Chapter 3.

基礎 C++ 程式設計導論 III

Introduction to Basic C++ Program Design III

講師：陳俊安 Colten

排序 sorting

- 字面上意思來說，就是將某個東西按照某種規則順序排好
- 像是 小 -> 大 或是 大 -> 小
- 而將一個東西整理成我們要的規則順序的過程就稱為排序

C++ 內建的排序工具 sort

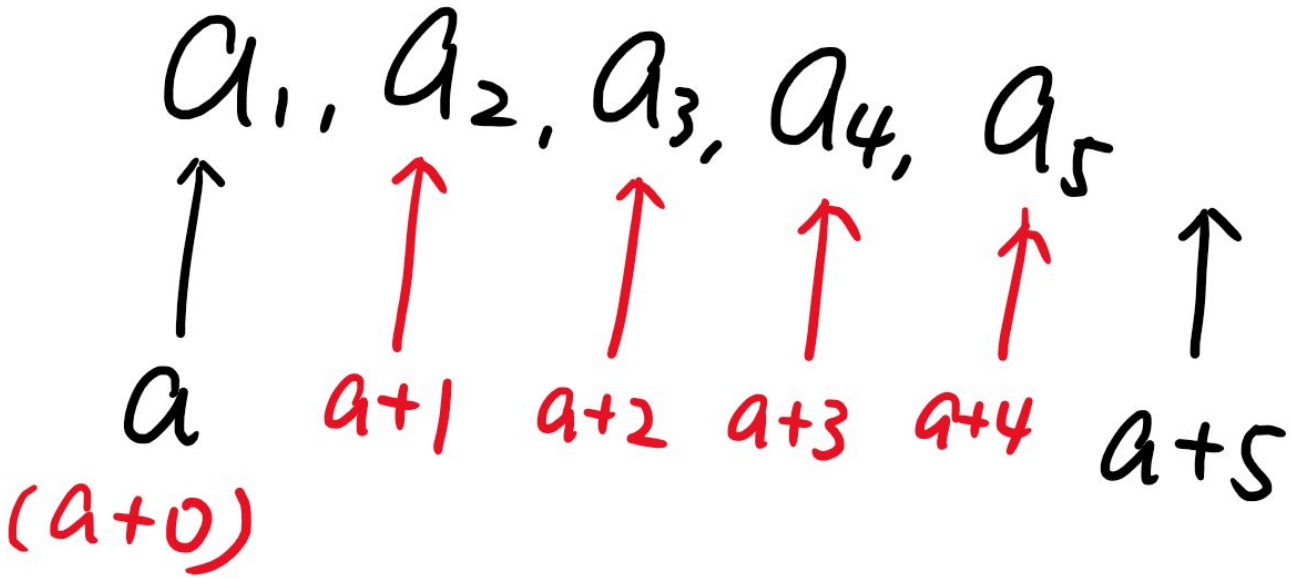
- 快速方便又好用（要 #include <algorithm>）
- 可以在有效率的時間內完成排序
- 事實上，排序的方法有很多種，每一種的效率的不太一樣
  - 氣泡排序
  - 插入排序
  - 選擇排序
  - 合併排序
  - and so on …

sort(begin , end);

- begin 與 end 這兩個位置所要放置的東西會是指標的形式
- 表示你要排序某個序列中區間 [begin , end) 的位置 (左閉右開)
- Ex. sort(a + 2 , a + 5); // 排序 a 序列中索引 [3 , 6)
- 預設會是由小到大排序

```
12      int a[5] = { 5 , 4 , 3 , 2 , 1 };
13      sort( a + 2 , a + 5 );
14
15      for(int i=0;i<5;i++) cout << a[i] << " ";
16      // 5 4 1 2 3
```

一般陣列的指標



sort(begin , end);

- 如果要排序整個 a 序列就會寫成：
- sort(a , a + 5); // a 就是 a + 0 的意思
- 記得是左閉右開的區間，雖然 a 序列長度只有 5，但是指標的位置必須是 a + 5

