資料結構與演算法實作

3 資料結構與演算法實作

- 二元搜尋法，BINARY search with FLAMENCO dance
  https://youtu.be/iP897Z5Nerk

- 氣泡排序法
  Bubble-sort with Hungarian ("Csángó") folk dance
  https://youtu.be/lyZQPjUT5B4

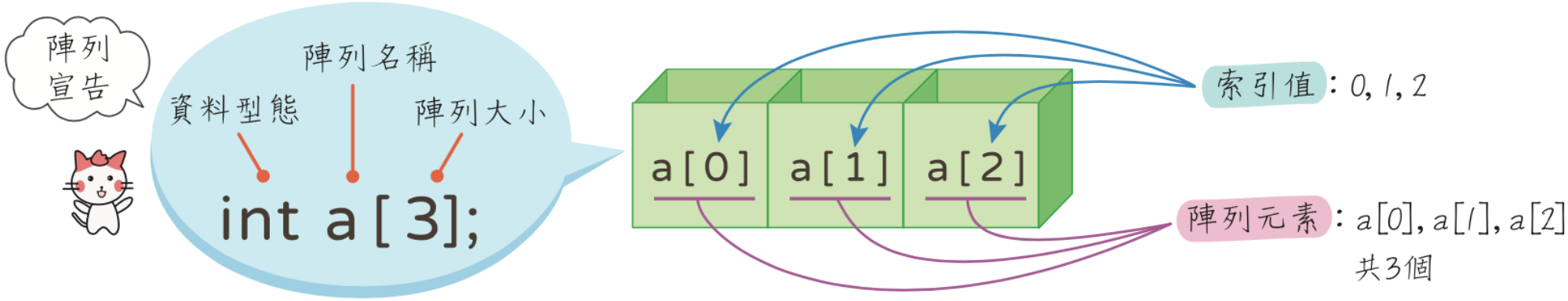
- [資訊之芽 算法班]　01. 複雜度分析
  https://youtu.be/_r7cfVrn28c

3-1 陣列資料結構實作

3-1-1 陣列的表示法

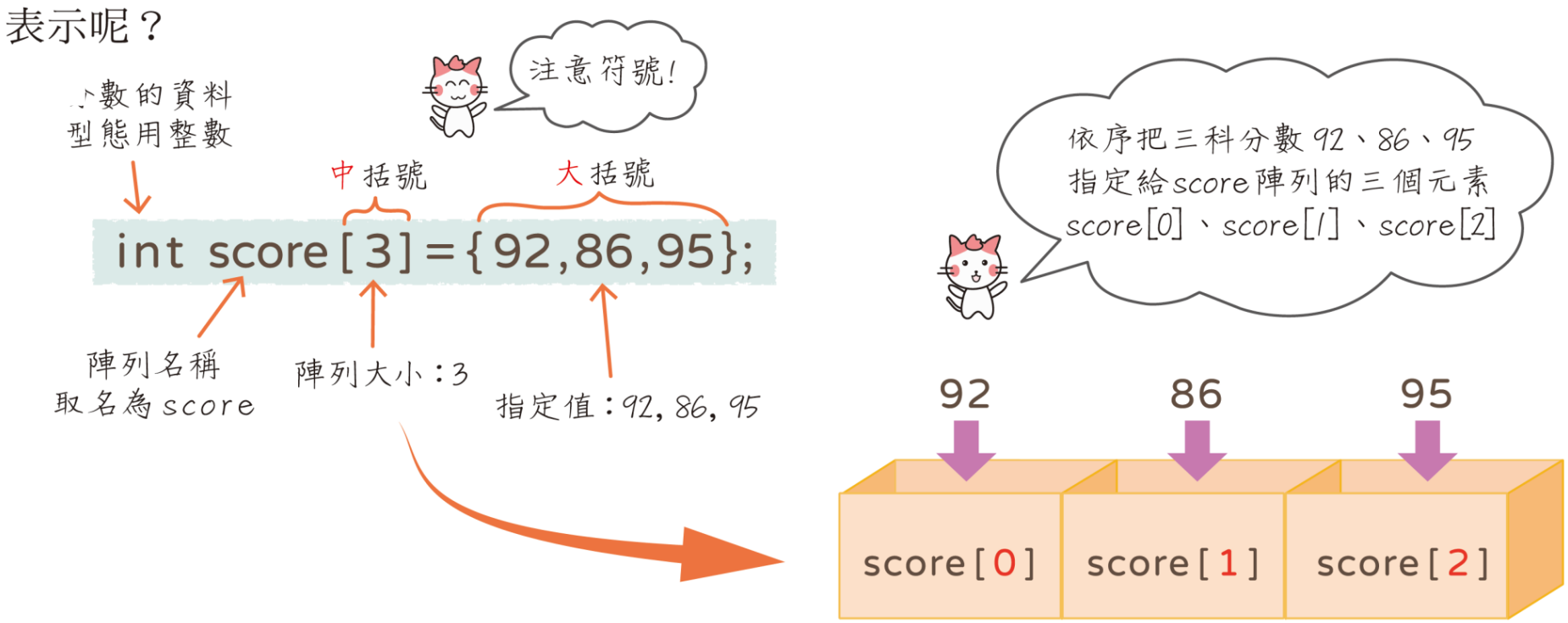
一、陣列的宣告

在程式中要使用陣列，和變數一樣，也必須要先做宣告的動作。讓我們來看看如何宣告喔！



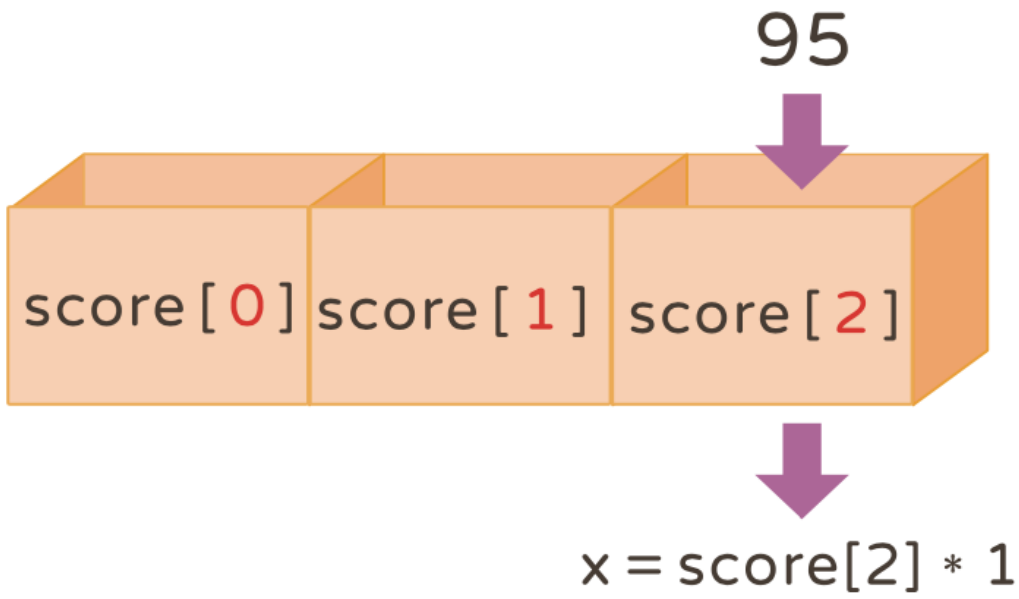
二、陣列指定值

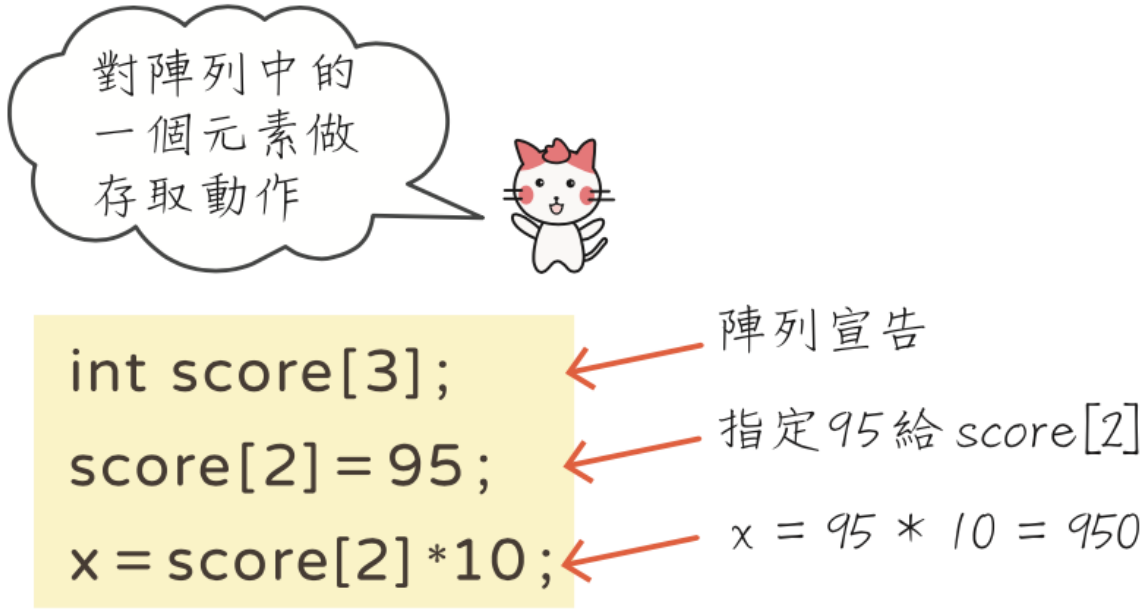
學會陣列宣告後，要如何把資料存到陣列元素中呢？（也就是指定值給陣列元素）讓我們繼續看看下面這個例子：
已知國、英、數三個科目的成績，分數分別是92、86、95，如何將成績以陣列來表示呢？



