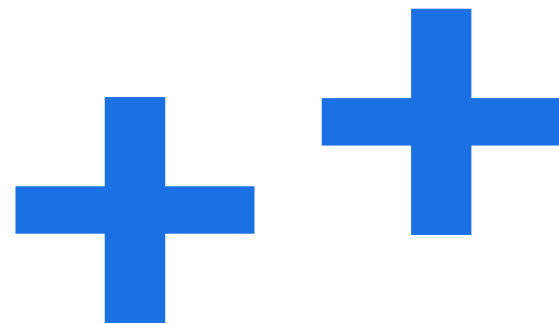


排序



排序的目的之一

- 較容易閱讀
- 利於統計及整理
- 可大幅減少數據搜尋的時間



5個整數，
由小到大排列。

問題



可以這樣比...

原始數據	第一回合				第二回合				第三回合				第四回合			
4	3	3	3	3	2	2	2	2	2	2	2	2	1	繼續往下比...		
3	4	2	2	2	3	3	3	3	3	1	1	1	2			
2	2	4	4	4	4	4	1	1	1	3	3	3	3			
5	5	5	5	1	1	1	4	4	4	4	4	4	4			
1	1	1	1	5	5	5	5	5	5	5	5	5	5			

方法：

比4大回合，

每一回合又比4小回合

兩兩相比，小的往上提



將方法變成 程式碼

```
#include <stdio.h>
int main() {
    int array[11];
    int i,j,temp;
    for (i=0;i<5;i++)
        cin>>array[i];

    for (i=0;i<5;i++){ //比4大回合
        for(j=0;j<5;j++){ //比4小回合
            if(array[j]>array[j+1]){
                temp=array[j];
                array[j]=array[j+1];
                array[j+1]=temp;
            }
        }
    }
    for (i=0;i<5;i++)
        cout<<array[i]<<endl;

    return 0;
}
```

輸入用的
程式碼

若前一個比
後一個大，
就做交換

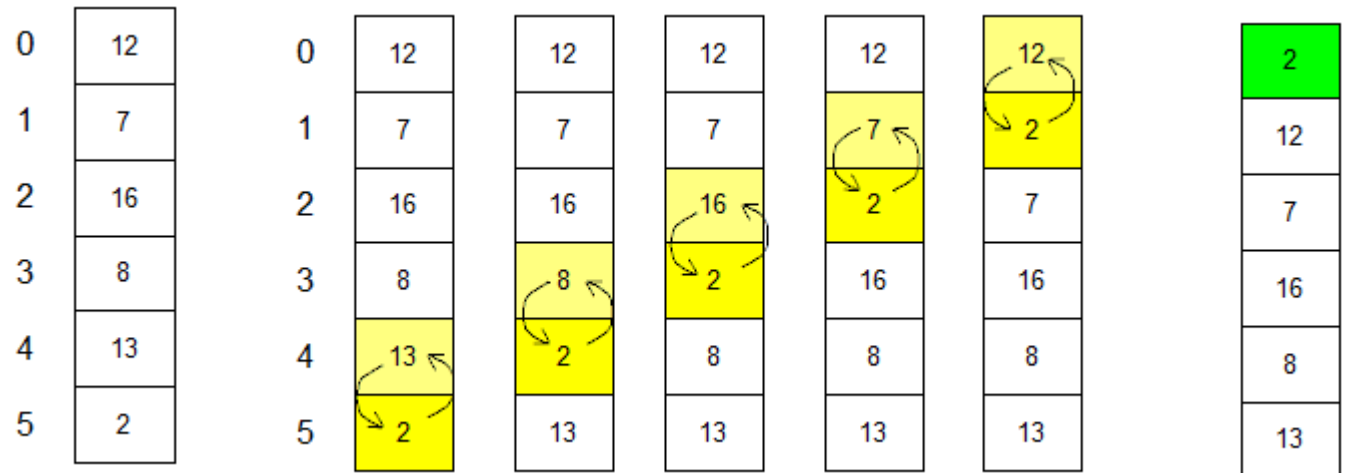
輸出用的
程式碼

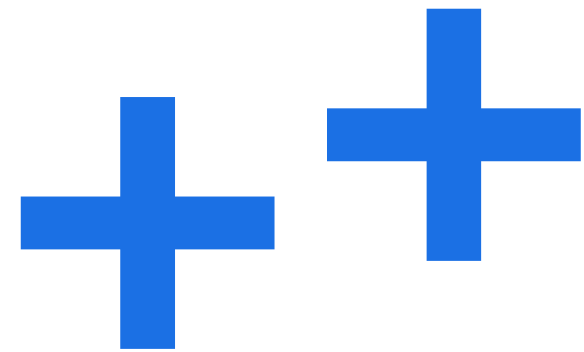


這是氣泡排序法 (Bubble Sort)

- 重複地走訪要排序的數列，一次比較兩個元素，如果它們的順序錯誤，就把它們交換過來。
- 走訪數列的工作是重複進行，直到不再需要交換，也就是說該數列已經排序完成。
- 越小的元素會經由交換，慢慢「浮」到數列的頂端。

CS50說氣泡





還有其他方法嗎？



也可以這樣比

原始數據	Step 1	Step 2	Step 3
4	1	1	1
3	4	2	2
2	3	4	3
5	2	3	4
1	5	5	5

方法：
在每一回合中找出最小值



將方法變成程式碼

歡迎參考 [edX 上開授的 CS 50 課程](#) 示範影片：

```
#include<iostream>
using namespace std;
int main() {
    int num[10], flag;
    int i, j, temp=0, t;
    for (i=0; i<5; i++){
        cin>>num[i];
    }
    for (i=0; i<5; i++){
        flag=i;
        for(j=i+1; j<5; j++){
            if(num[flag]>num[j]){
                flag=j;
            }
        }
        temp=num[flag];
        num[flag]=num[i];
        num[i]=temp;
        for(t=0; t<5; t++){
            cout<<num[t]<<" ";
        }
        cout<<endl;
    }
    return 0;
}
```

```
[C++_28_2]$ ./"main"
5
4
3
2
1
1 4 3 2 5
1 2 3 4 5
1 2 3 4 5
1 2 3 4 5
1 2 3 4 5
[C++_28_2]$
```



這是選擇排序法 (Selection Sort)

- 在所有的數據中，
當由大至小排序，則將最大值放入第一位置；若由小至大排序時，則將最大值放入位置末端。
- 例如當N個數據需要由大至小排序時，首先以第一個位置的資料，依次向2、3、4 ...N個位置的資料作比較。
- 如果數據大於或等於其中一個位置，則兩個位置的數據不變；若小於其中一個位置，則兩個位置的數據互換。



排序，還有很多種方法呢！

- 內部排序：排序的數據量小，可以完全在內存內進行排序。
 - 氣泡排序法 (Bubble Sort)
 - 選擇排序法 (Selection Sort)
 - 插入排序法 (Insertion Sort)
 - 合併排序法 (Merge Sort)
 - 快速排序法 (Quick Sort)
 - 堆積排序法 (Heap Sort)
 - 謝耳排序法 (Shell Sort)
 - 基數排序法 (Radix Sort)
- 外部排序：排序的數據量無法直接在內存進行排序，而必須使用到外存儲器。
 - 直接合併排序法 (Direct Merge Sort)
 - k路合併法 (k-Way Merge)
 - 多相合併法 (Polyphase Merge)

