

程式設計 (一)

CH5. 迴圈控制

Ming-Hung Wang 王銘宏

tonymhwang@cs.ccu.edu.tw

Department of Computer Science and Information Engineering
National Chung Cheng University

Last Semester, 2021

本章目錄

1. 指派運算子
2. 遞增遞減運算子
3. while 迴圈敘述式
4. 「計數器控制的迴圈」與 for 迴圈敘述式
5. 「警示值控制的迴圈」與 do while 迴圈敘述式
6. break、continue 敘述式
7. 迴圈應用：輾轉相除法
8. 巢狀迴圈
9. scanf 回傳值：EOF (End Of File)

指派運算子 (賦值運算子)

指派運算子「=」其實我們早在之前就已經一直在使用了，不過這次要介紹的是指派運算子的變化。

指派運算子

在寫程式的情況，我們很常會遇到將某變數加上多少的情況，這時候我們會用到這樣的敘述：

```
val = val + n;
```

不過因為這樣寫久了實在很冗長，所以工程師就設計了運算子「+=」，代表「將自己加上...」。原本的式子可以改成：

```
val += n;
```

指派運算子

不只加法，減、乘、除、取餘數
也都可以與「=」結合：

運算子	範例	等價於
$+=$	$a += b$	$a = a + b$
$-=$	$a -= b$	$a = a - b$
$*=$	$a *= b$	$a = a * b$
$/=$	$a /= b$	$a = a / b$
$\%=$	$a \% = b$	$a = a \% b$

遞增遞減運算子

在寫程式時，我們很常會用到「 $+ = 1$ 」或「 $- = 1$ 」的情況，所以工程師又發明了新的運算子「 $++$ 」與「 $--$ 」來代表加 1 與減 1。

遞增遞減運算子

遞增運算子 (++) 與遞減運算子 (--) 個別分為
前置遞增 (preincrement)、後置遞增 (postincrement) 與
前置遞減 (predecrement)、後置遞減 (postdecrement)。

運算式	說明
$++a$	先將 a 加 1，再以 a 的新值進行運算
$a++$	以 a 目前的值進行運算，再將 a 加 1
$--a$	先將 a 減 1，再以 a 的新值進行運算
$a--$	以 a 目前的值進行運算，再將 a 減 1

遞增遞減運算子

下面這兩串程式碼效果是相同的：

```
1 // increment & decrement
2 #include <stdio.h>
3
4 int main(){
5     int a = 3;
6     int b = 5;
7     int c = a++;
8     int d = --b;
9     printf("%d\n", ++c);
10    printf("%d\n", d--);
11 }
```

"D:\codeblocks\increment & decrement.exe"

4
4

```
1 // increment & decrement
2 #include <stdio.h>
3
4 int main(){
5     int a = 3;
6     int b = 5;
7     int c = a;
8     a += 1;
9     b -= 1;
10    int d = b;
11    c += 1;
12    printf("%d\n", c);
13    printf("%d\n", d);
14    d -= 1;
15 }
```

"D:\codeblocks\increment & decrement.exe"

4
4

遞增遞減運算子

若只要單獨做遞增或遞減的動作，
下面 4 種方式作用都是相同的。

```
a = a + 1;  
b = b - 1;
```

```
a += 1;  
b -= 1;
```

```
a++;  
b--;
```

```
++a;  
--b;
```

遞增遞減運算子

運算優先度：

	運算子	結合性	形式
最優先	++ (後置) -- (後置)	由左至右	後置
	+ (正號) - (負號) ++ (前置) -- (前置)	由右至左	單元性
	* (乘以) / (除以) % (取餘數)	由左至右	乘法
	+ (加法) - (減法)	由左至右	加法
	< <= > >=	由左至右	關係
	== !=	由左至右	相等
	&&	由左至右	邏輯AND
		由左至右	邏輯OR
	?: (三元運算子)	由右至左	條件
最不優先	= += -= *= /=	由右至左	設置