三、陣列的使用

除了一次指定值給整個陣列外，也可以直接指定值給陣列中的某一個元素，也就是，可以把陣列元素看成一個變數，一樣可以參與運算。



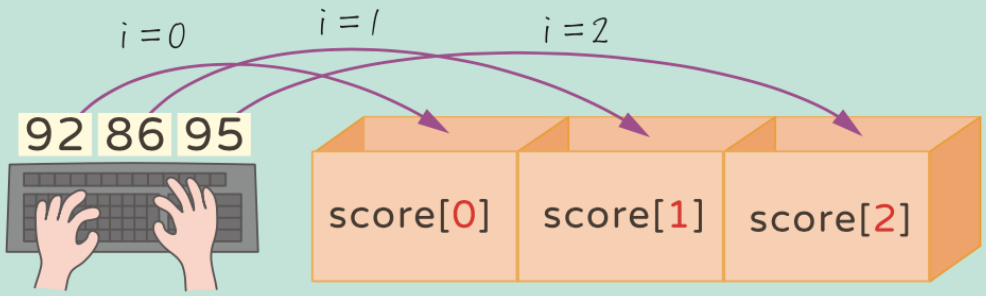


三、陣列的使用

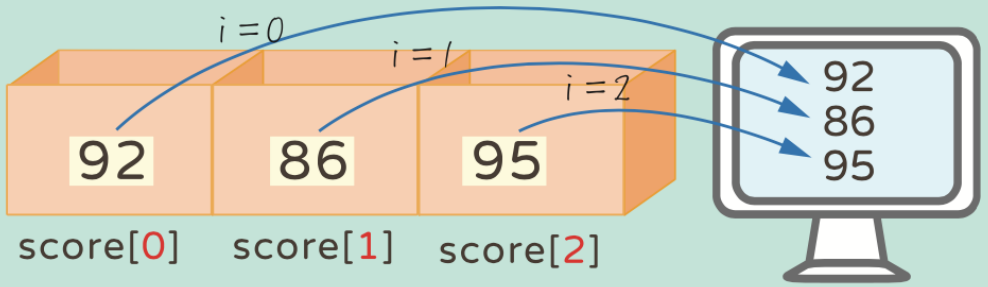
若要對整個陣列進行存取時，通常可搭配for迴圈來使用：


```
int score[3];
for (i=0; i<3; i++)
    cin >> score[i];
```

利用for迴圈，依序將鍵盤輸入的資料(92、86、95)指定給score[0]、score[1]、score[2]




```
int score[3]={92,86,95};
for (i=0; i<3; i++)
    cout << score[i] << endl;
```



利用for迴圈，依序將陣列三個元素score[0]、score[1]、score[2]的值輸出(顯示)到螢幕上

實力打卡

（ 解 ） 1. 下列 C++ 程式語言的陣列表示法，何者正確？

(A) a{2}　(B) a(2)　(C) a[2]　(D) a<2> 。

（ 解 ） 2. 在 C++ 程式碼中宣告 int a[3]={1,2,3};，a[2] 的值為何？

(A) 0　(B) 1　(C) 2　(D) 3 。

（ 解 ） 3. 在 C++ 程式碼中宣告 a[10]，陣列 a 共有多少元素？

(A) 10　(B) 0　(C) 9　(D) 11 。

任務

校運會時，Bob的田徑三項的成績（100公尺、跳高、擲鉛球）分別為90, 86, 92。請寫一程式，將成績以陣列存放，再以迴圈計算總分，最後算出平均分數。

100公尺	跳高	擲鉛球
90	86	92

解析

仿照第一章計算加總的方法，把其中原本是每次加變數i到sum中的程式片段，改為每次加入陣列中的一個值score[i] 到sum中，即：

```
sum=0;
for (i=1; i<=n; i++)
    sum = sum + i;
```

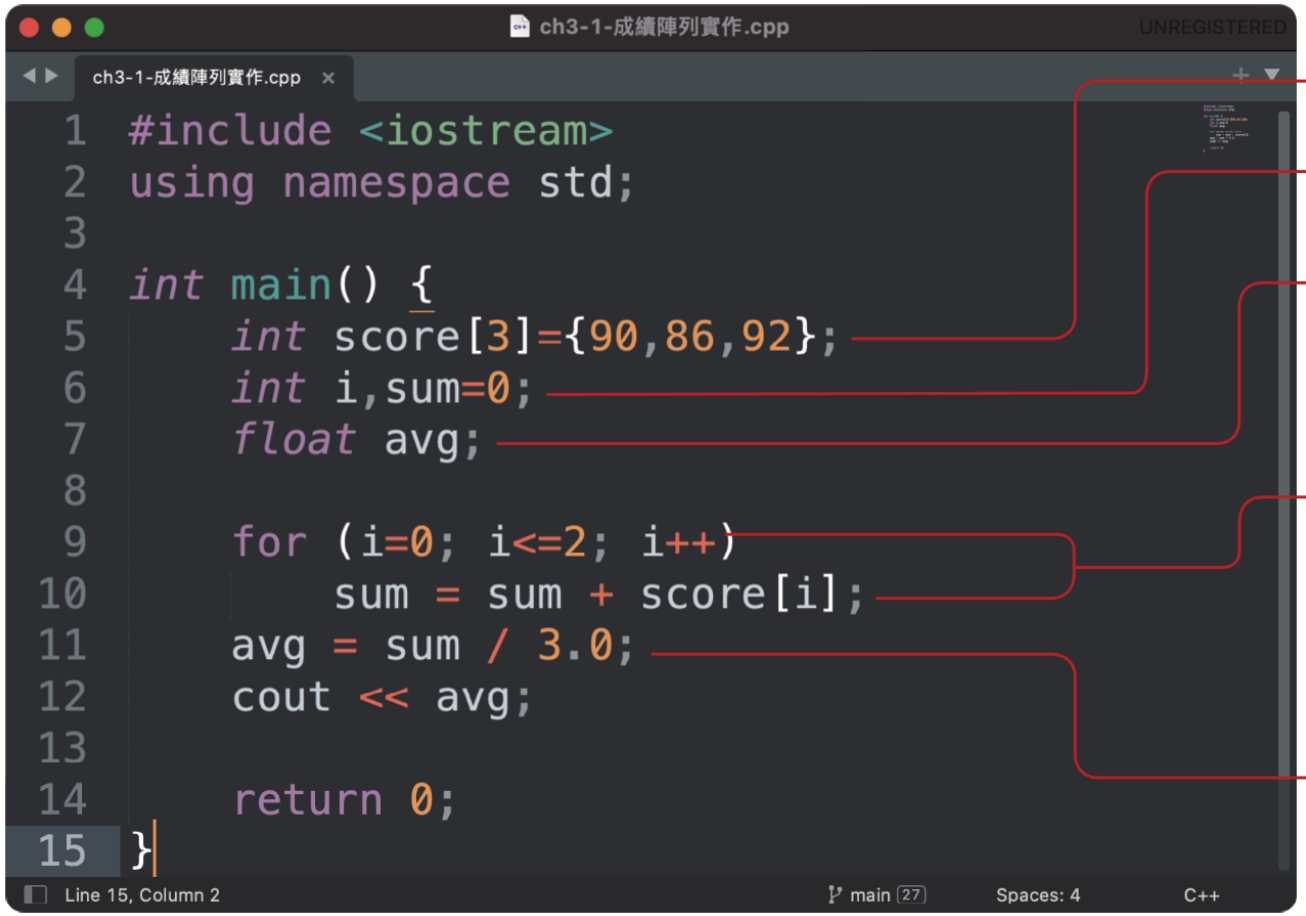
→

```
sum=0;
for (i=0;  i<=2; i++)
    sum = sum + score[i];
```

實例演練 田徑三項成績處理

實作

ch3-1-成績陣列實作.cpp



宣告陣列並指定初始值

宣告總和初始值為 0

宣告平均值為浮點數（可能帶有小數）

迴圈的 i 分別為 0,1,2，所以分別加 score[0], score[1], score[2] 到 sum 中

整數 sum 如果除以整數 3，avg 將會得到整數的商，所以改除以 3.0

執行結果

89.3333

3-1-2 二維陣列實作與應用

之前的例子，我們只考慮了一位同學國、英、數三個科目的成績，那現在假設要表示五位同學的成績呢？

「int a[3], b[3], c[3], d[3], e[3];」

「有沒有比較好的方法？全班40位同學，英文字母不夠啦！」

「我們可以改用二維的陣列喔！就像多排的置物櫃，每位同學可使用由左到右連續3 個格子，例如5位同學的話，可以改寫為

int s[5][3];」

「這樣就算是100位同學也沒問題！」

5個一維陣列

```
int a[3], b[3], c[3], d[3], e[3];
```

a[0]	a[1]	a[2]
b[0]	b[1]	b[2]
c[0]	c[1]	c[2]
d[0]	d[1]	d[2]
e[0]	e[1]	e[2]

1個二維陣列(5×3)

