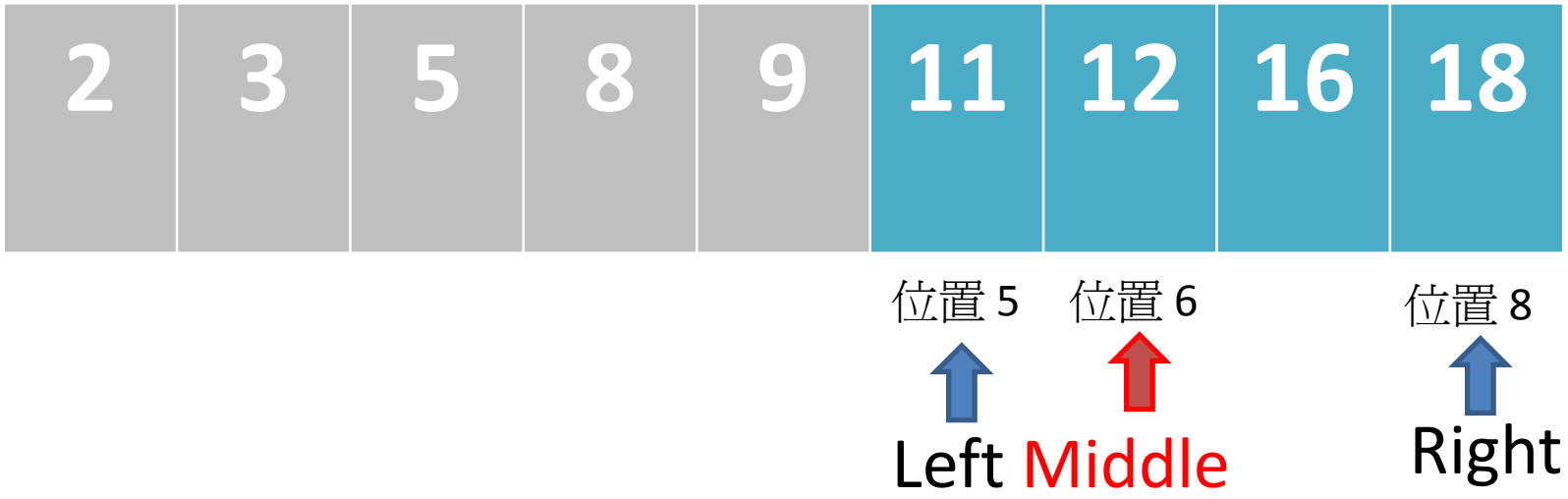


範例一(續)

考慮排序數列 2, 3, 5, 8, 9, 11, 12, 16, 18 ，搜尋值爲 11 時：

Step2 : 接著選擇後半段中間值 (位置6，第七個數值)12 比較, 因爲11 < 12, 所以往前半部移動。

Middle = ⌊(Left + Right)/2⌋= ⌊(5 + 8)/2⌋=6

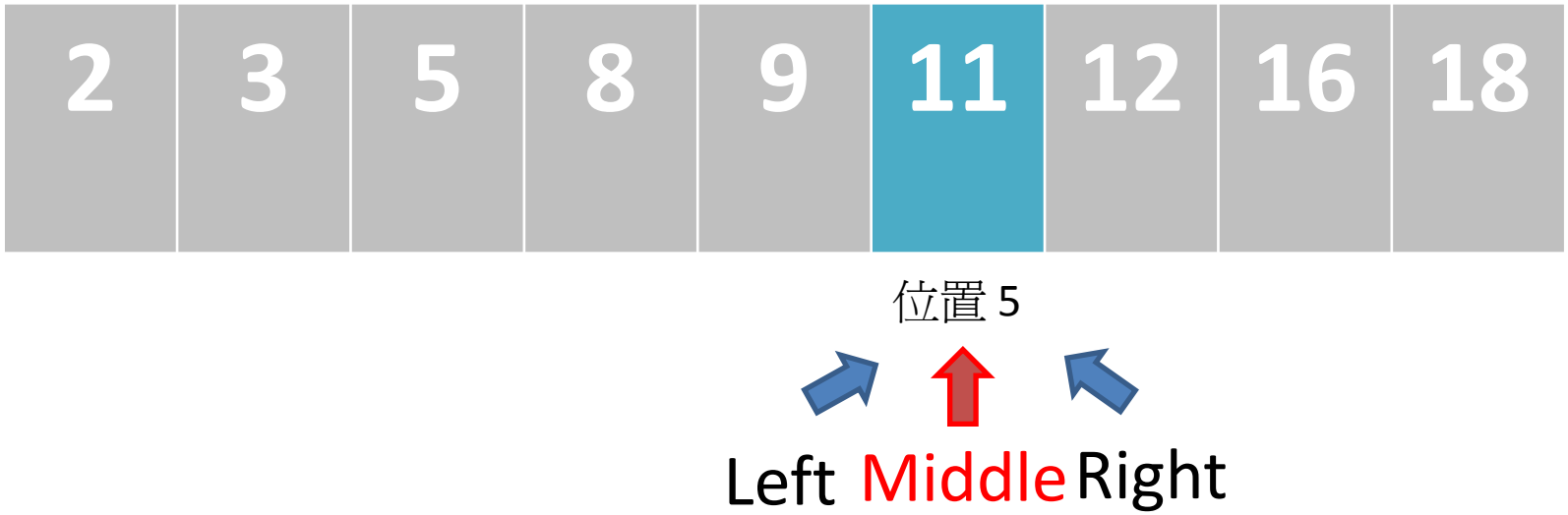


範例一(續)

