程式設計-第五章-函式

洪麗玲

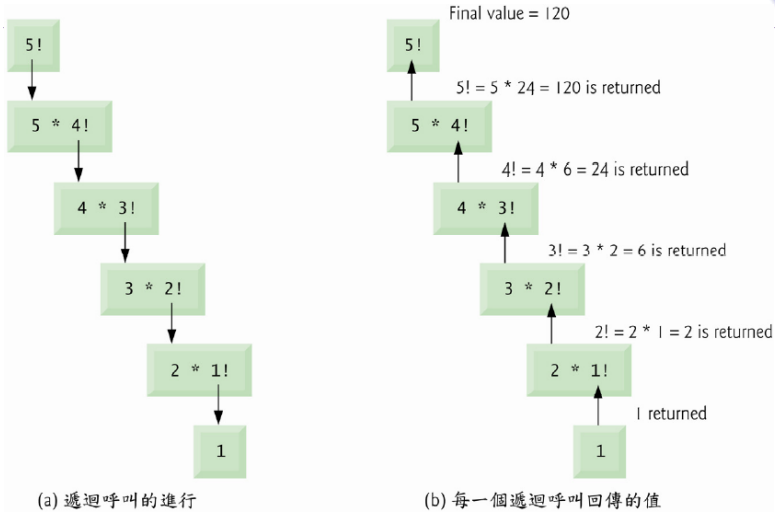
5.14 遞迴

- **遞迴函式 (recursive function)** 就是一種可以直接或間接呼叫自己的函式。遞迴是進階電腦科學課程中所討論的一項複雜的課題。
- 用遞迴方法計算階乘
  - 非負整數的階乘n，寫成$n!$ (讀作「n階乘」) 如以下乘積：

$$n \cdot (n\ 1) \cdot (\ n\ 2) \cdot \cdot 1$$

的乘積，而1! 等於1，0! 也定義成1

- 5! 的計算可以如圖5.17所示執行。



(a) 遞迴呼叫的進行　(b) 每一個遞迴呼叫回傳的值


```
0! = 1
1! = 1
2! = 2
3! = 6
4! = 24
5! = 120
6! = 720
7! = 5040
8! = 40320
9! = 362880
10! = 3628800
11! = 39916800
12! = 479001600
13! = 6227020800
14! = 87178291200
15! = 1307674368000
16! = 20922789888000
17! = 355687428096000
18! = 6402373705728000
19! = 121645100408832000
20! = 2432902008176640000
21! = 14197454024290336768
```

圖5.18　以遞迴函式計算階乘

- 圖5.18的程式利用遞迴來計算並印出0到10之整數的階乘值。

```
1  // Fig. 5.18: fig05_18.c
2  // Recursive factorial function.
3  #include <stdio.h>
4
5  unsigned long long int factorial( unsigned int number );
6
7  // function main begins program execution
8  int main( void )
9  {
10    unsigned int i; // counter
11
12    // during each iteration, calculate
13    // factorial( i ) and display result
14    for ( i = 0; i <= 21; ++i ) {
15       printf( "%u! = %llu\n", i, factorial( i ) );
16    } // end for
17 } // end main
18
19 // recursive definition of function factorial
20 unsigned long long int factorial( unsigned int number )
21 {
22    // base case
23    if ( number <= 1 ) {
24       return 1;
25    } // end if
```


```
14    for ( i = 0; i <= 21; ++i ) {
15       printf( "%u! = %llu\n", i, factorial( i ) );
16    } // end for
17 } // end main
18
19 // recursive definition of function factorial
20 unsigned long long int factorial( unsigned int number )
21 {
22    // base case
23    if ( number <= 1 ) {
24       return 1;
25    } // end if
26    else { // recursive step
27       return ( number * factorial( number - 1 ) );
28    } // end else
29 } // end function factorial
```

圖5.18　以遞迴函式計算階乘(2/3)

