【第五章】

函式

講師: 李根逸 (Ken-Yi Lee), E-mail: feis.tw@gmail.com

課程大綱

- 函式宣告 [P139]
- 函式呼叫 [P140]
- C 標準函式庫 [P141]
- 函式定義 [P144]
- 變數可視範圍 [P151]
- 產生亂數 [P157]
- 函式遞迴 [P160]

函式宣告

- 函式的概念來自於數學。可以把每個函式想成是一個程式模組，包含一段程式片段，具有某種設計好的功能。

- 函式宣告：

```
回傳值的資料型態 函式名稱(參數資料型態 [參數名稱], ...);
```

```
double sqrt( double x);
```

  - 一個函式可以接受零到多個參數輸入，回傳一個值。
  - [補充] 在函式宣告時，參數名稱不是必要可以省略。

函式呼叫

- 函式呼叫語法:
  - 函式名稱([參數值], ...)
  - 範例:

```
sqrt(4.0)
```

- 呼叫函式時，會將函式需要的參數值算出後執行函式內容。函式執行完後則會回傳一個指定型態的值。

```
double v = 2.0;

double s = sqrt(2.0*v);
double s = sqrt(2.0*2.0);
double s = sqrt(4.0);
double s = 2.0;
```

- 在程式碼中呼叫函式之前需要宣告或定義函式

C 標準函式庫

- C 語言有內建事先寫好的的函式庫，例如 **sqrt**（求平方根）這類的數學函式：


```
sqrt(4.0) == 2.0
```

  - 備註: sqrt 需引用 <math.h>

- 除此之外還有 **printf**、**scanf** 和 **system** 這種非數學函式：


```
printf("Hello world") == 11
```

[查詢常用函式] Windows: MSDN 網站, Linux 或 Unix-like: man 指令

<math.h> 數學函式庫

常見的數學函式	說明	範例
double **sqrt** (double x)	x 的平方根	sqrt(900.0) 是 30.0
double **exp** (double x)	e 的 x 次方	exp(1.0) 是 2.718282
double **log** (double x)	x 的自然對數 (底為 e)	log(2.718282) 是 1
double **fabs** (double x)	x 的絕對值	fabs(-5.0) 是 5.0
double **ceil** (double x)	不小於 x 的最小整數	ceil(9.2) 是 10.0
double **floor** (double x)	不太於 x 的最大整數	floor(9.2) 是 9.0
double **pow** (double x, double y)	x 的 y 次方	pow(2, 7) 是 128.0
double **fmod** (double x, double y)	x 除以 y 的浮點餘數	fmod (13.657, 2.333) 是 1.992
double **sin** (double x)	x 的正弦值	sin(0.0) 是 0.0
double **cos** (double x)	x 的餘弦值	cos(0.0) 是 1.0
double **tan** (double x)	x 的正切值	tan(0.0) 是 0.0

《補充》C 的內建函式庫

<assert.h>	協助偵錯函式
<ctype.h>	字元處理
<errno.h>	錯誤處理函式
<float.h>	浮點數相關定義
<limits.h>	資料型態屬性
<locale.h>	地區相關
<math.h>	數學
<setjmp.h>	在函式間跳躍

<signal.h>	訊號 (signal) 處理
<stdarg.h>	讓函式使用不固定個數參數
<stddef.h>	標準定義
<stdio.h>	輸入輸出
<stdlib.h>	一般函式
<string.h>	字串處理
<time.h>	時間計算

函式定義

- 函式定義語法：

```
回傳值的資料型態 函式名稱(參數資料型態 參數名稱, ...) {
  /* 程式碼內容 */
  return 回傳值;
}
```

  - 函式只能定義一次，且不可在函式內定義函式

- 範例：**max2** 是一個用來求兩整數最大值的函式

```
int max2(int x, int y) {
  if (x >= y) {
    return x;
  } else {
    return y;
  }
}
```

main 也是一個函式，會在程式一開始執行時被呼叫

```c
/* 求兩整數最大值 */
#include <stdio.h>
#include <stdlib.h>
int max2(int, int);

int main() {
    int a, b;
    printf("請輸入一個整數:");
    scanf("%d", &a);
    printf("請輸入另一個整數: ");
    scanf("%d", &b);
    printf("其中比較大的整數是 %d\n", max2(a, b));
    system("pause");
    return 0;
}

int max2(int x, int y) {
    if (x >= y) {
        return x;
    } else {
        return y;
    }
}
```

函式宣告
(跟變數一樣，在呼叫函式之前我們需要先宣告)

函式呼叫
(執行指定的函式)

函式定義
(函式的真正內容所在)