while 迴圈敘述式

while 迴圈敘述式

如果 if 是判斷並執行 1 次，
那 while 就是一直判斷並執行直到不成立為止

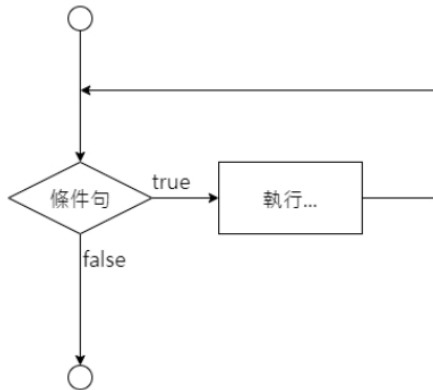
中文裡的句型為：
當 (條件) 還成立時，就持續 (執行...)

```
while (條件句) {  
    敘述句1;  
    敘述句2;  
    .....  
}
```

while 迴圈敘述式

```
while (條件句) {  
    ...  
}
```

while 迴圈流程圖：



while 迴圈敘述式

while 迴圈範例：

```
1 //while loop
2 #include <stdio.h>
3
4 int main(){
5     int product = 4;
6     while(product < 100){
7         printf("%d\n", product);
8         product *= 4;
9     }
10    printf("leaved while loop\n");
11    printf("%d\n", product);
12 }
13
14 "D:\codeblocks\while loop.exe"
4
16
64
leaved while loop
256
```

在設計迴圈時，常用的技巧

- 計數器控制的迴圈：使用 counter 來記錄目前是第幾次迴圈，或紀錄已經達成了幾次指定條件，並在 counter 到達某數值時結束迴圈。
- 警示器控制的迴圈：利用警示值 (sentinel value 或稱旗標值 flag value) 來結束迴圈。
例如：重複輸入成績時，要求輸入不可能為成績的數值 (假設為-1) 時，代表使用者輸入完畢。

while 迴圈敘述式

計數器控制的迴圈

```
1 //while loop
2 #include <stdio.h>
3
4 int main(){
5     int grade = 0;
6     int inputCounter = 0;
7     int passCounter = 0;
8     printf("輸入10個成績\n");
9     while(inputCounter < 10){
10         printf("輸入成績: ");
11         scanf("%d", &grade);
12         inputCounter++;
13         if(grade >= 60){
14             passCounter++;
15         }
16     }
17     printf("總共%d人及格\n", passCounter);
18 }
```

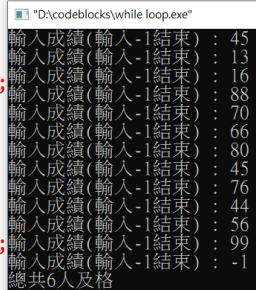
"D:\codeblocks\while loc

```
輸入10個成績
輸入成績: 80
輸入成績: 77
輸入成績: 50
輸入成績: 66
輸入成績: 58
輸入成績: 65
輸入成績: 46
輸入成績: 63
輸入成績: 49
輸入成績: 50
總共5人及格
```


while 迴圈敘述式

警示訊號控制的迴圈

```
1 //while loop
2 #include <stdio.h>
3
4 int main(){
5     int grade = 0;
6     int passCounter = 0;
7     while(grade != -1){
8         printf("輸入成績(輸入-1結束) : ");
9         scanf("%d", &grade);
10        if (grade >= 60){
11            passCounter++;
12        }
13    }
14    printf("總共%d人及格\n", passCounter);
15 }
```



```
"D:\codeblocks\while loop.exe"
輸入成績(輸入-1結束) : 45
輸入成績(輸入-1結束) : 13
輸入成績(輸入-1結束) : 16
輸入成績(輸入-1結束) : 88
輸入成績(輸入-1結束) : 70
輸入成績(輸入-1結束) : 66
輸入成績(輸入-1結束) : 80
輸入成績(輸入-1結束) : 45
輸入成績(輸入-1結束) : 76
輸入成績(輸入-1結束) : 44
輸入成績(輸入-1結束) : 56
輸入成績(輸入-1結束) : 99
輸入成績(輸入-1結束) : -1
總共6人及格
```

