

程式設計 (**Programming**)

真理大學 資訊工程系 吳汶涓老師

CH06 陣列(array)



本章綱要

6-1 簡介

6-2 陣列

6-3 定義陣列

6-4 使用陣列的例子

6-5 傳遞陣列給函式

6-6 陣列的排序

6-7 範例研究：使用陣列來計算平均數、眾數等

6-8 搜尋陣列

6-9 多維陣列

6-10 可變長度陣列

6-11 安全程式設計

6.6 陣列的排序


■ 排序資料 (Sorting)

- 電腦最重要的應用之一，幾乎每一個組織單位都必須排序資料

■ 氣泡排序 (bubble sort、sinking sort)

- 會對陣列處理數個回合
- 每回合需比較相鄰一對元素的順序
- 重複

■ 範例

- 原始陣列: [3, 4, 2, 6, 7]

- 第一回合: [3, 2, 4, 6, 7]
- 第二回合: [2, 3, 4, 6, 7]
- 較小的元素會如氣泡浮出水面一樣，慢慢地上升至陣列的頂點

```

1  /* Fig. 6.15: fig06_15.c */
3  #include <stdio.h>
4  #define SIZE 10
7  int main( void )
8  {
10     int a[ SIZE ] = { 2, 6, 4, 8, 10, 12, 89, 68, 45, 37 };
11     int pass;
12     int i;
13     int hold; /* temporary to swap array elements */
15     printf( "Data items in original order\n" );
18     for ( i = 0; i < SIZE; i++ ) {
19         printf( "%4d", a[ i ] );
20     }
22     /* bubble sort */
24     for ( pass = 1; pass < SIZE; pass++ ) {
27         for ( i = 0; i < SIZE - 1; i++ ) {
31             if ( a[ i ] > a[ i + 1 ] ) {
32                 hold = a[ i ];
33                 a[ i ] = a[ i + 1 ];
34                 a[ i + 1 ] = hold;
35             }
36         }
37     }
39     printf( "\nData items in ascending order\n" );
40
42     for ( i = 0; i < SIZE; i++ ) {
43         printf( "%4d", a[ i ] );
44     }
46     printf( "\n" );
47     return 0;
48 }

```

若陣列元素的順序顛倒，
此動作會將它們交換

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

課本pp. 6-25

■ 分析

- 氣泡排序容易撰寫、簡單
- 但是，執行速度相當慢
 - 一回合中，小的值只前進一個位置，但最大值保證會移到陣列最底部

■ 如何修改氣泡排序法，使得排序更有效率？

- pp. 6-54, 習題6.11
- 一回合後，最大值會定位好，所以下個回合應不用作 n 次的比較
- 其他課程 (大二的資料結構、演算法)

6.7 範例研究

- 使用陣列來計算平均值、中位數以及眾數
 - **Mean**：平均數
 - **Median**：中位數，已排序串列的中間元素
 - 1, 2, 3, 4, 5 串列
 - 3 是中位數
 - **Mode**：眾數，出現頻率最高的數值
 - 1, 1, 1, 2, 3, 3, 4, 5 串列
 - 1 是眾數
 - 範例：使用陣列來儲存99個的調查資料，資料內容都是1~9之間的數字，程式必須計算出99個數值的平均數、中位數和眾數。

```

4  #include <stdio.h>
5  #define SIZE 99 ← 定義常數
6
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
15 int main( void )
16 {
17     int frequency[ 10 ] = { 0 }; ← 陣列初始化
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
33     mean( response );
34     median( response );
35     mode( frequency, response ); ← 需傳送紀錄頻率的陣列
36     return 0;
37 }

```

課本pp. 6-27

```

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

```

陣列的接收

強制轉型

```

*****
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

```



```

61 void median( int answer[] )
62 {
63     printf( "\n%s\n%s\n%s\n%s",
64             "*****", " Median", "*****",
65             "The unsorted array of responses is" );
66
67     printArray( answer );
69     bubbleSort( answer );
71     printf( "\n\nThe sorted array is" );
72     printArray( answer );
73
75     printf( "\n\nThe median is element %d of\n"
76             "the sorted %d element array.\n"
77             "For this run the median is %d\n\n",
78             SIZE / 2, SIZE, answer[ SIZE / 2 ] );
79 }

```

排序完成，中位數
是中間的元素

```

155 /* output array contents */
156 void printArray( const int a[] )
157 {
158     int j;
161     for ( j = 0; j < SIZE; j++ ) {
163         if ( j % 20 == 0 ) {
164             printf( "\n" );
165         }
167         printf( "%2d", a[ j ] );
168     }
169 }