函式呼叫時會先將參數的值算出後再傳入函式

將定義放置在呼叫前後都可以，但是在呼叫前一定要宣告或定義

【思考】為什麼要分成宣告跟定義？

```c
/* 求兩整數最大值 */
#include <stdio.h>
#include <stdlib.h>
int max2(int x, int y) {
    if (x >= y) {
        return x;
    } else {
        return y;
    }
}

int main() {
    int a, b;
    printf("請輸入一個整數:");
    scanf("%d", &a);
    printf("請輸入另一個整數: ");
    scanf("%d", &b);
    printf("其中比較大的整數是 %d\n", max2(a, b));
    system("pause");
    return 0;
}
```

函式定義
(函式的真正內容所在)

將定義放置在呼叫前
可以省去宣告的動作

函式呼叫
(執行指定的函式)

《範例》定義數學函式

- 試寫一名為 **abs** 的函式，傳入一個整數後回傳該整數的絕對值
- 試寫一名為 **power** 的函式，傳入底數 **x** 與指數 **y** 兩個浮點數 **(double)** 後，回傳 **x** 的 **y** 次方是多少 ？

《範例》定義模組化函式

- 修改 **triangle.cpp** 這一程式，將『畫 **N** 個星星』模組化成 **triangle_1.cpp** 這個樣子
  - 這裡我們寫一個函式 **void draw_stars(int** len) 來畫星星

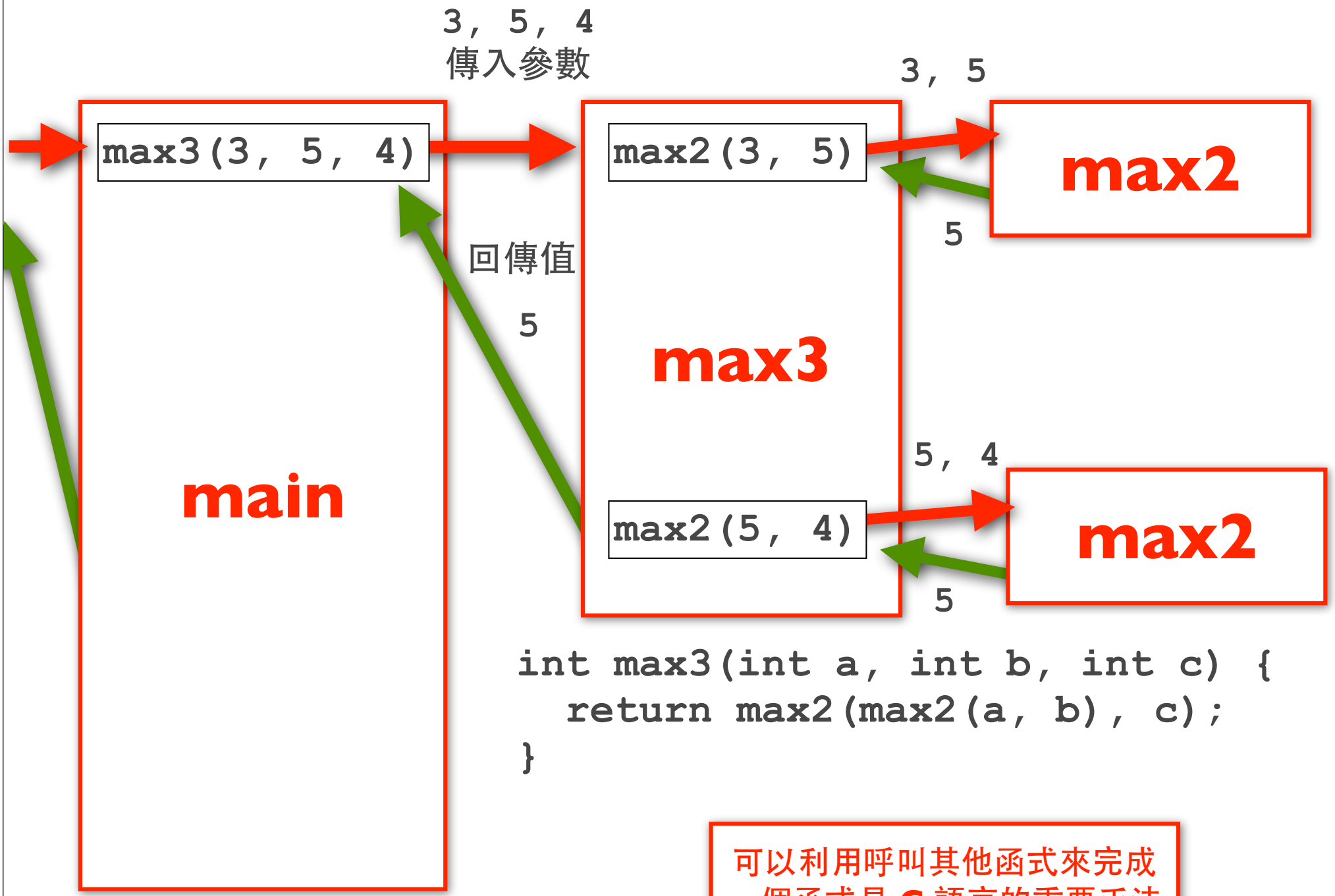
```
Please enter an integer: 3
*
**
***

Please enter an integer: 5
*
**
***
****
*****
```

