

CH04 函式、遞迴

Function & Recursion

日期：2014/11/16

主講人：陳靖德

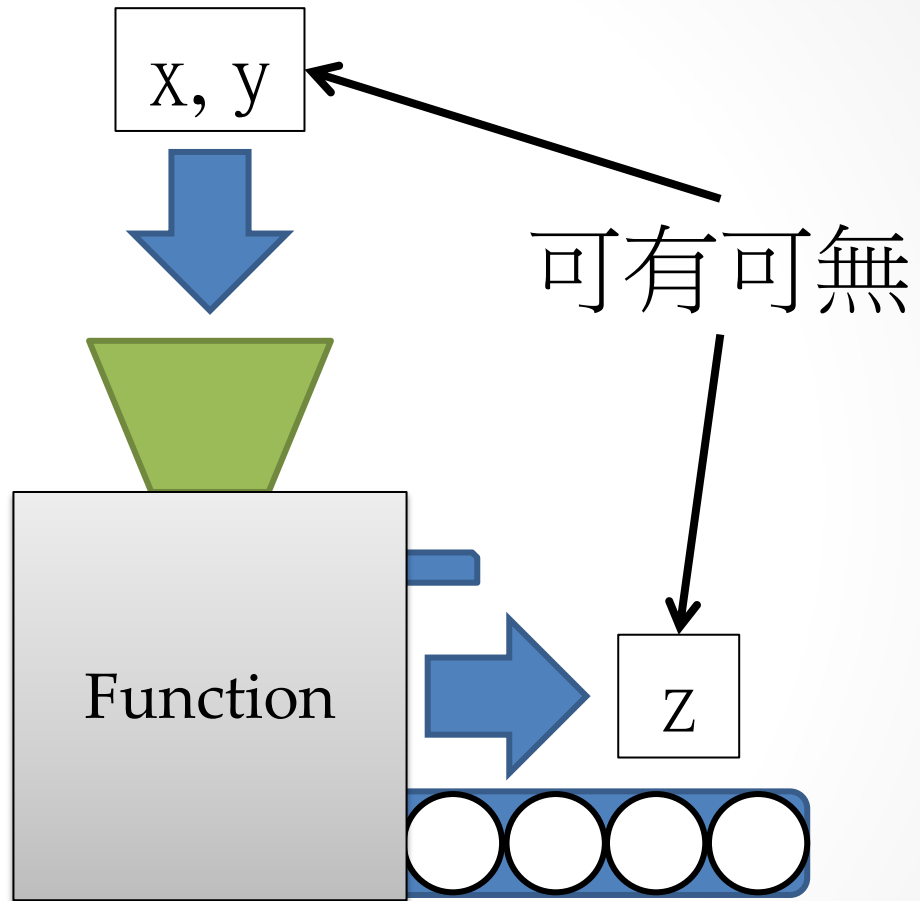
函数

...

Function

函式

- 概念



使用

```
function.cpp
1  #include <stdio.h>
2  #include <math.h>
3
4  int main() {
5
6      double a = pow(5, 2);
7      return 0;
8  }
```

參數

$a = 5^2$

呼叫pow

double _cdecl pow (double _X, double _Y) - math.h (151) - Ctrl+Click for

註：要使用pow前，記得 include <math.h>

練習一

- 輸入：a、b兩個整數
輸出： a^1 到 a^b
格式請參考右圖
(不用對齊)
- 請使用pow函式

```
2 10
2 ^ 1 =      2
2 ^ 2 =      4
2 ^ 3 =      8
2 ^ 4 =     16
2 ^ 5 =     32
2 ^ 6 =     64
2 ^ 7 =    128
2 ^ 8 =    256
2 ^ 9 =    512
2 ^ 10 =   1024
請按任意鍵繼續 . . .
```

小提示：pow回傳的是double
(順帶一提，C語言的^是xor運算)

函式優點

- 可重複利用(模組化)
- 提高可讀性(抽象化)

```
double z = pow(5, 2);
```

```
double z = a;  
int i;  
for (i = 1; i < b; i++) {  
    z *= a;  
}
```

註：其實pow可以使用小數點，右邊的版本不行

自定義函式

function.cpp

```
1  #include <stdio.h>
2
3  int max(int a, int b);  函式原型
4
5  int main() {
6      |
7      |   return 0;
8      |
9      |
10 int max(int a, int b) {
11     |   if (a > b) {
12     |       |   return a;
13     |       |
14     |       |   else {
15     |       |       |   return b;
16     |       |       |
17     |       |       |
18     |       |       |
```