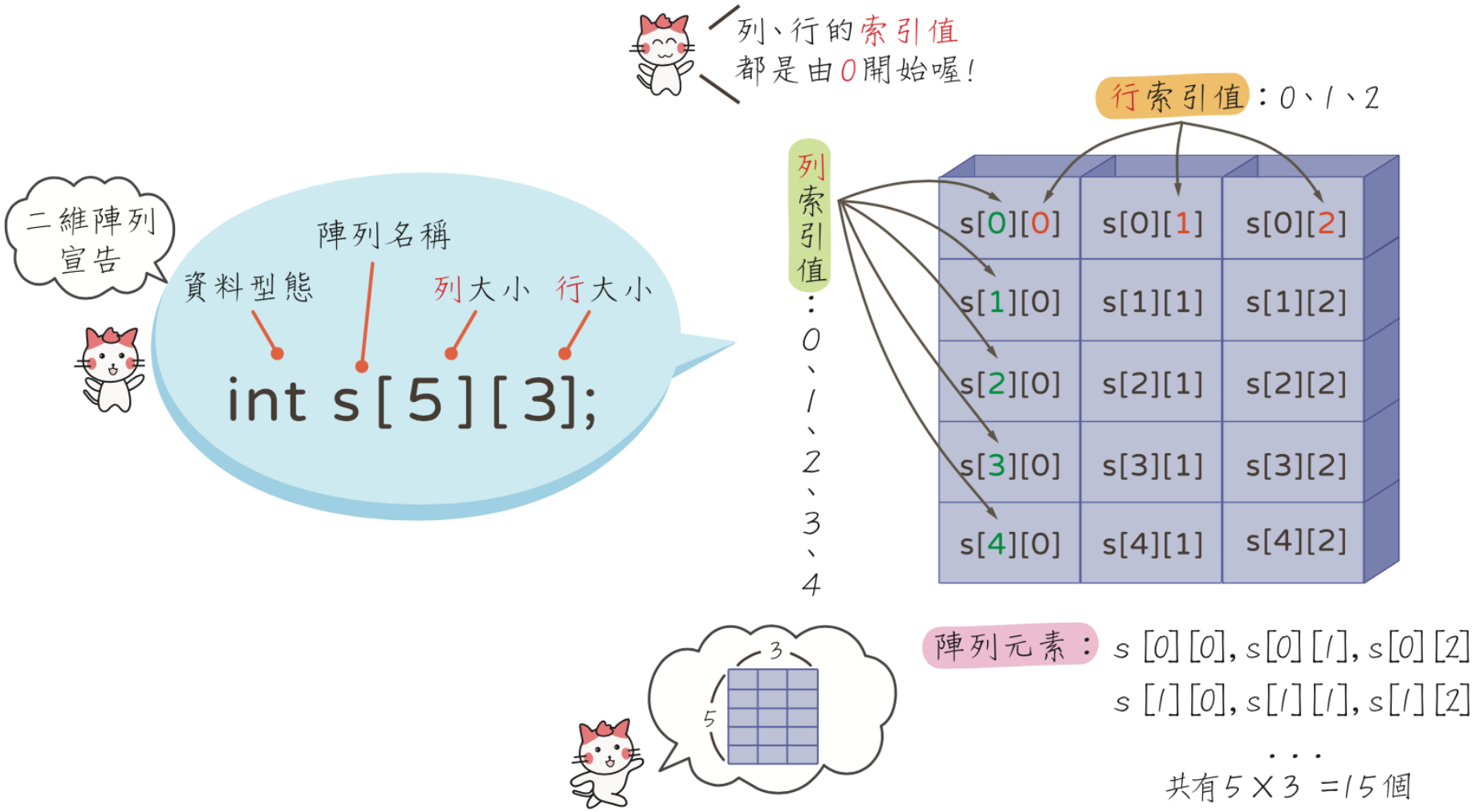
```
int s[5][3];
```

s[0][0]	s[0][1]	s[0][2]
s[1][0]	s[1][1]	s[1][2]
s[2][0]	s[2][1]	s[2][2]
s[3][0]	s[3][1]	s[3][2]
s[4][0]	s[4][1]	s[4][2]

最上面的第一排的3個置物格，都屬於Alice的

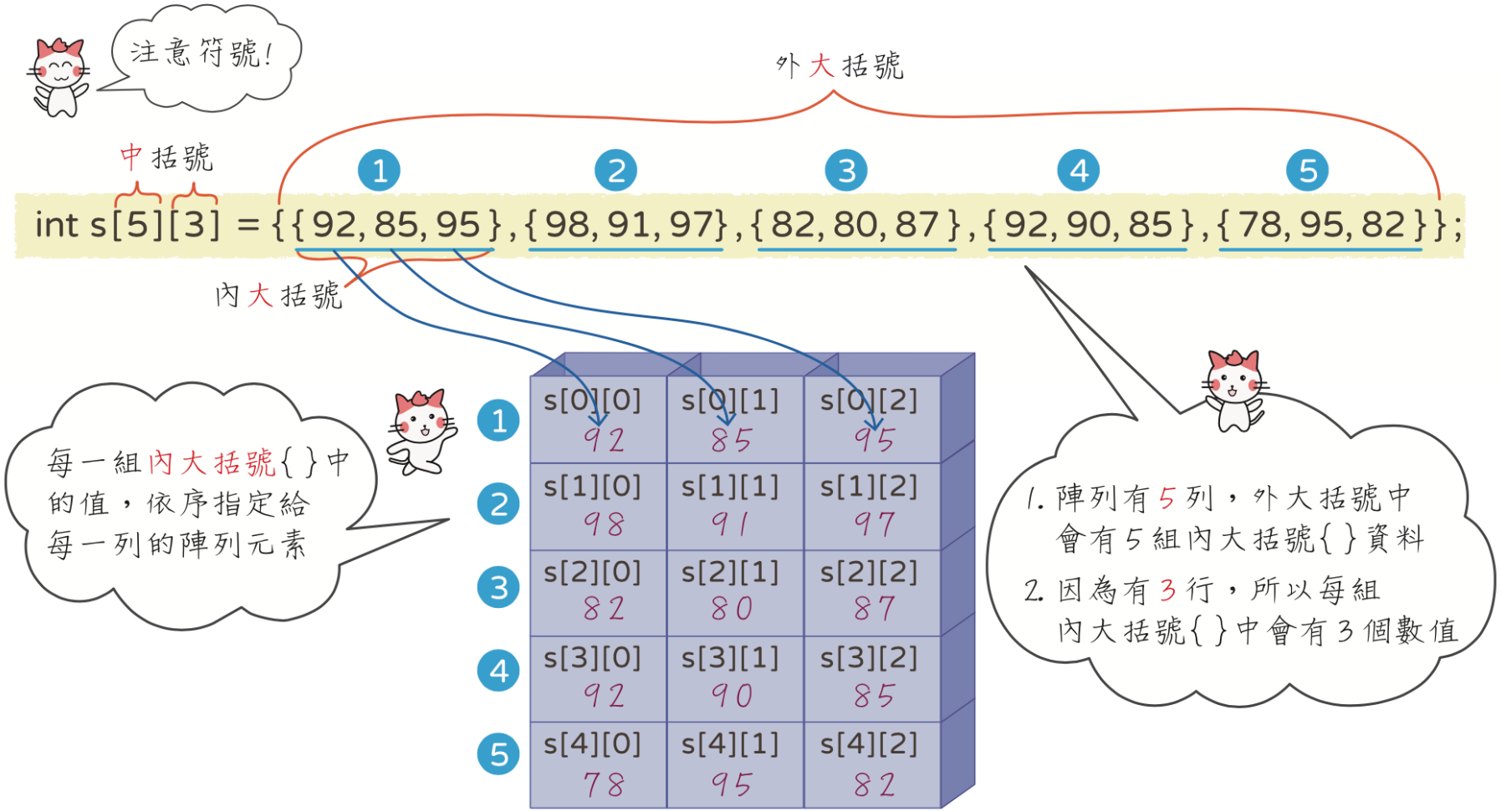
一、宣告

二維陣列的宣告，和一維陣列差不多，一起看一下吧。



二、指定值

要指定值給二維的陣列元素，也是和一維陣列類似，例如要把五位同學的國、英、數三個科目的成績存放到陣列元素中，可以用下列的方法。



三、二維陣列的使用

```
s[3][2] = 100;
x = s[3][2]*2;
```

指定100給二維陣列元素s[3][2]

x = 100 * 2 = 200

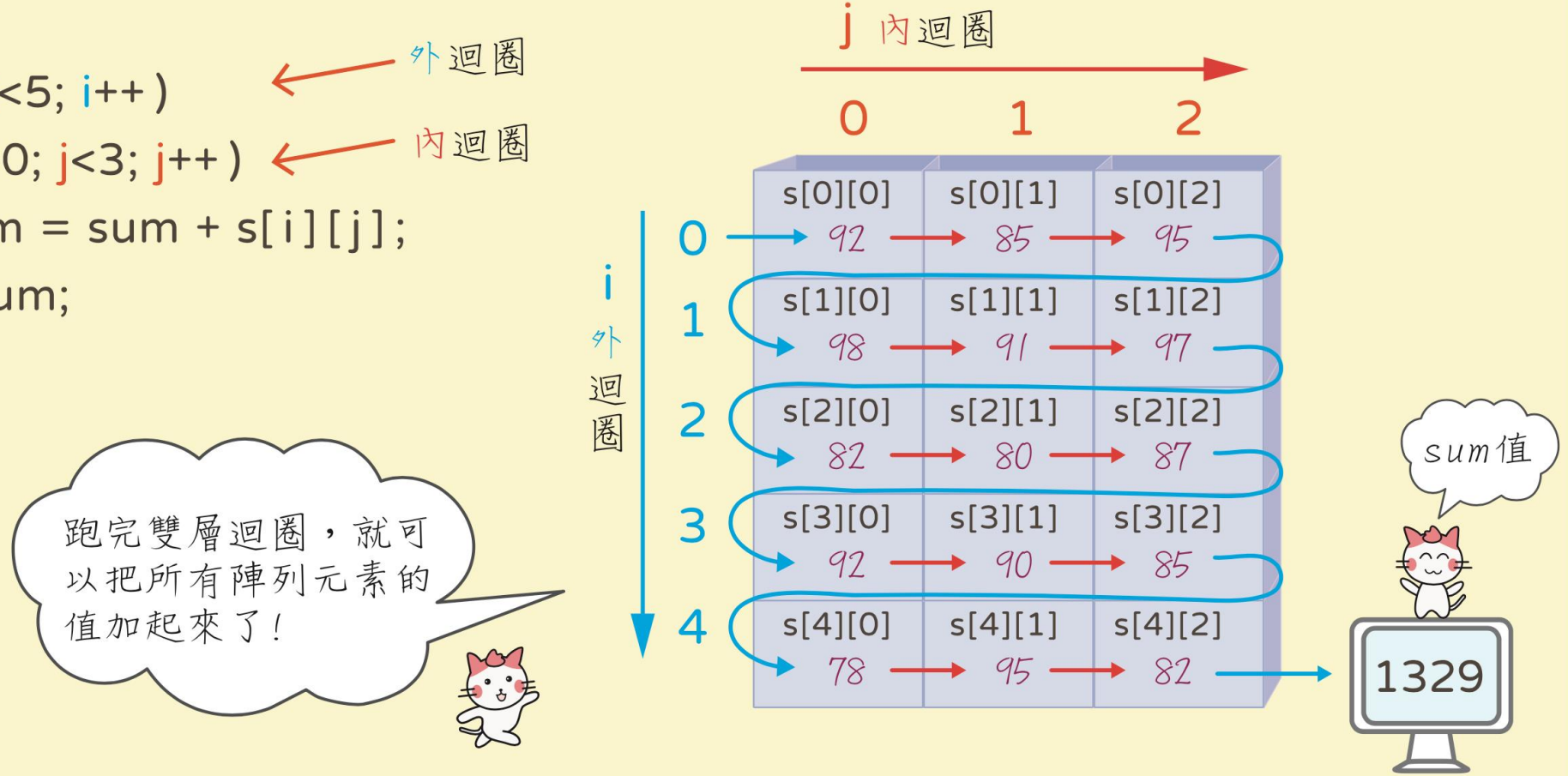
三、二維陣列的使用

對整個陣列的存取時，會常常搭配雙層for迴圈來使用喔！比如說，要計算五位同學國、英、數三科的總分可以這樣來設計。

```
for (i=0; i<5; i++)
    for (j=0; j<3; j++)
        sum = sum + s[i][j];
cout << sum;
```