```

```

*****
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

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

```

```

133 void bubbleSort( int a[] )
134 {
135     int pass;
136     int j;
137     int hold;
138
140     for ( pass = 1; pass < SIZE; pass++ ) {
143         for ( j = 0; j < SIZE - 1; j++ ) {
146             if ( a[ j ] > a[ j + 1 ] ) {
147                 hold = a[ j ];
148                 a[ j ] = a[ j + 1 ];
149                 a[ j + 1 ] = hold;
150             }
151         }
152     }
153 }

```

```

82 void mode( int freq[], const int answer[] )
83 {
84     int rating;
85     int j;
86     int h;
87     int largest = 0;
88     int modeValue = 0;
89
90     printf( "\n%s\n%s\n%s\n",
91             "*****", " Mode", "*****" );
94     for ( rating = 1; rating <= 9; rating++ ) {
95         freq[ rating ] = 0;
96     }
99     for ( j = 0; j < SIZE; j++ ) {
100         ++freq[ answer[ j ] ]; /*summarize frequencies*/
101     }
102
104     printf( "%s%11s%19s\n\n%54s\n%54s\n\n",
105             "Response", "Frequency", "Histogram",
106             "1 1 2 2", "5 0 5 0 5" );
107
109     for ( rating = 1; rating <= 9; rating++ ) {
110         printf( "%8d%11d", rating, freq[ rating ] );
113         if ( freq[ rating ] > largest ) {
114             largest = freq[ rating ];
115             modeValue = rating;
116         }
119         for ( h = 1; h <= freq[ rating ]; h++ ) {
120             printf( "*" );
121         }
123         printf( "\n" );
124     }
126
127     printf( "The mode is the most frequent value.\n"
128            "For this run the mode is %d which occurred"
129            " %d times.\n", modeValue, largest );
130 }

```

陣列初始化

```

*****
Mode
*****
Response  Frequency      Histogram
                    5      1      1      2      2
                    5      0      5      0      5

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.

```

練習

- 請修改上面範例的程式，達到以下要求：
 - **Response**的內容改為用**rand**產生100個1~9的數字
 - 修改**mode**函式使具有處理數個相同**mode**值的能力
 - 修改**median**函式，當陣列有偶數個元素時，中間值為兩個中間值的平均
 - 修改氣泡排序法，讓排序更有效率



6.8 搜尋陣列

- 判斷陣列中是否含有某個關鍵值
 - 關鍵值(**key**)可由使用者輸入或自行設定
- 線性搜尋 (**linear search**)
 - 簡單
 - 將陣列中的每一個元素與關鍵值互相比較
 - 對於小型的陣列或未排序過的陣列來說，線性搜尋是很有用的
 - Example: [2, 3, 5, 1, 2, 6] 陣列中是否存在4的值？
一個一個找，總共找6次

```

3  #include <stdio.h>
4  #define SIZE 100
7  int linearSearch( const int array[], int key, int size );
8
10 int main( void )
11 {
12     int a[ SIZE ];
13     int x;
14     int searchKey;
15     int element;
18     for ( x = 0; x < SIZE; x++ ) {
19         a[ x ] = 2 * x;
20     }
21
22     printf( "Enter integer search key:\n" );
23     scanf( "%d", &searchKey );
26     element = linearSearch( a, searchKey, SIZE );
27
29     if ( element != -1 ) {
30         printf( "Found value in element %d\n", element );
31     }
32     else {
33         printf( "Value not found\n" );
34     }
36     return 0;
37 }
42 int linearSearch( const int array[], int key, int size )
43 {
44     int n;
47     for ( n = 0; n < size; ++n ) {
49         if ( array[ n ] == key ) {
50             return n;
51         }
52     }
54     return -1;
55 }

```

create data

Enter integer search key:
36
Found value in element 18

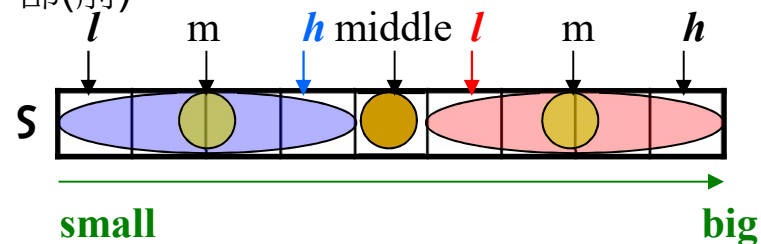
Enter integer search key:
37
Value not found