「計數器控制的迴圈」與 for 迴圈敘述式

計數器控制的迴圈與 for 迴圈敘述式

我們已經知道的 2 種迴圈方法：

1. 計數器控制的迴圈
2. 警示值控制的迴圈

計數器控制的迴圈，在迴圈開始前就已經知道重複的次數了，所以被稱為「明確的重複」。而警示值控制的迴圈，則被稱為「非明確的重複」，因為我們事先並不知道迴圈會重複多少次。

計數器控制的迴圈

計數器控制的迴圈需要有：

1. 計數器變數 (控制變數) 的名稱
2. 計數器變數的初始值
3. 每一次重複時計數器變數的遞增或遞減量
4. 計數器變數的中止值

計數器控制的迴圈與 for 迴圈敘述式

一個簡單的計數器控制的迴圈：

```
1 //counter-controlled repetition & for loop
2 #include <stdio.h>
3
4 int main(){
5     int counter = 0; //計數器變數名稱&初始值
6     while (counter < 10){ //計數器變數的中止值
7         printf("%d ", counter);
8         counter++; //計數器變數的遞增或遞減量
9     }
10    printf("\nleaved loop\n");
11    printf("%d\n", counter);
12 }
13 "D:\codeblocks\counter-controlled repetition & for loop.exe"
```

```
0 1 2 3 4 5 6 7 8 9
leaved loop
10
```

for 迴圈敘述式

```
for (初始敘述句; 條件句; 循環敘述句) {  
    ...  
}
```

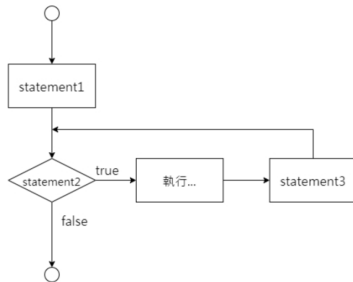
for 迴圈括號中的 3 個敘述句稱為標頭運算式。
使用 for 迴圈可以非常明確的表示計數器控制的
初始值、中止值與遞增或遞減量。

計數器控制的迴圈與 for 迴圈敘述式

for 與 while 比較與流程圖

```
for (statement1; statement2; statement3) {  
    statements  
    //...  
}
```

```
statement1;  
while (statement2) {  
    statements  
    //...  
    statement3;  
}
```



計數器控制的迴圈與 for 迴圈敘述式

使用 for 敘述式的一個簡單的計數器控制的迴圈

```
1 //counter-controlled repetition & for loop
2 #include <stdio.h>
3
4 int main(){
5     int counter;
6     for (counter = 0; counter < 10; counter++) {
7         printf("%d ", counter);
8     }
9     printf("\nleaved loop\n");
10    printf("%d\n", counter);
11 }
12 "D:\codeblocks\counter-controlled repetition & for loop.exe"
```

```
0 1 2 3 4 5 6 7 8 9
leaved loop
10
```


計數器控制的迴圈與 for 迴圈敘述式

在 for 敘述式的初始值宣告變數

於版本 C99 以後，for 敘述式的初始處可以宣告變數，但在此宣告的變數不能在此 for 迴圈外面使用。

```
1 //counter-controlled repetition & for loop
2 #include <stdio.h>
3
4 int main(){
5     for (int i = 0; i < 20; i++){
6         if(!(i % 3)) //判斷3的倍數
7             printf("%d ", i);
8     }
9     //離開for迴圈，已不能使用變數i
10    printf("\n");
11 }
12
```

"D:\codeblocks\counter-controlled repetition & for loop.exe"

0 3 6 9 12 15 18

計數器控制的迴圈與 for 迴圈敘述式

for 迴圈的標頭運算式是可有可無的

```
for (初始敘述句; 條件句; 循環敘述句) {  
    ...  
}
```

- 如果計數器變數 (控制變數) 已經在之前就設定好變數，則可以將初始敘述句省略。
- 如果不需要遞增或遞減，或已經在迴圈本體中完成，則可以省略循環敘述句。
- 如果省略條件俱，編譯器會認為控制條件永遠為 true，而形成無窮迴圈。