外迴圈

內迴圈



實力打卡

（ 解 ） 1. 在 C++ 程式碼中宣告 a[2][4]，陣列 a 共有多少元素？

(A) 3 (B) 6 (C) 8 (D) 12 。

（ 解 ） 2. 在 C++ 程式碼中宣告 int a[3][2]={{1,2},{3,4},{5,6}};，a[2][2] 的值為何？ (A) 2 (B) 4 (C) 6 (D) a 陣列中沒有這個值 。

任務

校運會時，田徑三項的成績已登錄到表格中。請寫一程式，把五位選手的成績以陣列存放，再以迴圈計算每位參賽選手的總積分。

選手	100公尺	跳高	擲鉛球	總積分
Bob	90	85	92	
Andy	95	91	96	
Ben	82	80	88	
Denis	91	90	85	
Eric	80	95	82	

解析

1. 以二維陣列存放五位選手的田徑三項分數，如int s[5][3]
2. 若想把總積分一起放入陣列中，則可使用int s[5][4]
3. 仿照上述方式，計算每位同學的總分，即把下列程式碼進行修改：

```
sum=0;
for （i=0; i<3; i++）
        sum = sum + s[i];
```

(1) 使用雙迴圈

- 外層的for迴圈完成指出每位選手，如第i位
- 內層的for迴圈則負責取出外層for迴圈代表的選手每項分數，如第j項

(2) 本例不使用sum存放5位選手全部的總積分，而是改為s[i][3]存放個別選手的總積分

```
for （i=0; i<5; i++）
      for （j=0; j<3; j++）
            取出 s[i][j] 加入到s[i][3]中；
```

田徑三項多人成績處理

```
s[i][3] = 0;
for (i=0; i<5; i++)
    for (j=0; j<3; j++)
        s[i][3] = s[i][3] + s[i][j];
```

實例演練 田徑三項多人成績處理

實作

ch3-2- 二維成績陣列實作.cpp

執行結果

267 282 250 266 257

ch3-2-二維成績陣列實作.cpp

```cpp
#include <iostream>
using namespace std;

int main() {
    int s[5][4] ={{90,85,92,0},{95,91,96,0},{82,80,88,0},{91,90,85,0},{80,95,82,0}};
    int i,j;
    for (i=0; i<5; i++){
        for (j=0; j<3; j++)
            s[i][3] = s[i][3] + s[i][j];
    }

    for (i=0;i<5; i++)
        cout << s[i][3] <<" ";

    return 0;
}
```