Function 函式

- 想像成一個具有完成特定功能的工具
- 像是 C++ 的 pow 函式
- pow(2,10) = 2^10 = 1024

Function 在數學上的例子

- f(x) = 4x + 7
- 當 x = 5 , y = f(x) = 27

在數學上也就意味著把 x = 5 帶進去函數裡面，得出一個新的結果

Function 在程式上

- 概念跟數學上的函數差不多，但是能做到的事情更多
- 整個過程很像在設計方程式，依照你自己設計的方程式，代入相對應的參數的時候會得出不一樣的結果

```c
int f(int x)
{
    return 4 * x + 20;
}
int main()
{
    printf("%d\n",f(5)); // 輸出 40
    printf("%d\n",f(6)); // 輸出 44
}
```

```c
int f(int x)
{
    return x / 2;
}
int main()
{
    printf("%d\n",f(5)); // 輸出 2 ( 不是 2.5，因為 f 函數是一個整數的函數 )
    printf("%d\n",f(6)); // 輸出 3
}
```

Function 的型態

- 每一個 Function 都必須有一個屬於自己的型態
- 就跟變數一樣，int、float、string 等等
- 這些型態意味著這一個 Function 會 return 什麼東西回來
  - 像是 main function 是 int 型態
  - 所以我們才會寫 return 0;

Function 的架構 - 函數的型態

首先你會發現所有函數前面都會有一個型態的宣告

這麼做是為了讓你的程式知道，你這一個函數會 return 什麼型態的東西

```
15 int solve1(int num)
16 {
17     return num * 10; // int
18 }
19 double solve2(double num)
20 {
21     return num * 3.14;
22 }
```

```
23 char solve3(char u)
24 {
25     return u;
26 }
27 bool solve4(int n)
28 {
29     if( n % 2 == 0 ) return true;
30     return false;
31
32     /* 也可寫成
33
34         if( n % 2 == 0 ) return 1;
35         else return 0;
36
37         或是
38
39         return ( n % 2 == 0 );
40     */
41 }
```

Function 的參數（Parameter）

- pow(2,10) 的 2 跟 10 我們就稱為 參數
- 你可以自己決定你的 Function 要塞幾個參數
- 假設你要 3 個參數

```
 7 int add(int a,int b,int c)
 8 {
 9     return a + b + c;
10 }
```

Function 的回傳（Return Value）

- 你的 Function 是什麼型態的就要回傳該型態的 Value 回來

```
 7 int add(int a,int b,int c)
 8 {
 9     return a + b + c;
10 }
```

```
12 void ouo(int a,string c)
13 {
14     cout << c << "\n";
15     return;
16 }
```

```
18 string owo()
19 {
20     return "Hello World\n";
21 }
```

Function

```cpp
int add(int a,int b,int c)
{
    return a + b + c;
}
void ouo(int a,string c)
{
    cout << c << "\n";
    return;
}
string owo()
{
    return "Hello World";
}
int main()
{
    cout << owo() << "\n"; // Hello World
    cout << add(1,2,3) << "\n"; // 6

    ouo(10,"OwO"); // OwO
}
```

其他要注意的東西 - 順序關係（由上到下）

```
16 // solve2 要寫在 solve 之前 ( 不然在編譯的時候程式會不知道 solve2 是什麼東西 )
17
18 int solve2(int num) // solve1 以參數 num = 5 呼叫 solve2
19 {
20     return num * 2; // 回傳 10
21 }
22 int solve(int num) // 傳入 5
23 {
24     return solve2(num) * 4; // 得到 solve2 的回傳 10 之後 * 4 =  40
25 }
26
27 int main()
28 {
29     printf("%d",solve(5)); // 輸出 40
30 }
```

補充：事先定義好 Function

```
 4 int solve2(int num); // 事先定義 solve2 這一個 Function 讓程式知道他是誰
 5
 6 // 只要可以讓程式先知道 solve2 是誰就可以讓 solve 寫在 solve2 前面了
 7
 8 int solve(int num) // 傳入 5
 9 {
10     return solve2(num) * 4;
11 }
12 int solve2(int num) // 將之前定義的 solve2 補上完整的內部資訊
13 {
14     return num * 2;
15 }
16
17 int main()
18 {
19     printf("%d",solve(5)); // 輸出 40
20 }
```