另外，雖然標頭運算式都可以省略，但是分隔運算式用的分號 (;) 是不能省略的。

應用：列出每年複利

若某人將 1000 元存入年利率 5% 的帳戶裡，
請列出 10 年內每年結算時帳戶的錢是多少。

```
D:\codeblocks\Practice.exe  
Year: 1, deposit: 1050.00  
Year: 2, deposit: 1102.50  
Year: 3, deposit: 1157.63  
Year: 4, deposit: 1215.51  
Year: 5, deposit: 1276.28  
Year: 6, deposit: 1340.10  
Year: 7, deposit: 1407.10  
Year: 8, deposit: 1477.46  
Year: 9, deposit: 1551.33  
Year: 10, deposit: 1628.89
```

計數器控制的迴圈與 for 迴圈敘述式

範例程式碼 1

```
1 //calculating compound interest
2 #include <stdio.h>
3
4 int main(){
5     double deposit = 1000;
6     double rate = 0.05;
7     for (int year = 1; year <= 10; year++) {
8         deposit *= 1.0 + rate;
9         printf("Year: %2d, deposit: %.2f\n", year, deposit);
10    }
11 }
```

計數器控制的迴圈與 for 迴圈敘述式

範例程式碼 2

使用 math.h 中的 pow 函式計算次方

```
double pow (double base, double exponent);
```

```
1 //calculating compound interest
2 #include <stdio.h>
3 #include <math.h>
4
5 int main(){
6     double starDeposit = 1000;
7     double rate = 0.05;
8     for (int year = 1; year <= 10; year++) {
9         double deposit = starDeposit * pow(1.0 + rate, year);
10        printf("Year: %2d, deposit: %.2f\n", year, deposit);
11    }
12 }
```

誤差為 1 的錯誤 (off-by-one error)

如果要讓迴圈執行 10 次，for 迴圈的標頭可寫成：

```
for (int i = 0; i < 10; i++)
```

或寫成：

```
for (int i = 1; i <= 10; i++)
```

如果判斷式寫錯的話，可能會造成誤差為 1 的錯誤，例如：

```
for (int i = 0; i <= 10; i++) //執行11次
```

```
for (int i = 1; i < 10; i++) //執行9次
```

「警示值控制的迴圈」與 do while 迴圈敘述式

警示值控制的迴圈

設計警示值 (或稱旗標值 flag value) 的方式:

1. 在一般變數 (例如儲存成績的變數或儲存搜尋結果的變數等) 設定一個功能上不可能出現的值作為旗標值, 例如 -1 或 999...
2. 宣告一個旗標變數, 作為目前狀態的標誌, 旗標變數通常是 bool 或 int 型態。

警示值控制的迴圈與 do while 迴圈敘述式

計算平均成績，設定警示值 -1 結束輸出

```
1 // sentinel-controlled repetition
2 #include<stdio.h>
3
4 int main(){
5     int grade = 0, sum = 0, counter = 0;
6     double avg;
7     while(grade != -1){
8         printf("輸入成績(輸入-1結束) : ");
9         scanf("%d", &grade);
10        if (grade != -1){
11            sum += grade;
12            counter++;
13        }
14    }
15    avg = (double)sum / counter;
16    printf("平均%.2f分\n", avg);
17 }
```

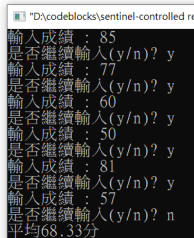


```
"D:\codeblocks\sentinel-controlled repetition.exe"
輸入成績(輸入-1結束) : 88
輸入成績(輸入-1結束) : 75
輸入成績(輸入-1結束) : 63
輸入成績(輸入-1結束) : 45
輸入成績(輸入-1結束) : -1
平均67.75分
```