自定義函式

function.cpp

```
1  #include <stdio.h>
2
3  int max(int a, int b);
4
5  int main() {
6
7      return 0;
8  }
9
10 int max(int a, int b) {
11     if (a > b) {
12         return a;
13     }
14     else {
15         return b;
16     }
17 }
18
```

在這裡的參數名稱會被忽略
(但建議加上，增加可讀性)

參數(用逗號分隔)

函式名稱 參數資料型態

int max(int a, int b) {

參數名稱

回傳資料型態
若不回傳，可設為void

自定義函式

function.cpp

```
1  #include <stdio.h>
2
3  int max(int a, int b);
4
5  int main() {
6      |
7      |   return 0;
8      |
9      |
10 int max(int a, int b) {
11     |   if (a > b) {
12     |       |   return a;  執行到return時，會直接離開函式
13     |       |
14     |       |   else {
15     |       |       |   return b;
16     |       |       |
17     |       |       |
18     |       |       |
```

自定義函式

function.cpp

```
1  #include <stdio.h>
```

```
2  
3  int fun1();
```

則fun1的回傳值是未定義的，
代表沒人知道

```
4  
5  int main() {  
6      int a = fun1();  
7      return 0;  
8  }
```

有回傳東西

```
9  
10 int fun1() {
```

沒有 return 東西

```
11  
12 }  
13
```



練習二

- 請寫出一個函式，能夠傳入三個數字，並回傳最大的數字。

```
2 3 4
4
2 4 3
4
3 2 4
4
3 4 2
4
4 2 3
4
4 3 2
4
_
```

小技巧

- 可以一直讀取，直到檔案結尾

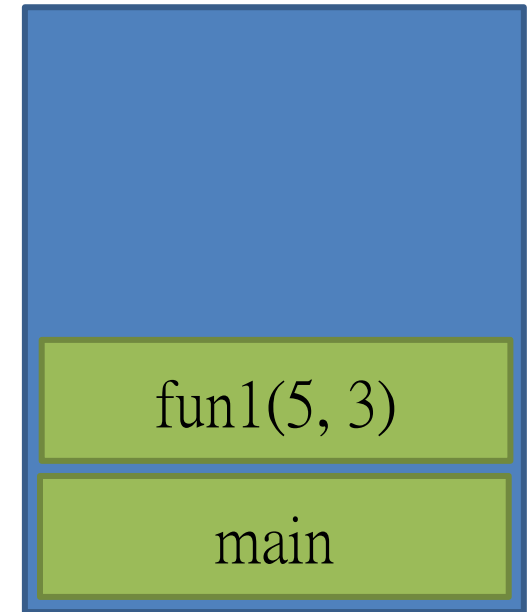
```
while (scanf("%d %d %d", &a, &b, &c) != EOF) {  
    .....  
}
```

↑
scanf也是函式，會回傳成功讀入的變數個數
如果讀到EOF，會回傳-1(EOF)

函式堆疊

CH04 Ex01.cpp

```
1  #include <stdio.h>
2  #include <stdlib.h>
3
4  void fun1(int a, int b);
5
6  int main() {
7      int a = 5, b = 3;
8      fun1(a, b);
9      printf("2. a = %d, b = %d\n", a, b);
10     system("pause");
11     return 0;
12 }
13
14 void fun1(int a, int b) {
15     a += 5;
16     b += 1;
17     printf("1. a = %d, b = %d\n", a, b);
18 }
19
```



函式堆疊

- 實際範例

| 呼叫堆疊 | | 語言 |
|---|--|-----|
| 名稱 | | |
| CH04 Ex01.exe!fun1(int a=5, int b=3) 行 14 | | C++ |
| CH04 Ex01.exe!main() 行 8 + 0xd 位元組 | | C++ |
| CH04 Ex01.exe!_tmainCRTStartup() 行 555 + 0x19 位元組 | | C |
| CH04 Ex01.exe!mainCRTStartup() 行 371 | | C |
| kernel32.dll!76f6919f() | | |
| [下面的框架可能錯誤及/或遺失，未載入 kernel32.dll 的符號] | | |
| ntdll.dll!77df0bbb() | | |
| ntdll.dll!77df0b91() | | |