宣告 5 列 4 行的二維陣列且指定初始值，每列最後一個元素做為存放計算出來的每位選手總分，存放總分的 s[i][3] 都指定為 0

3-2-1 搜尋演算法

一、循序搜尋

循序搜尋法(Sequential Search)就是按順序一個一個進行資料比對

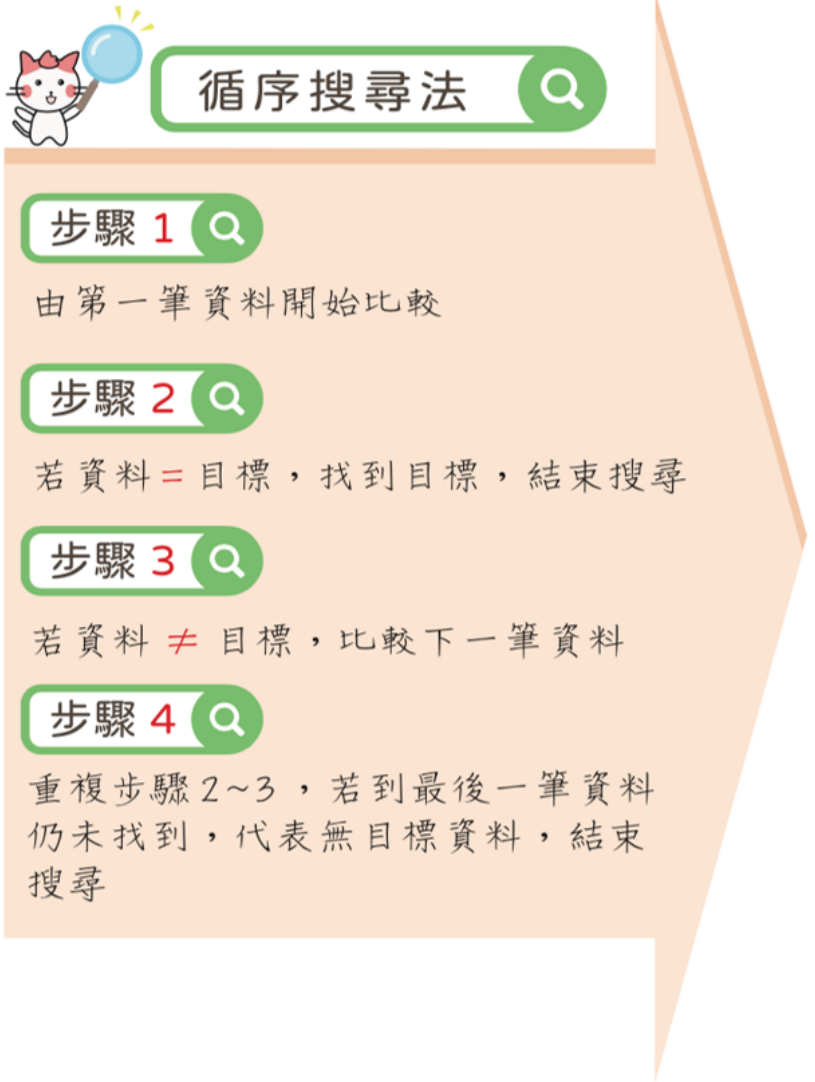
一、循序搜尋

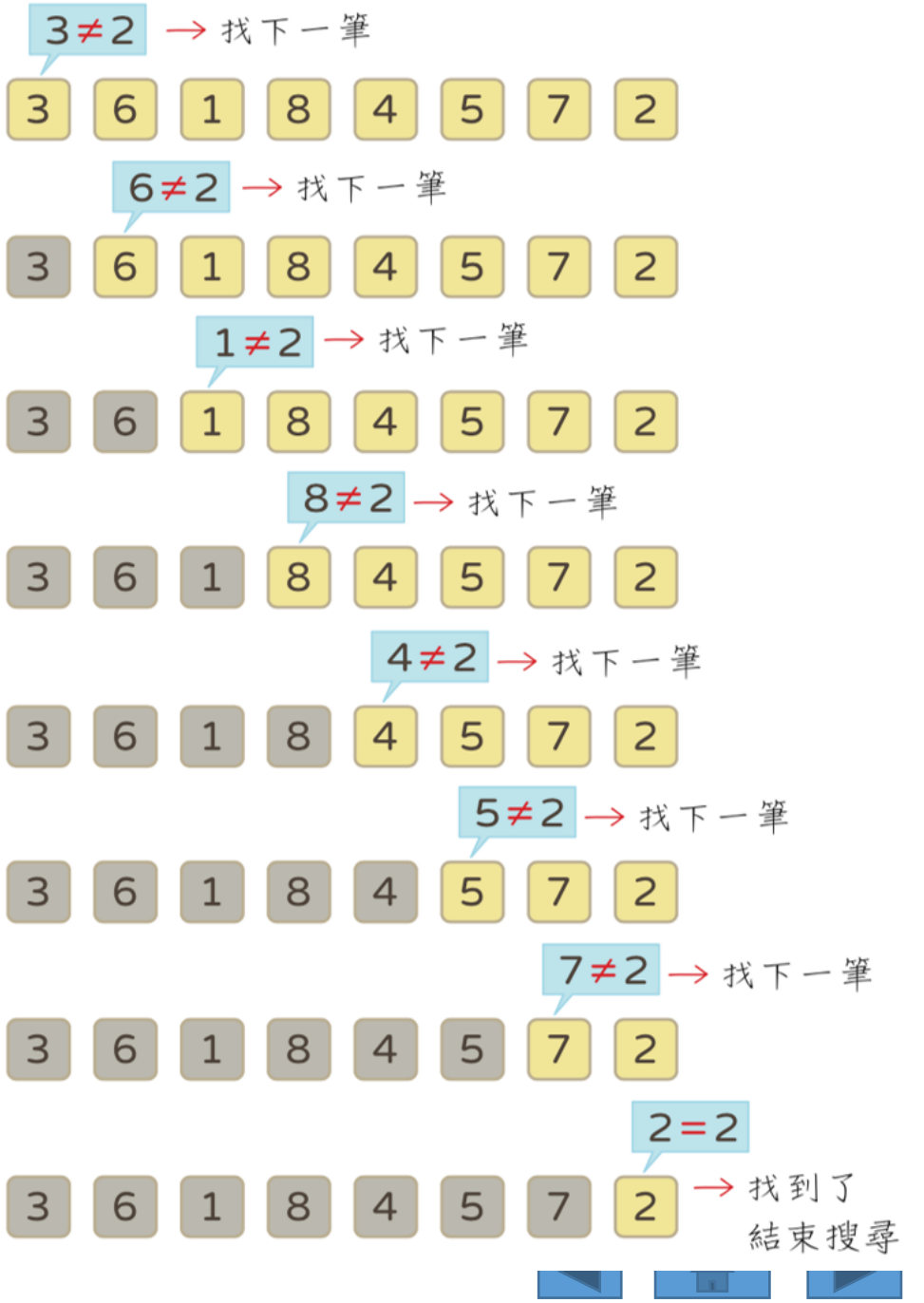
要從一組數字

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

中找到「2」，用

循序搜尋法如何運作？





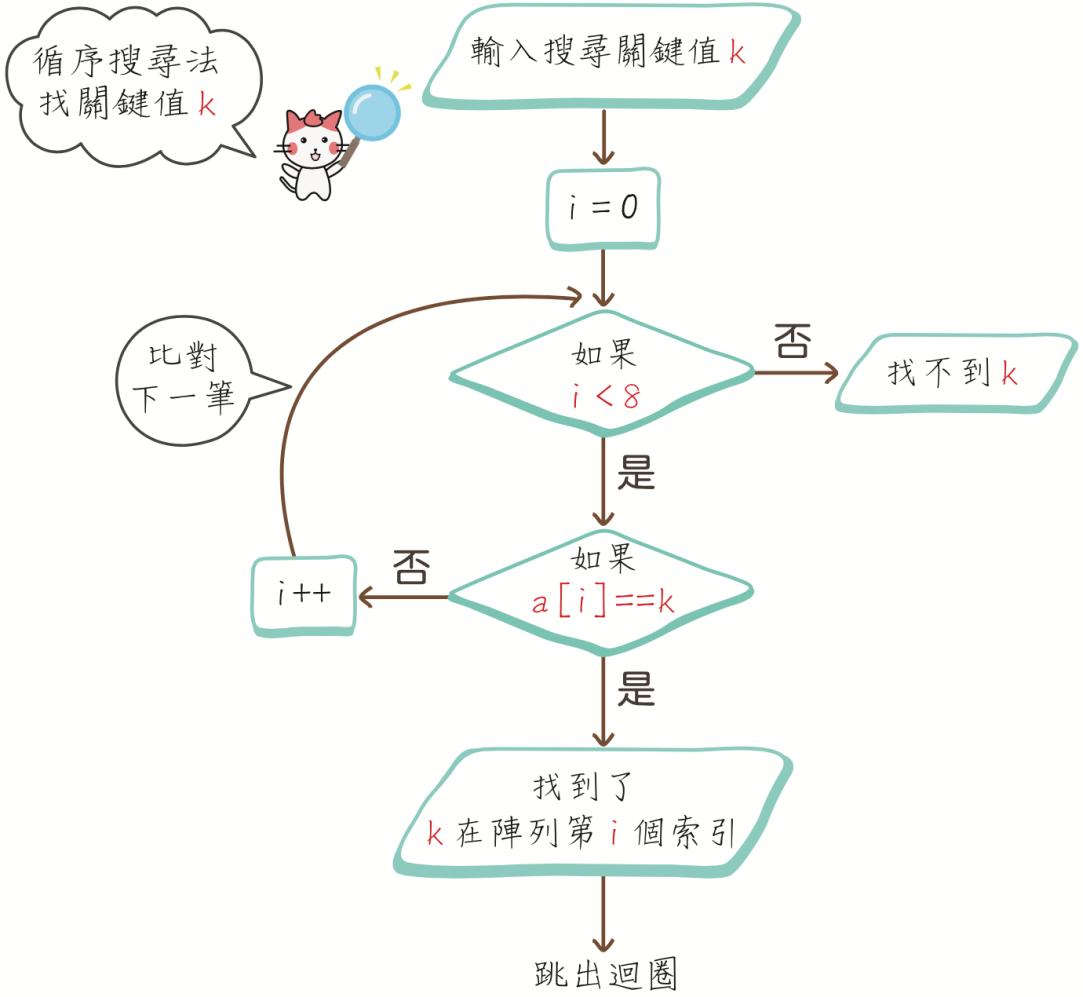
任務

使用循序搜尋法，從數列3, 6, 1, 8, 4, 5, 7, 2中進行關鍵值搜尋。找到時，輸出它的位置，否則輸出找不到。

實例演練 循序搜尋

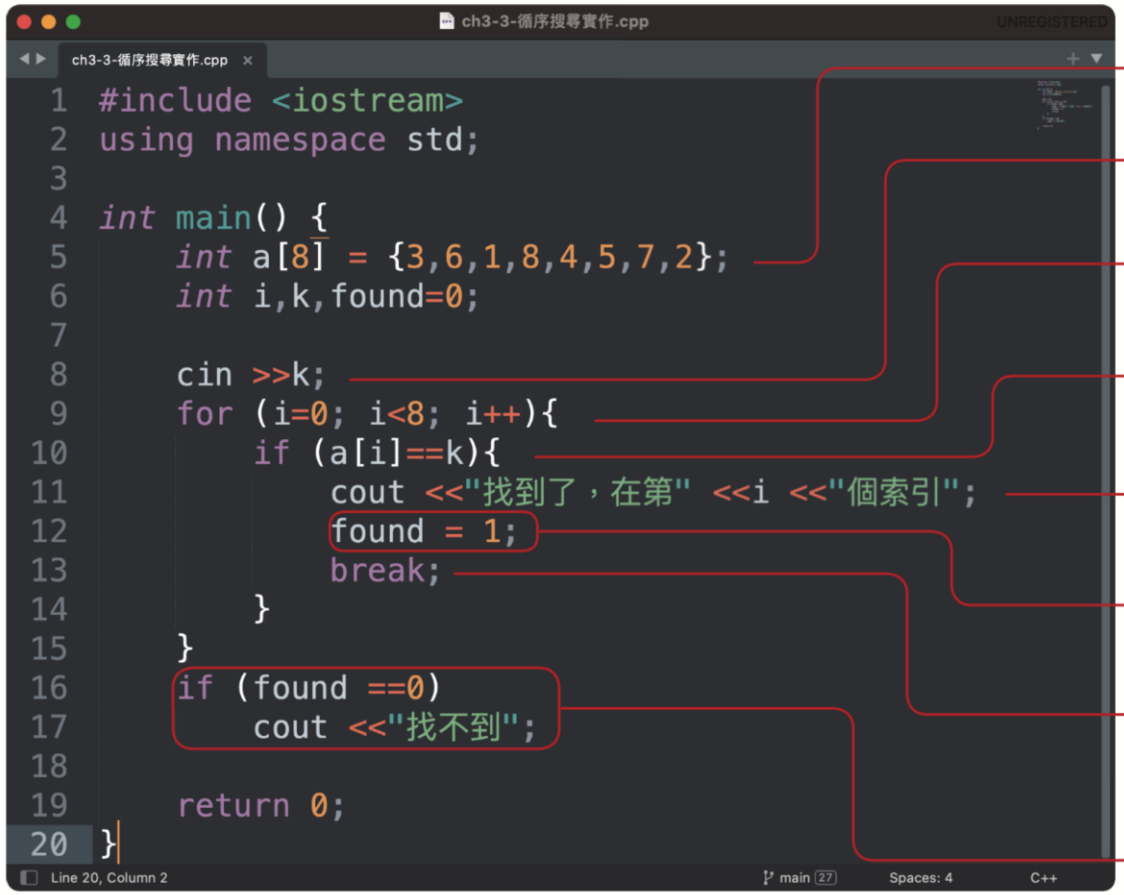
解析

1. 將數列存放在陣列中，a[8] = {3,6,1,8,4,5,7,2};
2. 輸入搜尋的關鍵值k
3. 使用for迴圈逐一檢查索引值i
   - 比較： if（a[i] == k）
   - 找到： 輸出陣列索引值 i
   - 跳離迴圈：利用break
4. 如果都找不到時，輸出「找不到」：
   - 在一開始時，指定變數 found=0
   - 如果找到，指定found=1
   - 程式最後檢查found若為0，則輸出找不到



實作

ch3-3- 循序搜尋實作 .cpp



宣告陣列及初始值

輸入關鍵值 k

i=0，即從索引值 0 開始找起

比較第 i 項是否等於關鍵值

輸出索引值

找到關鍵值，found 改為 1

跳離迴圈，到第 16 列

如果 found 還是 0，表示找不到

執行結果 1

2

找到了，在第 7 個索引

執行結果 2

10

找不到

知識情報

循序搜尋法的比較次數

從 n 個數字中搜尋資料時：

- 最差情形：需n次比較運算
- 最佳情形：只需1次比較運算
- 平均情形：需要 $(1+2+\cdots\cdots+n)\div n=\frac{(n+1)\times n}{2\times n}=\frac{n+1}{2}$ 次比較。
- 也就是說，最差的情況下，對n筆資料做循序搜尋的比較次數為n。

二、二分搜尋法

當資料量很大時，可先將資料事先排序，
改使用二分搜尋法(Binary Search)來進行搜尋

Alice選定一個介於 1 ~ 99 的數字做為謎底，由Bob當玩家來猜數字。每當Bob說出一個數字後，Alice會回覆「再大一些」、「再小一些」或「答對了！」。