警示值控制的迴圈與 do while 迴圈敘述式

讓使用者選擇是否繼續輸入，並使用旗標變數紀錄

```
1 // sentinel-controlled repetition
2 #include<stdio.h>
3
4 int main(){
5     int input = 0, sum = 0, counter = 0;
6     int flag = 1;
7     double avg;
8     while(flag){
9         printf("輸入成績 : ");
10        scanf("%d", &input);
11        sum += input;
12        counter++;
13        printf("是否繼續輸入(y/n)? ");
14        scanf("%c%c", &input);
15        flag = (input == 'y') ? 1 : 0;
16    }
17    avg = (double)sum / counter;
18    printf("平均%.2f分\n", avg);
19 }
```



```
"D:\codeblocks\sentinel-controlled re
輸入成績 : 85
是否繼續輸入(y/n)? y
輸入成績 : 77
是否繼續輸入(y/n)? y
輸入成績 : 60
是否繼續輸入(y/n)? y
輸入成績 : 50
是否繼續輸入(y/n)? y
輸入成績 : 81
是否繼續輸入(y/n)? y
輸入成績 : 57
是否繼續輸入(y/n)? n
平均68.33分
```

警示值控制的迴圈與 do while 迴圈敘述式

在上述 2 個例子裡，while 判斷句所判斷的 grade 與 flag 變數，必須在迴圈前事先指派數值。因為一個沒有初始化的變數其值會是保留在該記憶體位置中的「垃圾值 (garbage value)」。

如果使用 do while 迴圈，就可以不用在迴圈之前先指派預設值給那些要被判斷的變數。

do while 迴圈敘述式

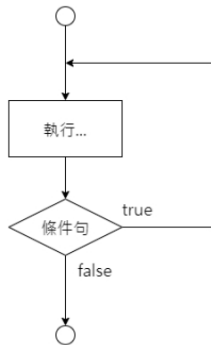
do while 敘述式十分類似於 while 敘述式。while 敘述式的迴圈繼續條件是在迴圈一開始檢驗，而 do while 迴圈則是在每次迴圈 之後 才檢驗。

```
do {  
    ...  
} while (條件句);
```

注意，在 while 的括號後面有一個分號。

警示值控制的迴圈與 do while 迴圈敘述式

do while 迴圈流程圖：

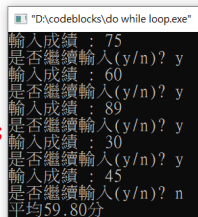


就結果上來說，while 迴圈與 do while 迴圈的差別只在第一次迴圈開始之前是否有進行條件判斷。

警示值控制的迴圈與 do while 迴圈敘述式

讓使用者選擇是否繼續輸入，使用 do while 迴圈

```
1 // do while loop
2 #include <stdio.h>
3
4 int main(){
5     int input = 0, sum = 0, counter = 0;
6     double avg;
7     do{
8         printf("輸入成績 : ");
9         scanf("%d", &input);
10        sum += input;
11        counter++;
12        printf("是否繼續輸入(y/n)? ");
13        scanf("%c%c", &input);
14    }while(input == 'y' || input == 'Y');
15    avg = (double)sum / counter;
16    printf("平均%.2f分\n", avg);
17 }
```



```
"D:\codeblocks\do while loop.exe"
輸入成績 : 75
是否繼續輸入(y/n)? y
輸入成績 : 60
是否繼續輸入(y/n)? y
輸入成績 : 89
是否繼續輸入(y/n)? y
輸入成績 : 30
是否繼續輸入(y/n)? y
輸入成績 : 45
是否繼續輸入(y/n)? n
平均59.80分
```

break、continue 敘述式

break 敘述式

當 break 敘述式

在 while、for、do while 或 switch 敘述式內執行時，
會使得程式馬上離開那個敘述式。

break、continue 敘述式

break 敘述式範例

```
1 // break & countinue
2 #include <stdio.h>
3
4 int main(){
5     for (int i = 0; i < 10; i++){
6         if (i == 4){
7             break; //跳出迴圈
8         }
9         printf("%d ", i);
10    }
11    printf("\nexited loop\n");
12 }
13
```


"D:\codeblocks\break & countinue.exe"