考慮排序數列 2, 3, 5, 8, 9, 11, 12, 16, 18 ，搜尋值爲 11 時：

Step3 : 因爲前半部只剩下(位置5，第六個數值) 11=11, 表示搜尋完成。

Middle = ⌊(Left + Right)/2⌋= ⌊(5 + 5)/2⌋=5



例題八

```
1  /* Fig. 6.19: fig06_19.c
2     Binary search of a sorted 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( void )
13 {
14    int a[ SIZE ]; /* create array a */
15    int i; /* counter for initializing elements 0-14 of array a */
16    int key; /* value to locate in array a */
17    int result; /* variable to hold location of key or -1 */
18
19    /* create data */
20    for ( i = 0; i < SIZE; i++ ) {
21       a[ i ] = 2 * i;
22    } /* end for */
23
```

圖 6.19　在已排序的陣列中進行二元搜尋

例題八(續)

```
24      printf( "Enter a number between 0 and 28: " );
25      scanf( "%d", &key );
26
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   } /* end main */
42
```

圖 6.19　在已排序的陣列中進行二元搜尋

例題八(續)

```
43  /* function to perform binary search of an array */
44  int binarySearch( const int b[], int searchKey, int low, int high )
45  {
46     int middle; /* variable to hold middle element of array */
47
48     /* loop until low subscript is greater than high subscript */
49     while ( low <= high ) {
50
51        /* determine middle element of subarray being searched */
52        middle = ( low + high ) / 2;
53
54        /* display subarray used in this loop iteration */
55        printRow( b, low, middle, high );
56
57        /* if searchKey matched middle element, return middle */
58        if ( searchKey == b[ middle ] ) {
59           return middle;
60        } /* end if */
61
62        /* if searchKey less than middle element, set new high */
63        else if ( searchKey < b[ middle ] ) {
64           high = middle - 1; /* search low end of array */
65        } /* end else if */
66
```

圖 6.19　在已排序的陣列中進行二元搜尋

例題八(續)

```
67          /* if searchKey greater than middle element, set new low */
68          else {
69             low = middle + 1; /* search high end of array */
70          } /* end else */
71       } /* end while */
72
73       return -1; /* searchKey not found */
74    } /* end function binarySearch */
75
76    /* Print a header for the output */
77    void printHeader( void )
78    {
79       int i; /* counter */
80
81       printf( "\nSubscripts:\n" );
82
83       /* output column head */
84       for ( i = 0; i < SIZE; i++ ) {
85          printf( "%3d ", i );
86       } /* end for */
87
88       printf( "\n" ); /* start new line of output */
89
```

圖 6.19　在已排序的陣列中進行二元搜尋

例題八(續)

```
90       /* output line of - characters */
91       for ( i = 1; i <= 4 * SIZE; i++ ) {
92          printf( "-" );
93       } /* end for */
94
95       printf( "\n" ); /* start new line of output */
96    } /* end function printHeader */
97
98    /* Print one row of output showing the current
99       part of the array being processed. */
100   void printRow( const int b[], int low, int mid, int high )
101   {
102      int i; /* counter for iterating through array b */
103
104      /* loop through entire array */
105      for ( i = 0; i < SIZE; i++ ) {
106
107         /* display spaces if outside current subarray range */
108         if ( i < low || i > high ) {
109            printf( "    " );
110         } /* end if */
111         else if ( i == mid ) { /* display middle element */
112            printf( "%3d*", b[ i ] ); /* mark middle value */
113         } /* end else if */
```

圖 6.19　在已排序的陣列中進行二元搜尋

例題八(續)

```
114         else { /* display other elements in subarray */
115            printf( "%3d ", b[ i ] );
116         } /* end else */
117      } /* end for */
118
119      printf( "\n" ); /* start new line of output */
120   } /* end function printRow */
```

```
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
```

圖 6.19　在已排序的陣列中進行二元搜尋

例題八(續)

```
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
```

圖 6.19　在已排序的陣列中進行二元搜尋

作業

- 請利作業6-2之程式，製作二元搜尋函式(老胡小鋪, aries.dyu.edu.tw/~thhu ，作業6-4)。

1. 先輸入姓名與身高，例如

   a 180

   b 167

   c 175

   d 190

   e 150

   f 130

   0 0。當身高爲0時，程式停止姓名與身高之輸入工作。

作業(續)

2. 依據身高排序爲

   f 130

   e 150

   b 167

   c 175

   a 180

   d 190

3. 當輸入身高爲 175，則印出“找到, c 175”。

4. 當輸入身高爲 170，則印出“找不到”。