呼叫堆疊 中斷點 輸出 記憶體 1

註：這是從Microsoft Visual C++ 2010 Express 截圖的

傳值呼叫

CH04 Ex01.cpp

```
1  #include <stdio.h>
2  #include <stdlib.h>
3
4  void fun1(int a, int b);
5
6  int main() {
7      int a = 5, b = 3;
8      fun1(a, b);
9      printf("2. a = %d, b = %d\n", a, b);
10     system("pause");
11     return 0;
12 }
13
14 void fun1(int a, int b) {
15     a += 5;
16     b += 1;
17     printf("1. a = %d, b = %d\n", a, b);
18 }
19
```

1. a = 10, b = 4

2. a = 5, b = 3

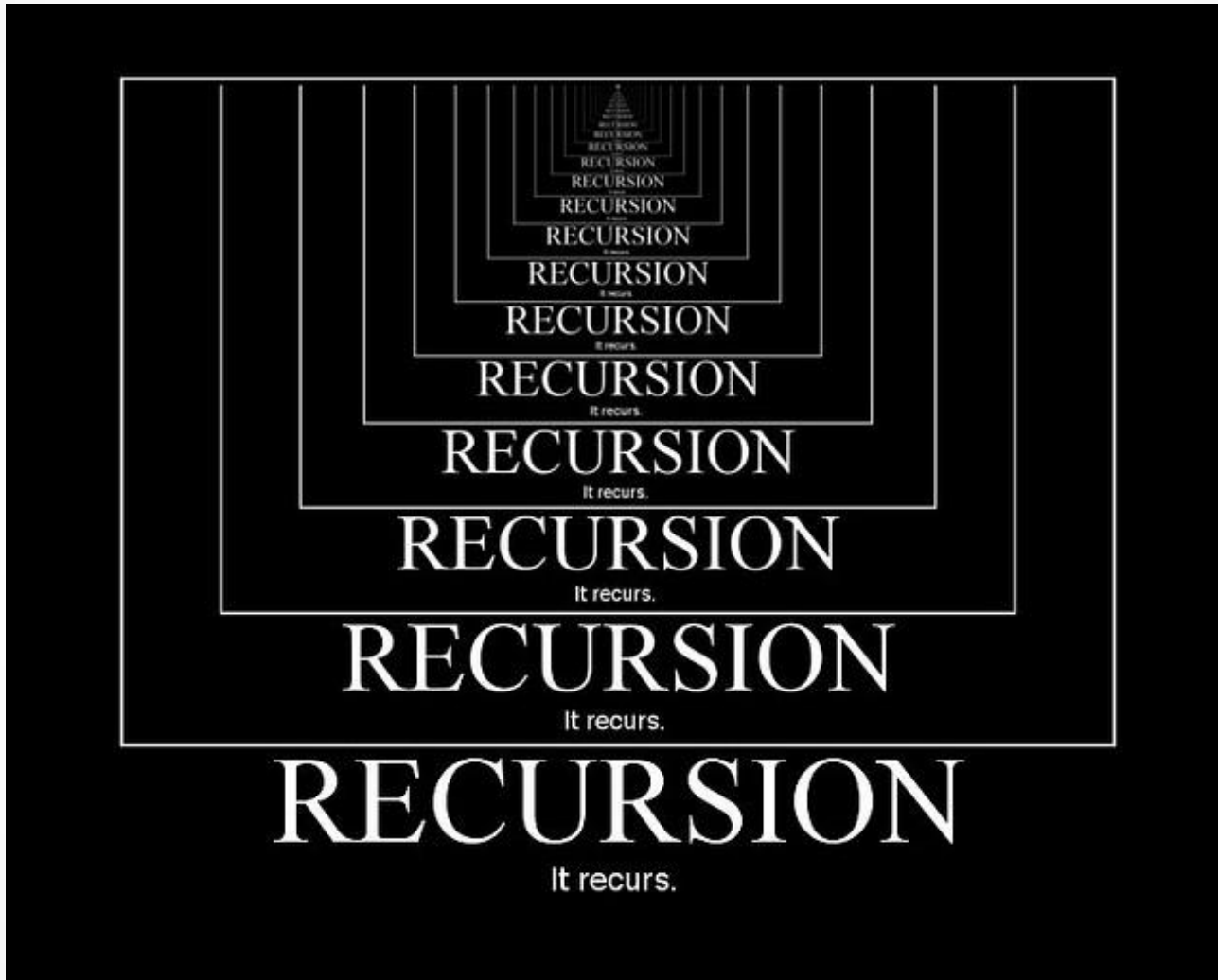
請按任意鍵繼續 . . .