0 1 2 3
exited loop

break、continue 敘述式

若有兩層以上的迴圈，則只會離開最裡面的一層：

```
1 // break & continue
2 #include <stdio.h>
3
4 int main(){
5     for (int i = 0; i < 5; i++){
6         for(int j = 0; j < 5; j++){
7             if (i == j){
8                 break; //跳出最內部的迴圈
9             }
10            printf("%d ", j);
11        }
12        printf("\n");
13    }
14 }
```



The terminal window shows the output of the program. It displays four lines of numbers: "0", "0 1", "0 1 2", and "0 1 2 3". This is because the inner loop prints values of j from 0 to i-1 for each i from 0 to 4. The break statement ensures that for each i, only the values of j less than i are printed, and then a new line is started.

continue 敘述式

當 continue 敘述式
在 while、for 或 do while 敘述式中執行時，
敘述式本體內尚未執行的敘述式會跳過，
而直接執行下一次的迴圈動作。

break、continue 敘述式

- 在 while 和 do while 敘述式中，迴圈繼續條件會在 continue 執行之後馬上檢驗。
- 在 for 敘述式中，會先執行遞增運算式，然後再檢驗迴圈繼續條件。

break、continue 敘述式

continue 敘述式範例

```
1 // break & countinue
2 #include <stdio.h>
3
4 int main(){
5     int num = 3;
6     for (int i = 1; i <= 10; i++){
7         if (i % num){
8             continue; //直接進入下個迴圈
9         }
10        printf("%d是%d的倍數\n", i , num);
11    }
12 }
13
```

"D:\codeblocks\break & countinue.exe"

3是3的倍數
6是3的倍數
9是3的倍數

迴圈應用：輾轉相除法

輾轉相除法是用來計算兩整數最大公因數的演算法，可以使用迴圈控制來達成。

迴圈應用：輾轉相除法

輾轉相除法的運算方式是：

1. 將兩整數代入到 $r = a \% b$ 的 a 與 b 中，求出 r
2. 將上一步的 b 與 r 的值重新代入 $r = a \% b$ 的 a 與 b 中，求出新的 r
3. 重複步驟 2，直到 r 等於 0
4. 最後的 b 即是兩整數的最大公因數

迴圈應用：輾轉相除法

以下是使用輾轉相除法計算 98 與 35 的
最大公因數的數學運算過程：

$$r_0 = 98 \% 35, r_0 = 28$$


$$r_1 = 35 \% 28, r_1 = 7$$

$$r_2 = 28 \% 7, r_2 = 0$$

\Rightarrow 98 與 35 的最大公因數為 7

迴圈應用：輾轉相除法

請試著設計程式以實現輾轉相除法

 D:\codeblocks\Practice.exe

```
Input 2 ints: 98 35  
a: 98 b: 35 r: 28  
a: 35 b: 28 r: 7  
a: 28 b: 7 r: 0  
=> GCD: 7
```

迴圈應用：輾轉相除法

參考解答

```
1 //Euclidean algorithm
2 #include <stdio.h>
3
4 int main(){
5     int a, b, r;
6     printf("Input 2 ints: ");
7     scanf("%d%d", &b, &r);
8     while(r){
9         a = b; //將上一次的b代入a
10        b = r; //將上一次算出的r代入b
11        r = a % b;
12        printf("a: %2d  b: %2d  r: %2d\n", a, b, r);
13    }
14    printf("=> GCD: %d\n", b);
15 }
```

巢狀迴圈

一層包著一層的迴圈

巢狀迴圈

由兩層 for 組成的巢狀迴圈

```
1 // Nested loop
2 #include <stdio.h>
3
4 int main(){
5     for (int i = 1; i <= 5; i++){
6         for (int j = 0; j < i; j++){
7             printf("*");
8         }
9         printf("\n");
10    }
11 }
12
```

"D:\codeblocks\Nested loop.exe"


```
*
**
***
****
*****
```

巢狀迴圈

- 巢狀迴圈若使用多層的 for 迴圈，則
第一層的計數器用 i
第二層的計數器用 j
第三層的計數器用 k
第四層的計數器用 m (盡量不要用到第四層以上)
- 當然，若能使用有意義的英文單字來取名就再好不過了。