學完 Function 我們來看看，sort 只能由小到大排序　嗎

- 當然不是！如果你想要自定義規則的話就必須要自己寫 compare
- compare function 的型態為 布林 (bool)，且包含兩個參數
- 參數的型態依照排序的序列型態而定
- 序列是什麼型態參數型態就是什麼
- 至於回傳的值當然只有 true 或 false

什麼時候回傳 true 什麼時候回傳 false

- compare 回傳的規則定義為：
- 第一個參數是否有比第二個參數小
- 因此如果當兩個參數相同大小時，切記！！！一定要回傳 false
- 不然你將會得到 Runtime Error

一切就緒！！將 sort 函式加入第三個參數即可

第三個參數要放什麼取決於你的 compare 的函式名稱

名稱叫 apple 就放 apple，banana 就放 banana

```
sort(a,a+5,cmp);
```

compare 範例（由大到小）

```cpp
#include <iostream>

#include <algorithm>

using namespace std;

bool cmp(int a,int b)
{
    return a > b;
}
int main(void)
{
    int a[5];

    for(int i=0;i<5;i++) cin >> a[i];

    sort(a,a+5,cmp);

    for(int i=0;i<5;i++) cout << a[i] << " ";

    cout << "\n";

    return 0;
}
```

例題：Ten Point Round #4 (Div. 2) pB 三個數字比大　小

compare 就可以水過去的題目

備註：~~這一題也可以不要用排序，只是你要寫 11 個 if~~

例題：Ten Point Round #4 (Div. 2) pB 三個數字比大　小

Examples

input	Copy
10 20 30	
output	Copy
20	

input	Copy
7 7 7	
output	Copy
7	

input	Copy
3 5 100	
output	Copy
3	

input	Copy
2 50 7	
output	Copy
50	

input	Copy
1 2 5	
output	Copy
1	

讀入資料後排序

```
19 int main(void)
20 {
21     cin >> a[0] >> a[1] >> a[2];
22
23     sort(a,a+3,cmp);
24
25     cout << a[1] << "\n";
26
27     return 0;
28 }
```

compare 函式

```
11 bool cmp(int a,int b)
12 {
13     if( a % 2 == 0 && b % 2 == 1 ) return true; // 如果 a 是偶 b 是奇 ; 則 a < b
14
15     if( a % 2 == 1 && b % 2 == 0 ) return false; // 如果 a 是奇 b 是偶 ; 則 a > b
16
17     return a < b; // a == b return false → 奇偶性相同，利用原本數字的大小進行比較
18 }
```

練習題

- 輸入一個陣列，將其由大到小排序
- 定義奇數一定比偶數大，其餘的用數字比大小，用這個規則排序一個陣列

Recursion 遞迴

競程圈流傳著一句話：
遞迴只應天上有，凡人應當用迴圈

遞迴

- 還記得我們在使用函式的時候必須呼叫 ㄇ
- 自己呼叫自己的這一個動作，我們就稱之為遞迴

```
 8 int solve(int num)
 9 {
10     if( num ≧ 100 ) return num;
11
12     return solve(num+5);
13 }
```

遞迴的難點

- 搞不清楚遞迴的流程到底是怎麼 跑的
- 我高一剛開始學程式的時候，遞迴卡我卡了一年，我高二才真的 搞懂他
- 接下來我會用盡可能簡單的方式解釋整個流程，讓我們一起領會遞迴的藝術吧~

遞迴其實就是高級版的迴圈

- 如果你要一次使用 100 層的巢狀 for 迴圈，總不可能真的一直複製貼上吧！？
- 如果想要一次使用很多變數，那我們會需要使用陣列
- 那麼如果你現在想要一次使用很多層迴圈，那我們會需要使用遞迴

遞迴寫爛會出現 RE or TLE ？

- 遞迴跟迴圈一樣，終止條件沒寫好會無窮下去，導致 TLE or RE
- 但無限迴圈，只會照乘 TLE，為什麼無限遞迴有可能會出現 RE　呢？
- 因為在遞迴的過程我們是一直往下一層去走，程式內部會有一個 stack 專門去儲存每一層當前的狀態，當你走太深的時候 stack 負荷不了了，就會爆掉，導致 Runtime Error
- 可以把他想像成，你現在有一條線，每遞迴下去一層就要放一些線下去，如果走太深線沒了，就沒辦法繼續下去了，類似的概念xD

