遞迴 (Recursion)

遞迴 (Recursion)

- 之前看的 function 都是透過別人來呼叫，那麼如果一個 function 自己呼叫自己會發生什麼事呢？
- 在 C 程式裡面確實允許 function 呼叫自己，這樣的動作稱作遞迴 (recursion)

範例 E09_05.c

```c
#include <stdio.h>
void up_and_down(int);
int main(void)
{
   up_and_down(1);
   return 0;
}
void up_and_down(int n)
{
   printf("Level %d: n location %p\n", n, &n);
   if (n < 4)
      up_and_down(n+1);
   printf("LEVEL %d: n location %p\n", n, &n);
}
```

尾端遞迴 (Tail Recursion)

- 這是最簡單的一種 recursion 型式，因為它運作起來就像迴圈一樣
- 它的特色是在 **`return`** 之前才做 recursive call
- 看看如何用迴圈和遞迴兩種不同方式來計算整數的階乘
- 範例 E09_06.c

Recursion and Reversal

- 什麼情況 recursion 方式比迴圈方式更好寫？
- 寫一個程式能把十進位整數值的二進位表示法顯示出來
- 101 =$1*2^2 + 0*2^1 + 1*2^0$，相當於十進位的 5
- 可以先用十進位來思考。譬如平常寫 32456 其實相當於 3*10000 + 2*1000 + 4*100 + 5*10 + 6
- 要得到個位數是多少，只要用 32456 % 10，就知道個位數是 6。要知道十位數是多少，只要把十位數降成個位數，也就是 32456/10，無條件捨去得到 3245，再用3245%10 就可以取得十位數
- 一直做下去就可以把每個位數都求出來。二進位運算完全一樣，只要把 %10 改成 %2，然後把 /10 改成 /2 就可以
- 範例E09_07.c

遞迴的優缺點

- 以Fibonacci 數列為例
- 1, 1, 2, 3, 5, 8, 13, 21, 34, 55, ...
- 最前面兩個數都是 1，接下來每一個數都是前兩個數的和，所以這個數列本身就是用遞迴方式定義。可以用底下的 function 來表示：

```
long Fibonacci(int n)
{
   if (n > 2)
      return Fibonacci(n-1) + Fibonacci(n-2);
   else
      return 1;
}
```