假如Bob每次從數字範圍的正中間數開始猜測，那麼每猜一次就能縮小一半的資料範圍，這種技巧其實就是二分搜尋法的應用。

二、二分搜尋法

一組排序好的9個數字：

1,2,6,8,11,12,15,17,25

假如要找的目標值為「12」

二分搜尋法

步驟 1

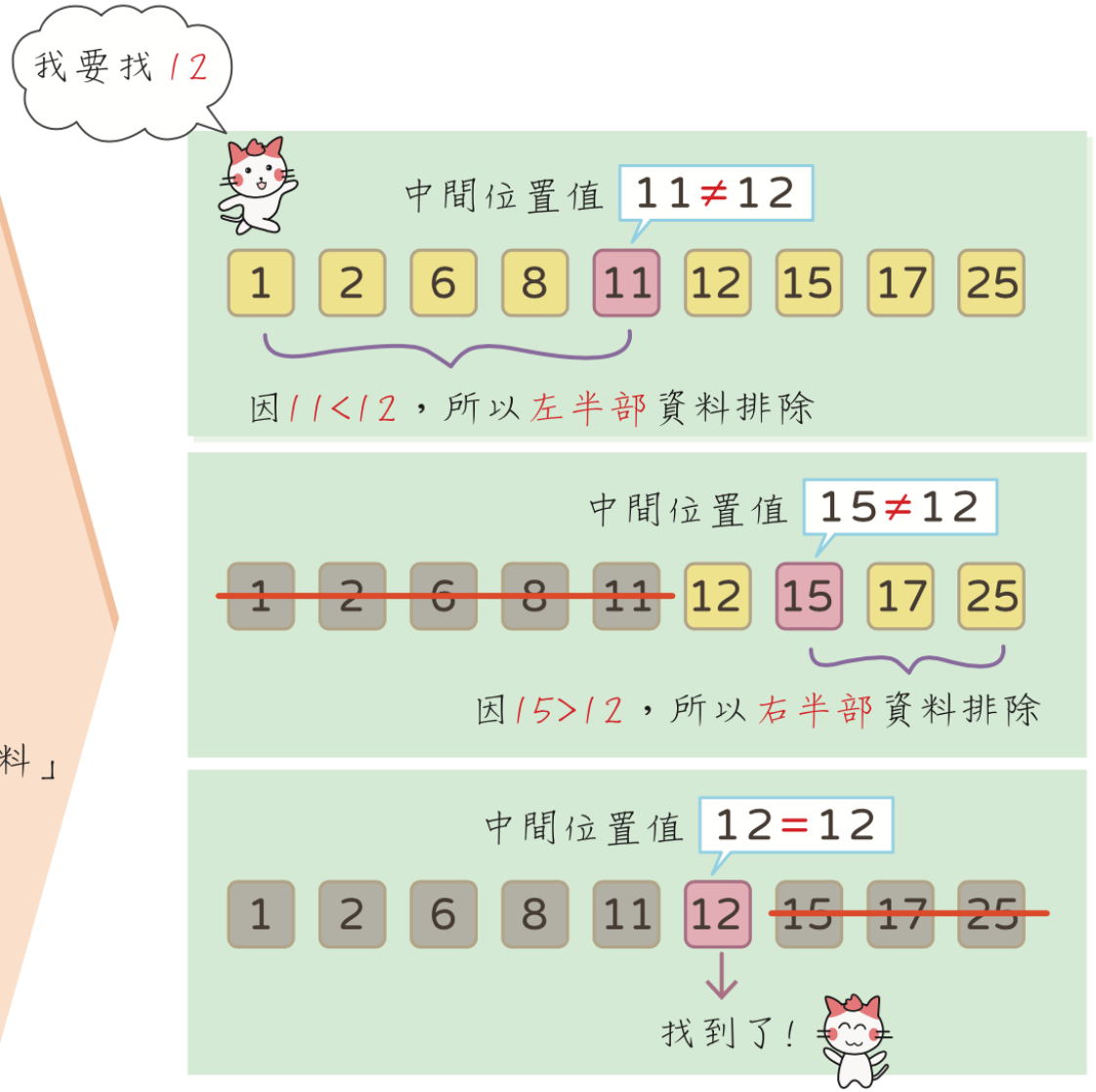
找出「待搜尋範圍」的「中間位置值」

步驟 2

- 若中間位置值＝目標值
  → 結束搜尋
- 若中間位置值≠目標值
  → 排除「中間位置值與另一側的資料」

步驟 3

重複步驟1~2，直到沒有待搜尋範圍



＊ 若資料數為「偶數」時，並不能剛好有「中間位置值」，選左邊、右邊數值皆可，本例採用取左邊的數值。

知識情報

二分搜尋法的比較次數

- 對含有n個數的數列搜尋時，每次搜尋後，剩下的數減半。
- 第 1 次搜尋後，約剩下 $\frac{n}{2}$ 個數

  第 2 次搜尋後，約剩下 $\frac{n}{2^2}$ 個數

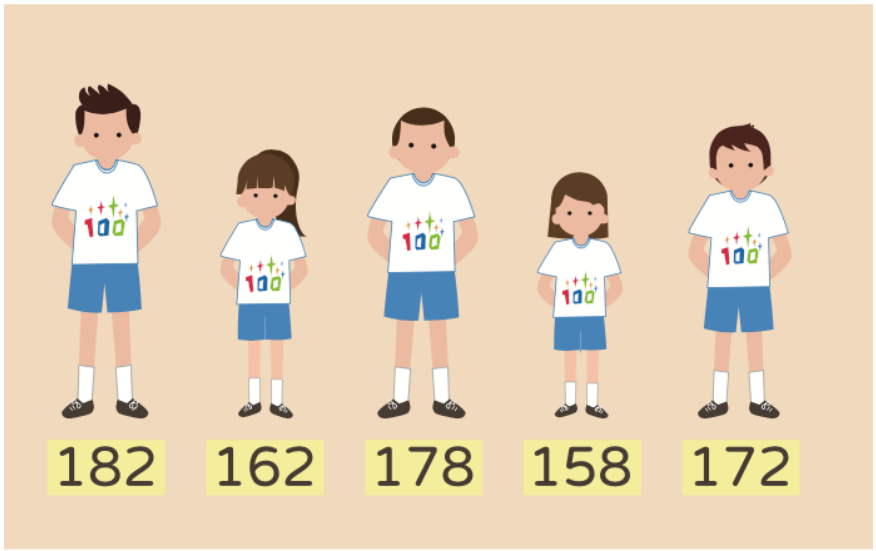
  第 3 次搜尋後，約剩下 $\frac{n}{2^3}$ 個數。

  以此類推，第k 次搜尋後，約剩下 $\frac{n}{2^k}$ 個數。

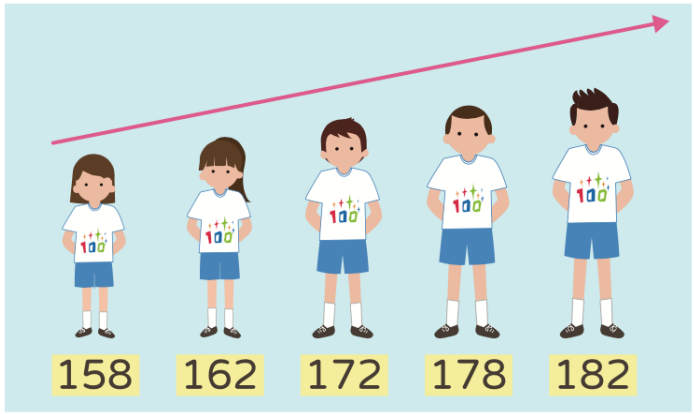
  當第 k 次搜尋後，只剩 1 個數時，$\frac{n}{2^k}=1$ 即$n=2^k$，