遞迴要怎麼停止？ return！

```
 8 void solve(int num)
 9 {
10     if( num == 5 ) // 當 num = 5 的時候接收到 return 讓遞迴停止
11     {
12         return;
13     }
14
15     solve(num-1);
16 }
17 int main()
18 {
19     solve(10);
20 }
```

遞迴範例 1 - 次方

$$a^b = f(a,b) = \begin{cases} 1 & \text{if } b = 0 \\ a * f(a, b-1) & \text{if } b \geq 1 \end{cases}$$

- 如果我們想要計算 a 的 n 次方
- 那我們就必須知道 a 的 n - 1 次方是多少，然後把他乘上 a
- 那我們又必須知道 a 的 n - 2 次方是多少，把他乘上 a 才是 a 的 n - 1 次方...

這樣子一直不斷地重複這一個動作... 因此我們可以使用遞迴來實作

遞迴範例 1 - 次方 code

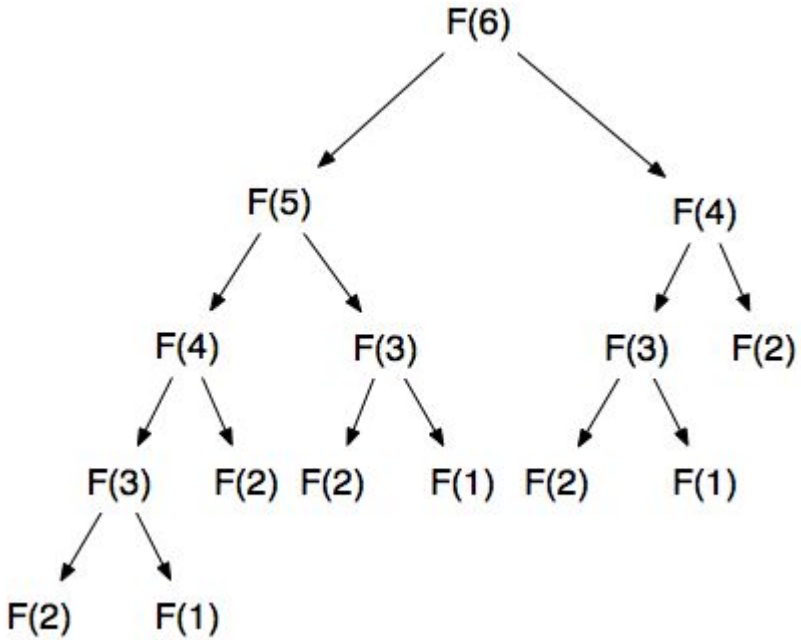
```
 8
 9 int power(int a,int n) // 計算 a 的 n 次方
10 {
11     if( n == 0 ) return 1;
12
13     return power(a,n-1) * a; // 想要知道 a^n 那我們必須先知道 a^(n-1)，所以我們去求 a^(n-1)
14 }
15
16 int main()
17 {
18     printf("%d",power(2,10));
19 }
20
```

稍微複雜一咪咪的例子：費氏數列

- F_0 = 0 遞迴終止條件
- F_1 = 1 遞迴終止條件
- F_i = F_(i-1) + F_(i-2)

$$\begin{cases} F_0 = 0 \\ F_1 = 1 \\ F_n = F_{n-1} + F_{n-2} \end{cases}$$

遞迴樹 (把遞迴的整個流程畫成一個圖)



實作

```
 8
 9 int F(int n) // 求費氏數列第 n 項
10 {
11     if( n == 0 ) return 0;
12     if( n == 1 ) return 1;
13
14     return F(n-1) + F(n-2);
15 }
```

前面的例子大概就是遞迴最簡單的例子

- 因為怕大家吸收太多東西，遞迴就先到這邊
- 更難的遞迴以後就會慢慢遇到了！
- 至少要知道基本的遞迴如何實作即可