練習

- 請撰寫一支程式功能可列出費氏函數：
  - 使用者輸入一個數字
  - 系統輸出該數的費氏函數
  - 可以輸入很多次

```
fibonacci(0) = 0
fibonacci(1) = 1
fibonacci(n) = fibonacci(n  1) + fibonacci(  n  2)
```

```
0, 1, 1, 2, 3, 5, 8, 13, 21,
```

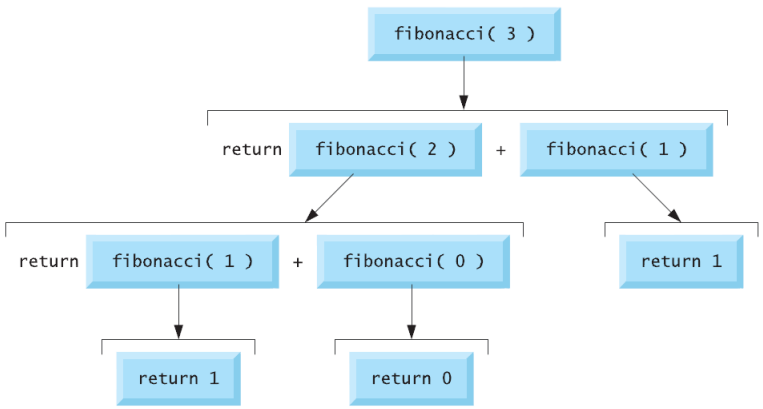



圖5.20　呼叫**fibonacci(3)**的遞迴呼叫

```
Enter an integer: 0
Fibonacci( 0 ) = 0
```

```
Enter an integer: 1
Fibonacci( 1 ) = 1
```

```
Enter an integer: 2
Fibonacci( 2 ) = 1
```

```
Enter an integer: 3
Fibonacci( 3 ) = 2
```

```
Enter an integer: 10
Fibonacci( 10 ) = 55
```

```
Enter an integer: 20
Fibonacci( 20 ) = 6765
```

```
Enter an integer: 30
Fibonacci( 30 ) = 832040
```

```
Enter an integer: 40
Fibonacci( 40 ) = 102334155
```

圖5.19　遞迴產生Fibonacci數範例

```
1  // Fig. 5.19: fig05_19.c
2  // Recursive fibonacci function
3  #include <stdio.h>
4
5  unsigned long long int fibonacci( unsigned int n ); // function prototype
6
7  // function main begins program execution
8  int main( void )
9  {
10    unsigned long long int result; // fibonacci value
11    unsigned int number; // number input by user
12
13    // obtain integer from user
14    printf( "%s", "Enter an integer: " );
15    scanf( "%u", &number );
16
17    // calculate fibonacci value for number input by user
18    result = fibonacci( number );
19
20    // display result
21    printf( "Fibonacci( %u ) = %llu\n", number, result );
22 } // end main
23
```

```
24 // Recursive definition of function fibonacci
25 unsigned long long int fibonacci( unsigned int n )
26 {
27    // base case
28    if ( 0 == n || 1 == n ) {
29       return n;
30    } // end if
31    else { // recursive step
32       return fibonacci( n - 1 ) + fibonacci( n - 2 );
33    } // end else
34 } // end function fibonacci
```


遞迴與迴圈

- 都含有重複性:
  - 迴圈使用重複描述;
  - 遞迴使用重複的函式呼叫
- 都有終止檢測:
  - 迴圈繼續條件失敗時
  - 遞迴到達基本情況時
- 重複控制
  - 迴圈會持續改變計數器的值直到不符合迴圈持續的條件為止;
  - 遞迴則持續將原來的問題簡化，直到問題變成基本狀況為止。
- 都可能產生無窮迴圈:
  - 當條件永遠符合的時候
  - 當問題無法收斂到基本狀況時

遞迴思考練習

- 請以遞迴完成下列程式要求
- 主程式要求使用者給定一個變數，呼叫函式計算
- 函式回傳從1累加到該數的總和

例如:輸入10

則輸出 55

- 提示:以前使用迴圈，現在改以遞迴呼叫函式的方式