  $n=2^k$ 等號兩邊取對數值後，得到 $k=\log_2 n$。
- 也就是說，最差的情況下，對 n 筆資料做二分搜尋的比較次數約為$\log_2 n$。

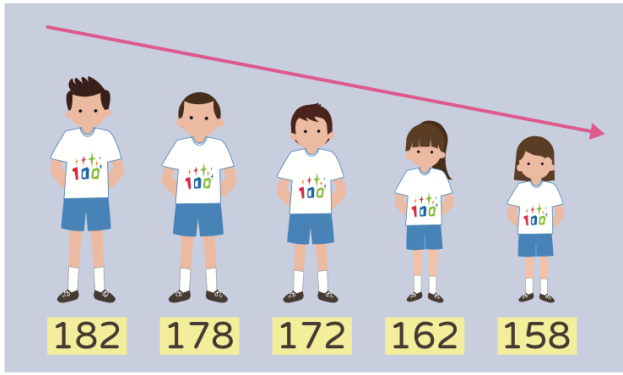
3-2-2 排序演算法



遞增排序



遞減排序



遞增排序是指將數字「由小到大」排序

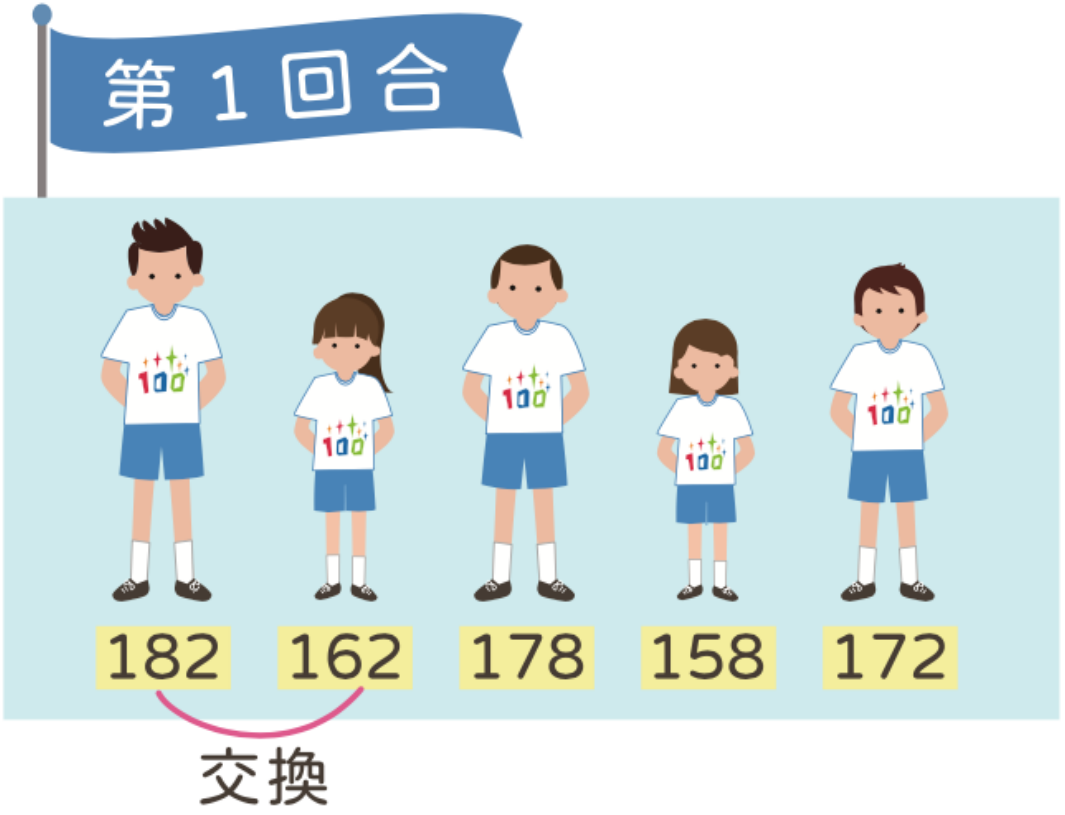
遞減排序是指將數字「由大到小」排序

一、氣泡排序

氣泡排序法(Bubble Sort)：重複與相鄰值比較與交換

步驟 1

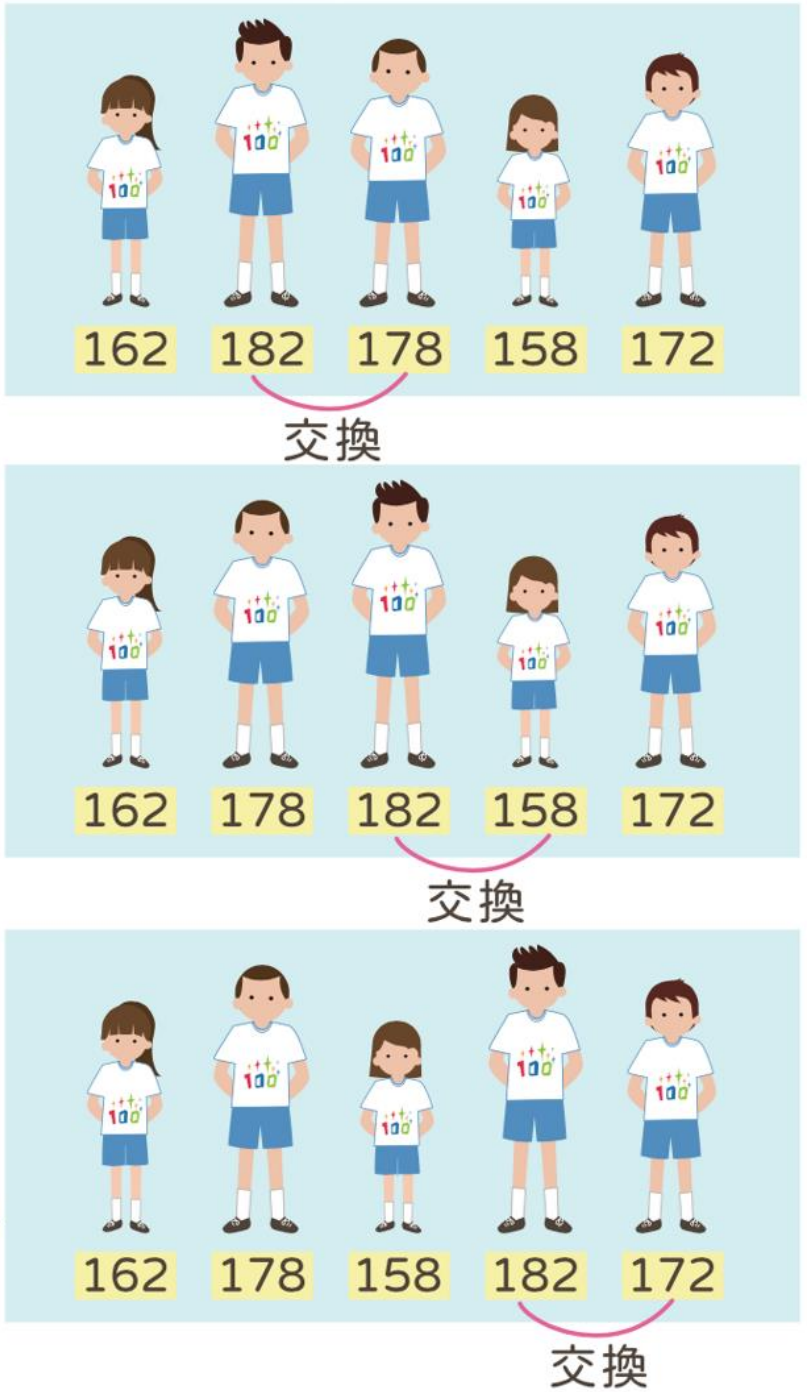
從左到右，相鄰兩位站立的同學進行比較，高的交換到右邊



一、氣泡排序

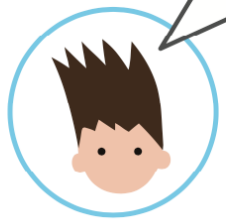
步驟 2

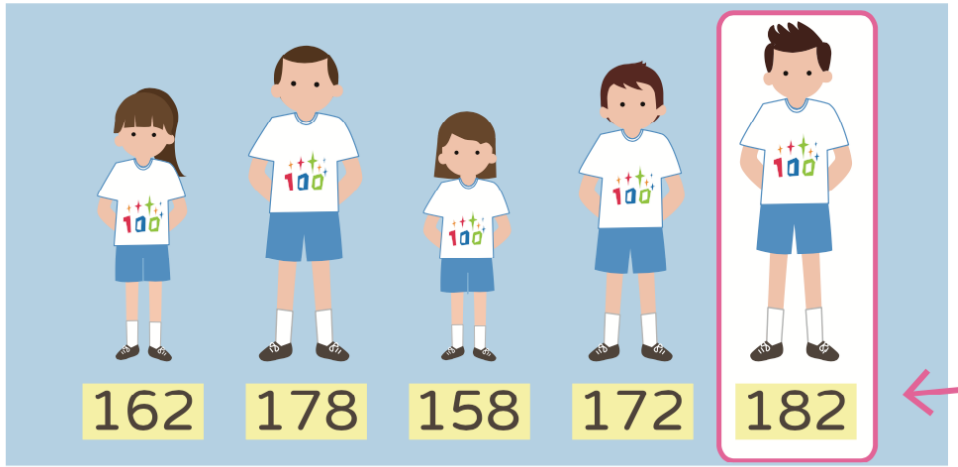
重複步驟 1 相鄰比較的動作，直到跟最右邊站立的同學比較完畢



第一回合，最高的同學已經排到最右邊了，
接下來第二回合只需排4位同學就可以了！

5 位同學進行排序，
在第 1 回合中，
一共比較 4 次，
交換 4 次



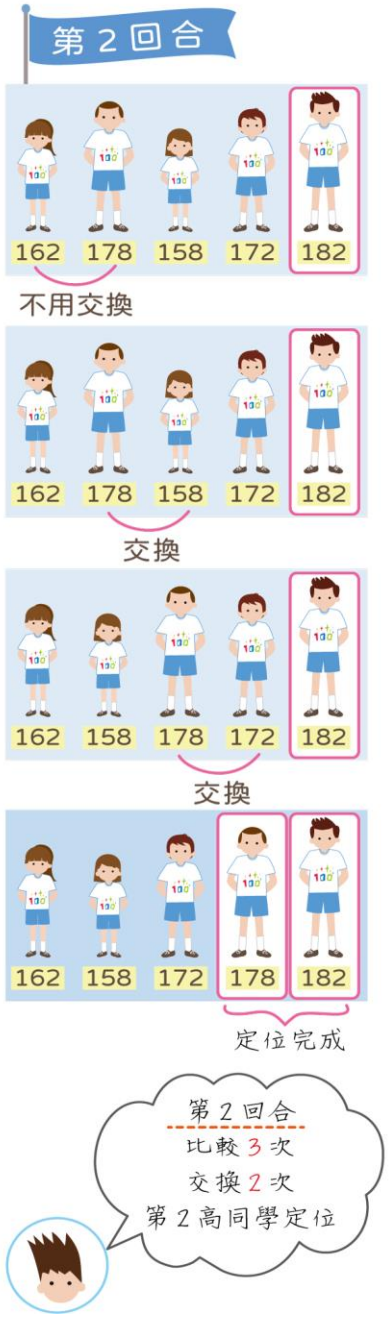


定位完成

經過一個回合，
最高的同學就被
交換到最右邊，
定位完成

一、氣泡排序

若5位同學，
利用氣泡排序法，
需經過4個回合，
比較次數：4+3+2+1=10次
才能完成排序作業。



實力打卡

運動會男子 100 公尺決賽成績出爐了，請利用「氣泡排序法」將 5 位選手的積分由小至大排序，並在下圖中寫出每一回合積分的排序結果。

	Bob	Andy	Ben	Denis	Eric
	90	95	82	91	80
第 1 回合					
第 2 回合					
第 3 回合					
第 4 回合					

解

100公尺
積分表

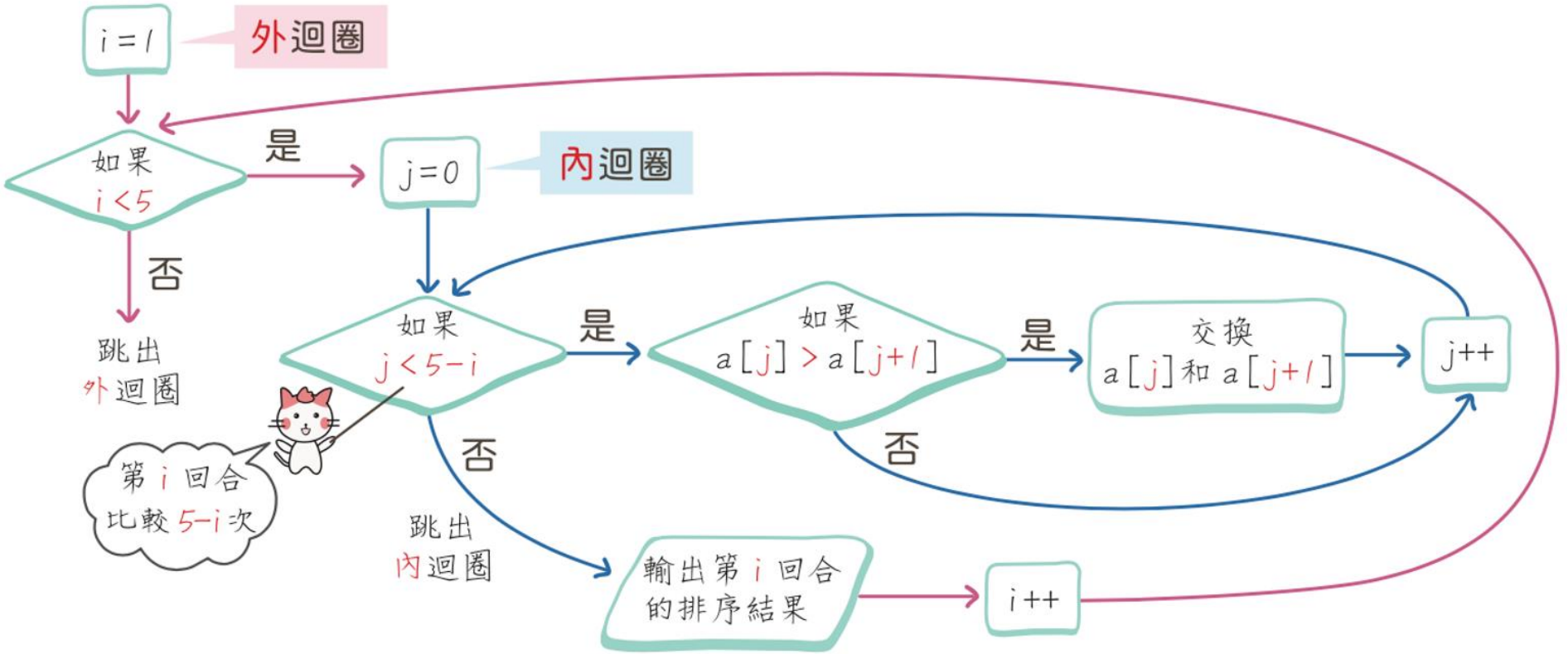
Bob	Andy	Ben	Denis	Eric
90	95	82	91	80
第1回合	90	82	91	80	95
第2回合	82	90	80	91	95
第3回合	82	80	90	91	95
第4回合	80	82	90	91	95