練習

使用巢狀迴圈來印出乘法表吧

 "D:\codeblocks\Nested loop.exe"

```
Input 2 ints: 5 3
1 * 1 = 1    2 * 1 = 2    3 * 1 = 3
1 * 2 = 2    2 * 2 = 4    3 * 2 = 6
1 * 3 = 3    2 * 3 = 6    3 * 3 = 9
1 * 4 = 4    2 * 4 = 8    3 * 4 = 12
1 * 5 = 5    2 * 5 = 10   3 * 5 = 15
```

巢狀迴圈

練習

使用巢狀迴圈來印出漏斗圖形吧

 "D:\codeblocks\Nested loop.exe"

```
Input num: 4
*****
 *
  *
   *
  *
 *
*****
*****
```

scanf 回傳值： EOF (End Of File)

scanf 回傳值：EOF (End Of File)

scanf 與 printf 在執行完畢之後，都會回傳一個整數：

- printf 會回傳印出的字元數
- scanf 會回傳成功讀入的數值個數

```
1 // printf & scanf return value
2 #include <stdio.h>
3
4 int main(){
5     int n, m, p, q;
6     n = printf("ABCDE\n"); //輸出的字元數
7     printf("printf return: %d\n", n);
8     n = scanf("%d, %d, %d", &m, &p, &q); //成功輸入的個數
9     printf("scanf return: %d\n", n);
10 }
11
```

"D:\codeblocks\printf & scanf return value.exe"

```
ABCDE
printf return: 6
125, 456, 98
scanf return: 3
```

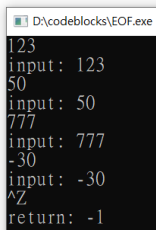
scanf 回傳值：EOF (End Of File)

當 scanf 讀到檔案結尾時，會回傳一個特殊常數
EOF (數值為 -1)
如果使用鍵盤輸入，
在 Linux/Unix 系統的 ctrl+d 組合鍵代表檔案結尾，
而 Windows 系統則以 ctrl+z 組合鍵代表檔案結尾。

scanf 回傳值：EOF (End Of File)

重複輸入直到 EOF 為止

```
1 // EOF
2 #include <stdio.h>
3
4 int main(){
5     int val, r;
6     do{
7         r = scanf("%d", &val);
8         if (r == EOF)
9             printf("return: %d\n", r);
10        else
11            printf("input: %d\n", val);
12    }while (r != EOF);
13 }
```

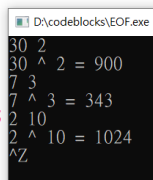


```
D:\codeblocks\EOF.exe
123
input: 123
50
input: 50
777
input: 777
-30
input: -30
^Z
return: -1
```

scanf 回傳值：EOF (End Of File)

重複輸入兩整數並計算次方，直到 EOF 為止

```
1 // EOF
2 #include <stdio.h>
3 #include <math.h>
4
5 int main(){
6     int base, exp, power;
7     while(scanf("%d%d", &base, &exp) != EOF){
8         power = round(pow(base, exp));
9         printf("%d ^ %d = %d\n", base, exp, power);
10    }
11 }
```



```
D:\codeblocks\EOF.exe
30 2
30 ^ 2 = 900
7 3
7 ^ 3 = 343
2 10
2 ^ 10 = 1024
^Z
```

- 備註：

pow 與 round 為 math.h 所提供的函式

pow 是次方運算，round 是四捨五入

使用 Windows CMD 編譯與執行

指令：

- cd 「資料夾位置」
=> 將所在位置移動到指定位置
- gcc 「欲編譯的檔名」 -o 「欲輸出的檔名」
=> 使用 gcc 編譯成執行檔
- 「執行檔檔名」
=> 執行執行檔

scanf 回傳值：EOF (End Of File)