線性搜尋會搜尋每個元素，直到找到符合為止

課本pp. 6-32

■ 二元搜尋 (binary search)

- 只能用在已經排序好的陣列
- 找出陣列的中間元素，將它(m)與搜尋關鍵值(k)作比較
 - $m=k$ ：表示已找到要找的元素
 - $m < k$ ：搜尋陣列的右半部(後)
 - $m > k$ ：搜尋陣列的左半部(前)
 - 重複

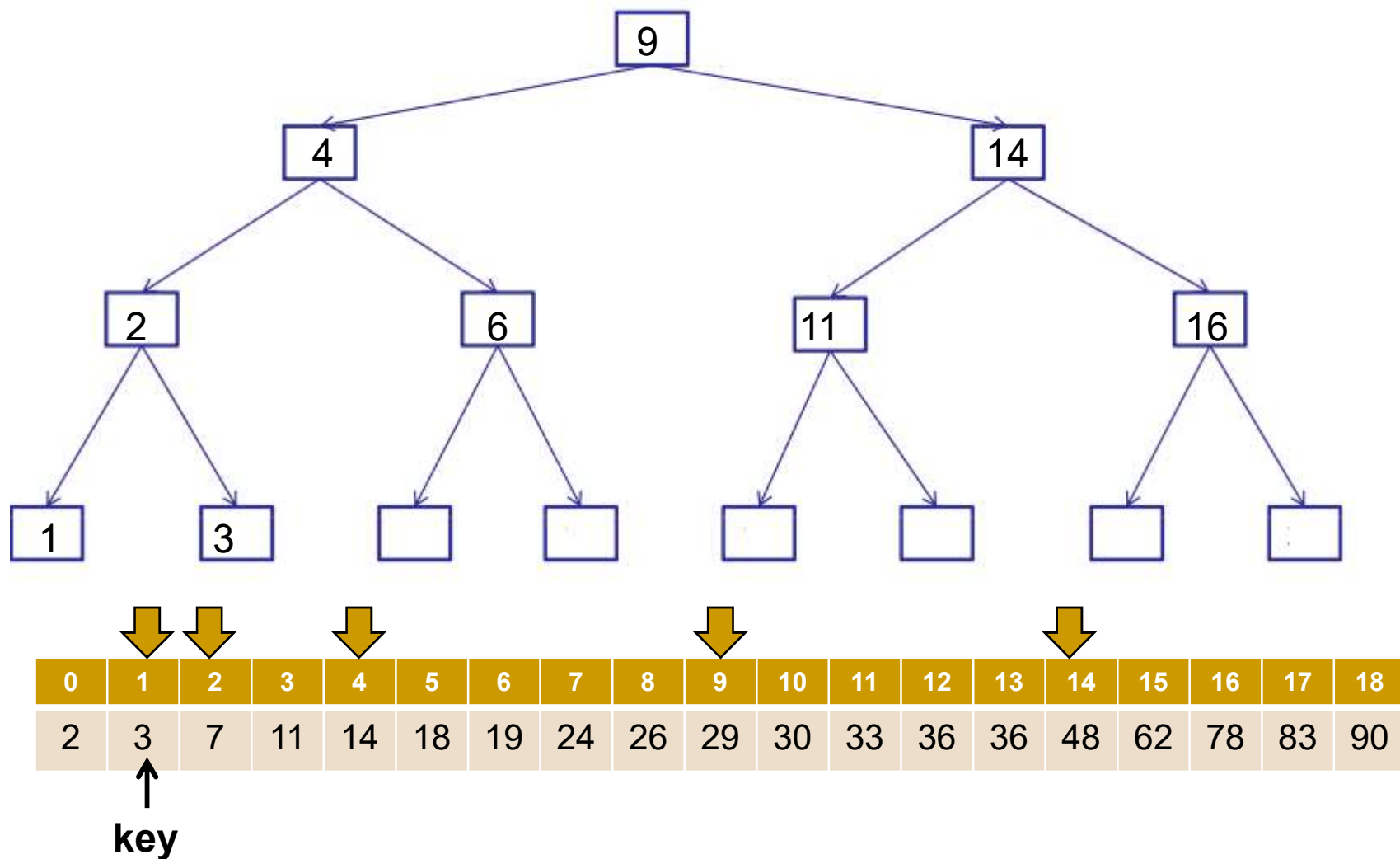


$$\text{middle} = \left\lfloor \frac{l + h}{2} \right\rfloor$$

- 非常快速：最多 t 個步驟，其中 2^t 大於元素數量

- 30個元素的陣列，至多五個步驟

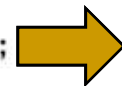
$2^5 > 30$, 因此最多五個步驟；也可以計算 $\lceil \log_2 30 \rceil$, 因為 $2^4 < \log_2 30 < 2^5$



```

1  /* Fig. 6.19: fig06_19.c */
3  #include <stdio.h>
4  #define SIZE 15
5
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
12 int main( void )
13 {
14     int a[ SIZE ];
15     int i;
16     int key;
17     int result;
18
20     for ( i = 0; i < SIZE; i++ ) {
21         a[ i ] = 2 * i;
22     }
24     printf( "Enter a number between 0 and 28: " );
25     scanf( "%d", &key );
26
27     printHeader();
30     result = binarySearch( a, key, 0, SIZE - 1 );
33     if ( result != -1 ) {
34         printf( "\n%d found in array element %d\n", key, result );
35     }
36     else {
37         printf( "\n%d not found\n", key );
38     }
40     return 0;
41 }
42

```



Enter a number between 0 and 28: 25

課本pp. 6-35


```

44 int binarySearch( const int b[], int searchKey, int low, int high )
45 {
46     int middle;
49     while ( low <= high ) {