實例演練 氣泡排序

任務

使用氣泡排序法，把3,2,4,5,1 由小排到大。同時，輸出每一回合的排序結果，進行觀察。

解析



實例演練 氣泡排序

```
for(i = 1; i<5; i++){
    for(j=0; j<5-i; j++){
        if(a[j]>a[j+1]){
            tmp = a[j];
            a[j] = a[j+1];
            a[j+1] = tmp;
        }
    }
}
```

外迴圈

內迴圈

共4回合，i=1、2、3、4

相鄰兩數，進行比較

如果左數>右數

左右兩數，進行交換

tmp是交換時用來暫存資料變數

因為陣列索引值由0開始，所以設定j由0開始

1. 利用外迴圈控制回合數，有5個數會有「5-1=4」回合，每回合i的值分別為1, 2, 3, 4。

```
for(i=1; i<5; i++){
        第 i 回合的重複相鄰兩數比較與交換；
}
```

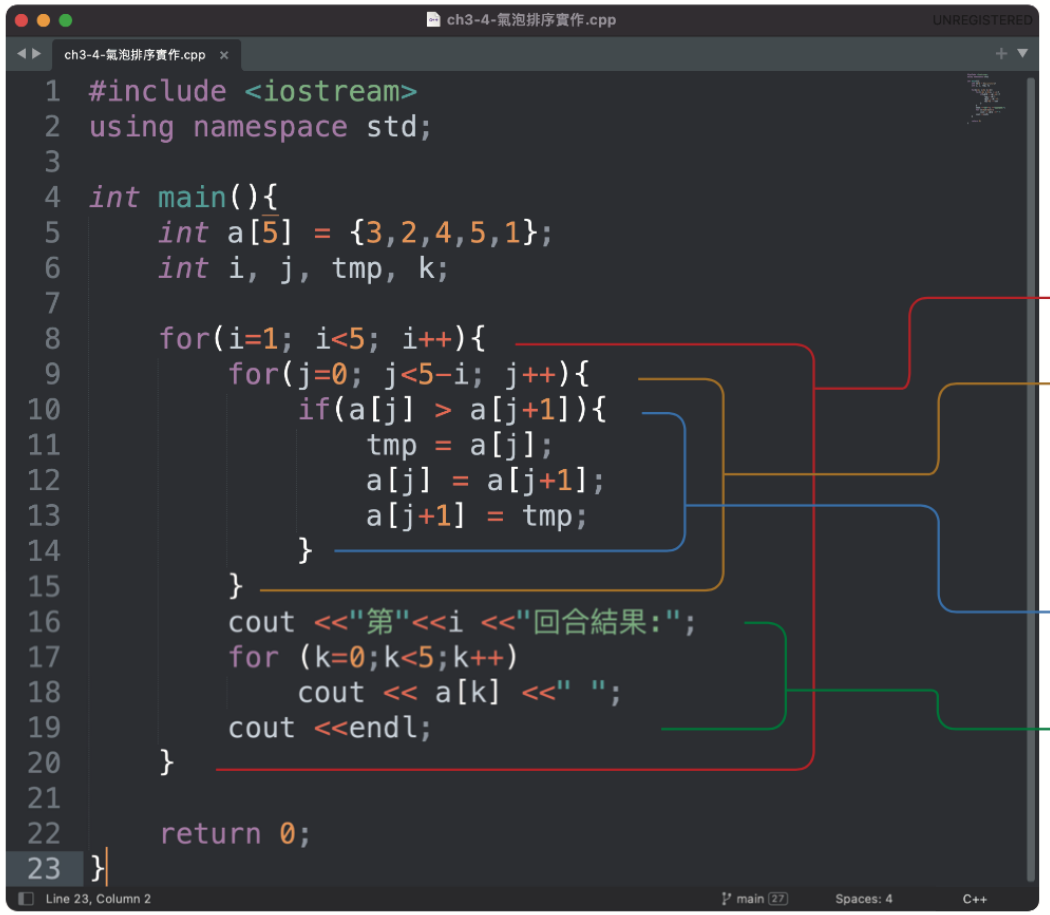
2. 內迴圈用來控制每一回合內，要完成「重複相鄰兩數比較與交換」的動作
   - j的值分別為0,1,2,……, (5-i)-1，即只有(5-i)次的比較，
     例如：i=1時，j從0到3，即有4 次比較。
   - 因為陣列索引值由 0 開始，所以 j 設定由 0 開始

```
for(j=0; j<5-i; j++){
        if（左數 > 右數）
            交換兩數；
}
```

實例演練 氣泡排序

實作

ch3-4-氣泡排序實作.cpp



外迴圈用來控制回合數

內迴圈用來控制每一回合內，要完成「重複相鄰兩數比較與交換」的動作

進行判斷，如果（左數 > 右數），則交換左右兩數

輸出第 i 回合的排序結果

執行結果

```
第1回合結果:2 3 4 1 5
第2回合結果:2 3 1 4 5
第3回合結果:2 1 3 4 5
第4回合結果:1 2 3 4 5
```

知識情報

氣泡排序法的比較次數

對n個數字進行氣泡排序，需n-1回合

1. 比較運算次數
   - 第1回合時，需n-1次
   - 第2回合時，需n-2次
   - 第3回合時，需n-3次…
   - 第n-1回合時，需1次
2. 總共$(n-1)+(n-2)+\cdots\cdots+1=\frac{(n-1)+1}{2}\times(n-1)=\frac{n(n-1)}{2}$次比較
3. 也就是說，最差的情況下，對n筆資料做氣泡排序的比較次數為 $\frac{n(n-1)}{2}$ 。

知識情報

合併排序[1]的比較次數

註:1. 請參考 2-2-2 節。

對 n 個數字進行合併排序，需要幾個回合的合併和比較呢？

- 總共需要「幾回合」的合併？

在P.59圖中的例子中，在完成合併前要先進行「比較大小」，第1回合把8個小陣列合併成4個，第2回合再從4個合併成2個，第3回合（也就是最後）合併成1個。每回合的合併，都可以讓下一回合需要合併的陣列數減少一半，直到只剩1個陣列為止，總共需要合併$\log_2 n$回合，例如8個數字的數列，共需合併3回合($\log_2 8=3$)。

知識情報

合併排序[1]的比較次數

註:1. 請參考 2-2-2 節。

- 每回合的合併需要幾次的比較呢？

1. 第 1 回合每兩個小陣列（各只有 1 個數字），最多需要比較 1 x 2 - 1 = 1 次，共有 $\frac{n}{2}$ 對小陣列做合併，比較次數為 1x $\frac{n}{2}$ 次
2. 第 2 回合每兩個小陣列（各有 2 個數字），最多需要比較 2 x 2 - 1 = 3 次，共有 $\frac{n}{4}$ 對小陣列做合併，比較次數為 3x $\frac{n}{4}$ 次
3. 第 3 回合每兩個小陣列（各有 4 個數字），最多需要比較 4 x 2 - 1 = 7 次，共有 $\frac{n}{8}$ 對小陣列做合併，比較次數為 7x $\frac{n}{8}$ 次

知識情報

合併排序[1]的比較次數

註:1. 請參考 2-2-2 節。

以此類推，最後回合只剩兩個小陣列（各有 $\frac{n}{2}$ 個數字），最多需要比較 $= \frac{n}{2} \times 2 - 1$ $= n - 1$ 次，只剩 1 對小陣列做合併，比較次數為 $(n-1) \times 1$ 次。

也就是說，最差的情況下，每回合最多需要比較次數約為 n，共有 $\log_2 n$ 回合，即對 n 筆資料做合併排序的比較次數約為 $n \times \log_2 n$。

3-3 演算法效能分析

- 程式的執行時間，雖然跟**硬體**(如CPU及記憶體)有關，但是也跟**演算法**設計的好壞大有關係
- **效能分析**可以做為評估演算法好壞的一種方式

3-3 演算法效能分析

- 對 n 筆資料做循序搜尋的比較次數為 n
- 對 n 筆資料做二分搜尋的比較次數為 $\log_2 n$
- 對 n 筆資料做氣泡排序的比較次數為 $\frac{n(n-1)}{2}$
- 對 n 筆資料做合併排序的比較次數為 $n\log_2 n$

3-3 演算法效能分析

▼表 3-1　不同演算法所需的最多比較次數

資料數量	比較次數			
	氣泡排序	合併排序	循序搜尋	二分搜尋
1	0	0	1	1
100	4950	664	100	7
1,000	499500	9,966	1,000	10
1,000,000	499,999,500,000	19,931,569	1,000,000	20
1,000,000,000	499,999,999,500,000,000	29,897,352,854	1,000,000,000	30
n	$\frac{n(n-1)}{2}$	$n\log_2 n$	n	$\log_2 n$

3-3 演算法效能分析

當n越來越大時，比較次數的差異就越巨大，也就是 $\frac{n(n-1)}{2}$ 大於$n\log_2 n$，再遠大於n，又遠大於 $\log_2 n$ 。因此，當資料量大時，更需要謹慎考量演算法的效能來挑選演算法。

「大家看一下比較圖，是不是更明白了呢？」