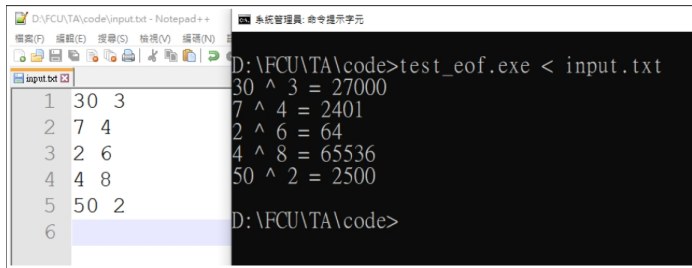
將計算次方的程式碼使用 CMD 編譯與執行



```
系統管理員: 命令提示字元
C:\Windows\System32>D:
D:\FCU\TA\code>cd D:\FCU\TA\code
D:\FCU\TA\code>gcc test_eof.c -o test_eof.exe
D:\FCU\TA\code>test_eof.exe
20 3
20 ^ 3 = 8000
7 3
7 ^ 3 = 343
2 12
2 ^ 12 = 4096
^Z
D:\FCU\TA\code>
```

scanf 回傳值：EOF (End Of File)

將文字檔作為標準輸入 (EOF 即該檔案的結尾)

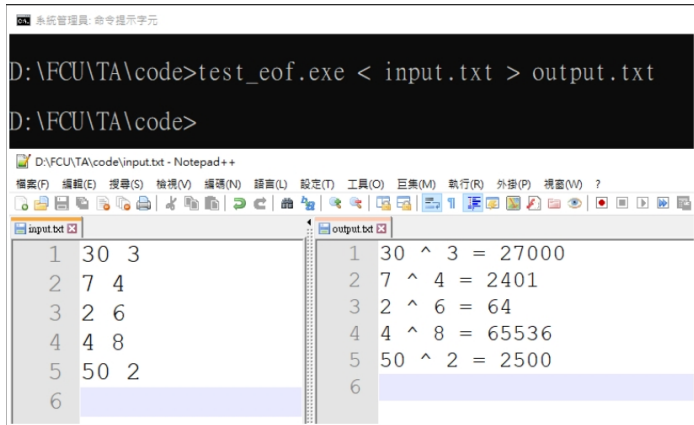


The screenshot displays two windows side-by-side. The left window is Notepad++ editing 'input.txt', which contains a list of numbers: 1 30 3, 2 7 4, 3 2 6, 4 4 8, 5 50 2, and 6. The right window is the Windows Command Prompt, showing the execution of 'test_eof.exe' with 'input.txt' as input. The program outputs five lines of calculations: 30 ^ 3 = 27000, 7 ^ 4 = 2401, 2 ^ 6 = 64, 4 ^ 8 = 65536, and 50 ^ 2 = 2500. The prompt then returns to 'D:\FCU\TA\code>'.

```
D:\FCU\TA\code>test_eof.exe < input.txt
30 ^ 3 = 27000
7 ^ 4 = 2401
2 ^ 6 = 64
4 ^ 8 = 65536
50 ^ 2 = 2500
D:\FCU\TA\code>
```

scanf 回傳值：EOF (End Of File)

將標準輸出設為文字檔 (輸出就不會顯示到螢幕上)



The screenshot shows a Windows command prompt window titled "系統管理員: 命令提示字元" (System Administrator: Command Prompt) with the command `D:\FCU\TA\code>test_eof.exe < input.txt > output.txt` entered. Below the command prompt, there are two Notepad++ windows. The left window, titled "D:\FCU\TA\code\input.txt - Notepad++", shows the contents of input.txt: a list of numbers 1 through 6, each followed by two space-separated integers. The right window, titled "output.txt", shows the output of the program: the same numbers 1 through 6, each followed by a mathematical expression and its result. The expressions are $30^3 = 27000$, $7^4 = 2401$, $2^6 = 64$, $4^8 = 65536$, and $50^2 = 2500$. The last line for number 6 is empty.

input.txt	output.txt
1 30 3	1 $30^3 = 27000$
2 7 4	2 $7^4 = 2401$
3 2 6	3 $2^6 = 64$
4 4 8	4 $4^8 = 65536$
5 50 2	5 $50^2 = 2500$
6	6

參考資料： Deitel, H. M., & Deitel, P. J. (2015). C: How to program.
Upper Saddle River, N.J: Prentice Hall.