52         middle = ( low + high ) / 2;
55         printRow( b, low, middle, high );
58         if ( searchKey == b[ middle ] ) {
59             return middle; ← found, 回傳索引位置
60         }
63         else if ( searchKey < b[ middle ] ) {
64             high = middle - 1; ← 若值太小, 尋找陣列左半部
65         }
68         else {
69             low = middle + 1; ← 若值太大, 尋找陣列右半部
70         }
71     }
73     return -1;
74 }

77 void printHeader( void )
78 {
79     int i;
81     printf( "\nSubscripts:\n" );
84     for ( i = 0; i < SIZE; i++ ) {
85         printf( "%3d ", i );
86     }
87
88     printf( "\n" );
91     for ( i = 1; i <= 4 * SIZE; i++ ) {
92         printf( "-" );
93     }
95     printf( "\n" );
96 }

```

```

100 void printRow( const int b[], int low, int mid, int high )
101 {
102     int i;
105     for ( i = 0; i < SIZE; i++ ) {
108         if ( i < low || i > high ) {
109             printf( "    " );
110         }
111         else if ( i == mid ) {
112             printf( "%3d*", b[ i ] );
113         }
114         else {
115             printf( "%3d ", b[ i ] );
116         }
117     }
119     printf( "\n" );
120 }

```

printHeader()函式印出的
 呼叫printRow()函式印出的

```

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

```

練習

- 修改課本圖6.19的C語言程式，改使用遞迴函式 **binarySearch** 來對陣列執行二元搜尋。此函式的參數為一個整數陣列以及陣列的起始下標和終止下標，若找到搜尋鍵，則回傳陣列的下標，否則會傳-1。