遞迴

...

Recursion

遞迴



Google “遞迴”



遞迴



網頁

新聞

地圖

圖片

影片

更多 ▾

搜尋工具

約有 2,010,000 項結果 (搜尋時間：0.22 秒)

您是不是要查：[遞迴](#)

遞迴

CH04 Ex02.cpp

```
1  #include <stdio.h>
2
3  int pow(int a, int b); //a ^ b
4
5  int main() {
6      printf("%d\n", pow(5, 3));
7      return 0;
8  }
9
10 int pow(int a, int b) {
11     if (b == 1) {
12         return a;
13     }
14     else {
15         return a * pow(a, b - 1);
16     }
17     呼叫自己
18 }
```

遞迴

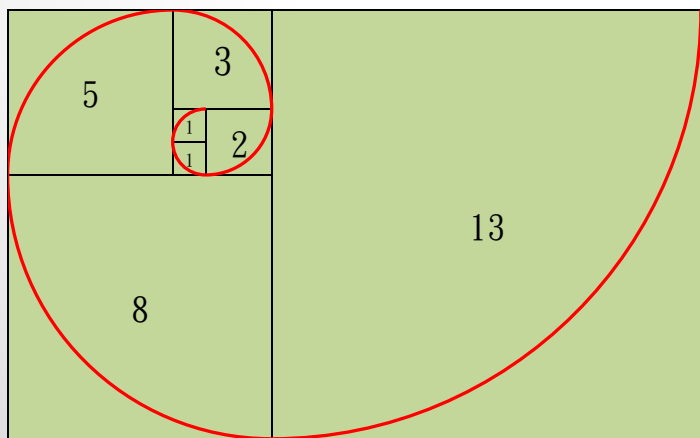
$$\begin{aligned}\text{pow}(5, 3) &= 5 * \text{pow}(5, 2) \\ &= 5 * [5 * \text{pow}(5, 1)] \\ &= 5 * [5 * 5] \\ &= 5 * 25 \\ &= 125\end{aligned}$$

遞迴優缺點

- 優點：
 - 如果問題可以分解成多個子問題(總計算量要小於等於原問題)，可以使用遞迴
- 缺點：
 - 函式呼叫有成本(∵傳值呼叫要複製)
 - 不好除錯(Debug)

練習三

- 使用遞迴計算出費式數列第0項 到 第20項



```
fib< 0> = 0
fib< 1> = 1
fib< 2> = 1
fib< 3> = 2
fib< 4> = 3
fib< 5> = 5
fib< 6> = 8
fib< 7> = 13
fib< 8> = 21
fib< 9> = 34
fib<10> = 55
fib<11> = 89
fib<12> = 144
fib<13> = 233
fib<14> = 377
fib<15> = 610
fib<16> = 987
fib<17> = 1597
fib<18> = 2584
fib<19> = 4181
fib<20> = 6765
請按任意鍵繼續 . . .
```

補充

Random

...

Random

- 大多數的亂數產生器都是產生「偽亂數」，給予相同的seed，會產生相同的數列
- 通常使用”現在時間”當作亂數的seed
- 雖然是「偽亂數」，可是已經很夠用了

```
seed =      100 ->   365,   1216,   5415,  16704,  24504,  11254,  24698
seed =    14586 -> 14902, 21863, 16907,  11883,  19349,   8615,  22895
seed = 1416316706 -> 22319,   8778,   6255,  31375,  26038,   4639,   7161
```


使用函式介紹

- `int rand()` in `stdlib.h`
隨機回傳一個介於 0 到 `RAND_MAX` (通常是32767)的整數
- `void srand(unsigned int seed)` in `stdlib.h`
使用給予的seed初始化亂數產生器
- `time_t time(time_t* timer)` in `time.h`
回傳從1970年1月1日 00:00 (UTC)到現在經過的總秒數。