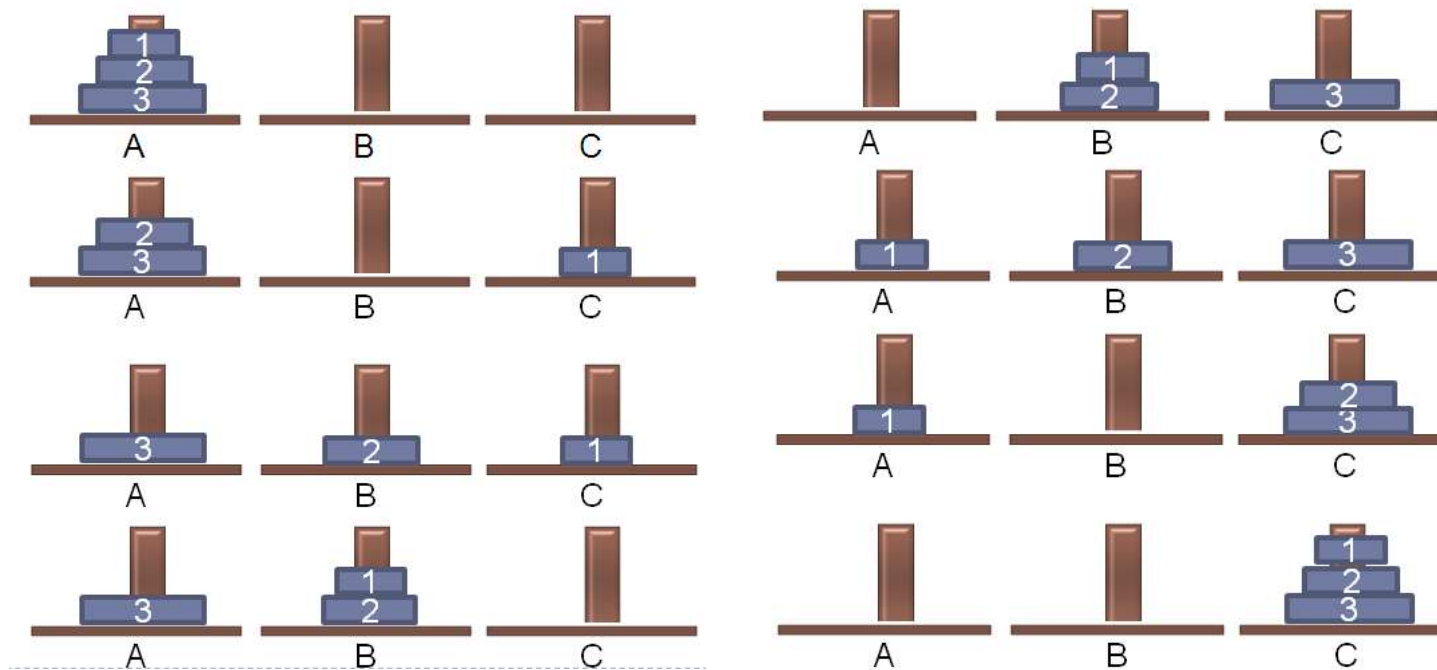


練習函式

- 計算斜邊長 (課本 pp5-54, ex5.15)
 - 自訂一個函式 `hyoptenuse()`，在已知直角三角形的兩邊長後，計算出斜邊長度。
- 河內塔 (Hanoi Tower)
 - 撰寫一程式，將 n 個由大到小疊在一根柱子上的碟子，依序搬到另一根柱子中，搬移過程須遵照規則並可利用另一柱子。
 - 一次移動一個碟子
 - 小碟子壓大碟子

河內塔(Hanoi Tower)

- 河內塔問題：在A、B、C三個塔上有數個圓盤，請求出將圓盤從A塔全數搬移到C塔的順序，且大的圓盤不可以疊到小圓盤上。



練習函式

■ 河內塔 (Hanoi Tower)

□ 撰寫一程式，將 n 個由大到小疊在一根柱子上的碟子，依序搬到另一根柱子中，搬移過程須遵照規則並可利用另一柱子。

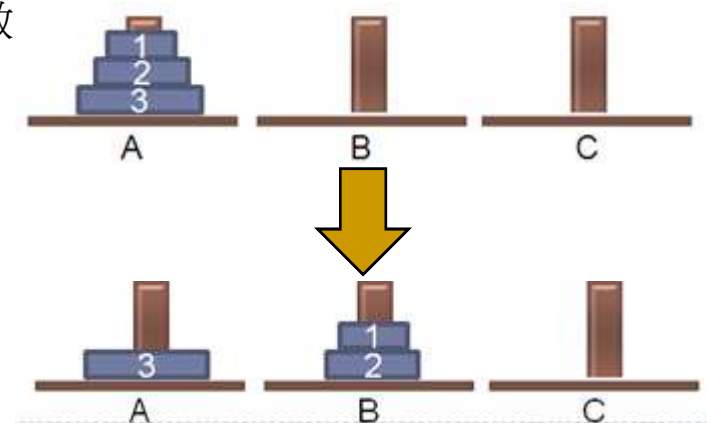
- 一次移動一個碟子
- 小碟子壓大碟子

■ 宣告函式 `hanoi(n, S, M, T)`，共有4個參數

碟子個數

起點 中介 終點

呼叫函式 `hanoi(n-1, A, C, B);`
`move A to C;`
`hanoi(n-1, B, A, C);`



練習陣列

- 利用陣列來儲存輸入的字串並達到以下功能
 - 計算字串長度
 - 將字串顛倒
- 猜數字或猜字母 (課本 pp5-60, ex5.32)
 - 程式從1~100中挑選出一個亂數當重點數
 - 由使用者進行猜測，不斷地將區間縮小
 - 利用二元搜尋

練習陣列

- 若一個四位數中恰好有兩個數字相同，稱為好數。
 - 請判斷某四位數是否為好數。
- 1223, 3464, 9001 是好數
- 1333, 5535, 1234 不是好數
- 四位數由亂數產生10個
- 每個亂數進行數字拆解分別存到陣列作統計
- 若剛好有兩個就是好數