範例

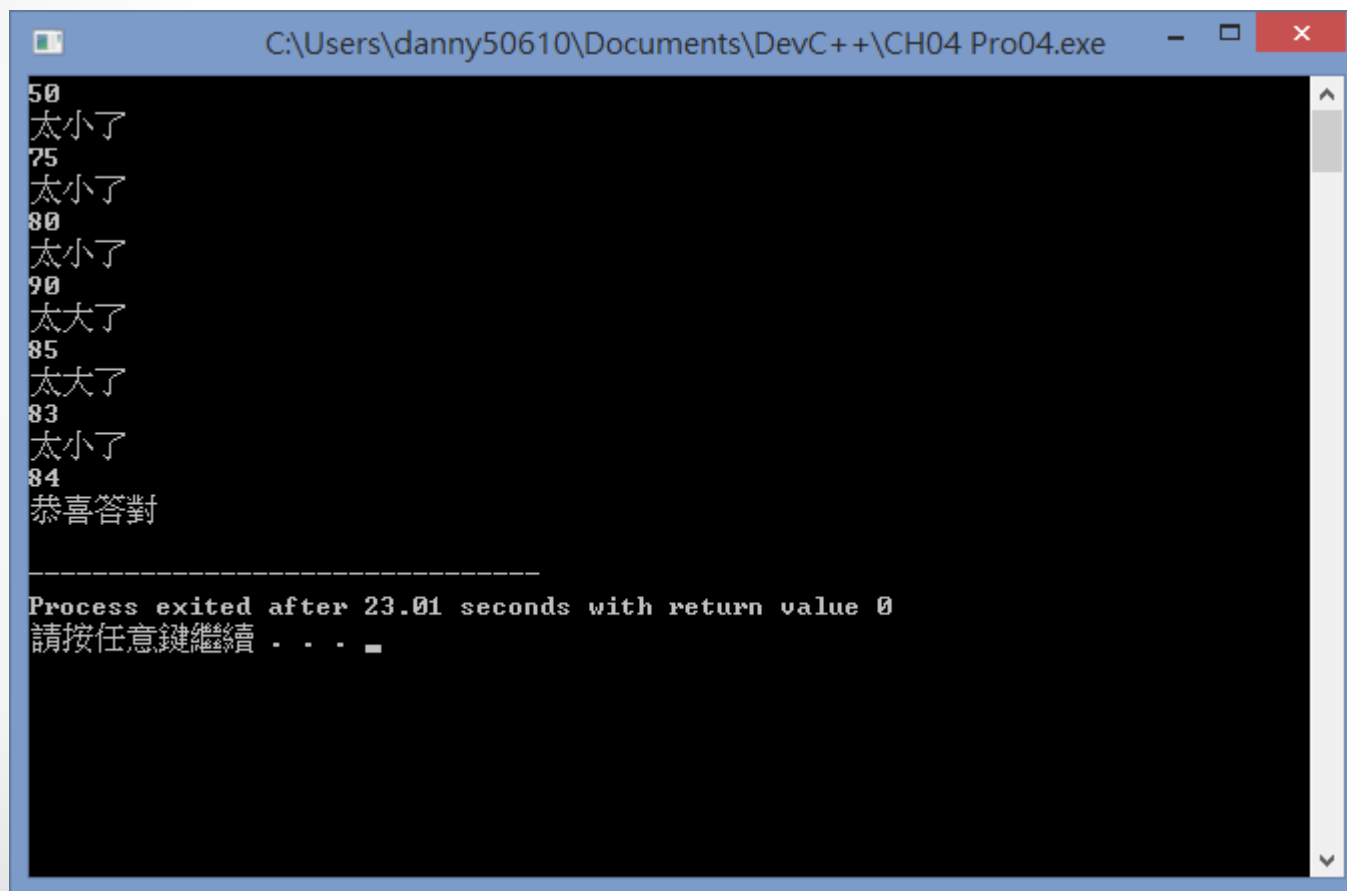
CH04 Ex04.cpp X

(未知的範圍)

```
1 #include <stdio.h>
2 #include <stdlib.h>
3 #include <time.h>
4
5 int main() {
6     srand(time(NULL));           //初始化亂數產生器
7     printf("%d\n", rand());      //產生亂數
8     printf("%d\n", rand() % 10); //只產生介於 0 ~ 9 的亂數
9     system("pause");
10    return 0;
11 }
12
```

練習四

- 寫一個猜數字的程式，答案介於1 ~ 100
(請參考下圖)



```
C:\Users\danny50610\Documents\DevC++\CH04 Pro04.exe
50
太小了
75
太小了
80
太小了
90
太大了
85
太大了
83
太小了
84
恭喜答對

-----
Process exited after 23.01 seconds with return value 0
請按任意鍵繼續 . . .
```

良好的程式設計

- 函式
 - 取有意義的名稱，最好能反映函式的功能
 - 一個函式只負責一件事
 - 參數不要過多
- 遞迴
 - 想清楚再開始寫遞迴
 - 記得設離開點
- 養成排版的好習慣