void 是個特殊的資料型態，代表虛無的意思，這裡用在不需要回傳值的函式。當回傳值型態是 **void** 時，**return** 後不接回傳值。

《範例》比大小

- 試寫一函式 **max3** 讓使用者輸入三個數字，並回傳三個數字中最大的
  - 提示：可以利用 max2 來幫你完成！




```
int max3(int a, int b, int c) {
  return max2(max2(a, b), c);
}
```

可以利用呼叫其他函式來完成
一個函式是 C 語言的重要手法

變數可視範圍

- **一般我們可以依照變數宣告的位置分為兩種變數：**
  - 全域變數 (Global variable)
    - 宣告在函式外 (從程式開始會配置，程式結束會釋放)
    - 因為容易造成名稱污染，請避免使用全域變數
  - 區域變數 (Local variable)
    - 宣告在函式或區塊內 (從宣告開始會配置，函式或區塊結束會釋放)
- **變數可視範圍 (scoping)**
  - 一個變數可以被某段程式碼使用要同時滿足兩個條件：
    - 該變數是全域變數或該變數宣告的區塊包含這段程式碼
    - 這段程式碼位於該變數宣告以後

```c
#include <stdio.h>
#include <stdlib.h>
int i = 5;

void p(){
  /* int i = -1; */
  i = i + 1;
  printf("%d\n", i);
  return;
}

int main(){
  printf("%d\n", i);
  int i = 6;
  i = i + 1;
  p();
  printf("%d\n", i);
  system("pause");
  return 0;
}
```

這程式的執行結果是？

```c
#include <stdio.h>
#include <stdlib.h>

int main(){
  int i = 3;
  printf("%d\n", i);
  if (i == 3) {
    i = i + 1;
    int i = 6;
    printf("%d\n", i);
    i = i + 1;
  }

  if (i == 3) {
    printf("%d\n", i);
  }
  system("pause");
  return 0;
}
```

這程式的執行結果是？

在 for 的小括號內宣告

- 在迴圈前宣告的變數，在迴圈結束後一樣可以存取：

```
int i = 1;
while (i <= 10) {
  printf("%d\n", i);
  i++;
}
printf("%d\n", i);
```

```
int i;
for (i = 1; i <= 10; i++) {
  printf("%d\n", i);
}
printf("%d", i);
```

- 在 **C99** 後我們可以在 **for** 小括號內的初始式宣告變數，值得注意的是此時的變數在迴圈執行過程中指的是同一個，而離開迴圈後該變數就離開可視範圍：

```
for (int i = 1; i <= 10; i++) {
  int j = i;
  printf("%d\n", j);
}
printf("%d\n", i); /* 不合法 */
```

這個特性可以讓 **for** 計數用的變數不會污染到後面的程式碼

《補充》靜態修飾字 (static)

- 除了之前所提的全域變數和區域變數外，我們在變數前可以加上 **static** 修飾字：
  - 加在全域變數前:
    - 該變數的生命週期不變
    - 該變數的可視範圍由全程式變成該檔案本身
  - 加在區域變數前:
    - 該變數的生命週期與全域變數相同　(整個程式開始時配置一次，結束時釋放一次)
      - 該變數從程式開始到結束只會初始化一次
    - 該變數的可視範圍不變

參考 **static** 範例檔

《補充》函式的參數預設值

- 我們在定義函式的參數時可以給予預設值：
  - 例如：

```
int max3(int a, int b, int c = 0) { ... }
```

    - 當呼叫 **max3(3, 4)** 時，就會如同呼叫 **max3(3, 4, 0)**
    - 注意的是，在有多個參數時，前面的參數如果設定有預設值的話，後面的參數也一定要有：
      - 這個是不合法的：

```
int max3(int a, int b = 0, int c) { ... }
```

產生亂數

- **C 標準函式庫中包含了一個能產生整數亂數的函式：**
  - long **rand**(): 回傳一長整數型態的亂數 [定義於 **stdlib.h**]
  - 每次呼叫 rand() 函式都會得到不同的回傳值

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

int main() {
    for  (int i = 1; i <= 5; i++) {
        printf("產生的亂數: %d\n", rand());
    }
    system("pause");
    return 0;
}
```

《範例》擲骰子

- 試寫一程式 **(dice.cpp)** 讓電腦亂數產生十個一到六之間的數字，模擬擲骰子的結果
  - 提示: rand() 的回傳值型態是長整數 (long)，我們可以利用求餘數 (%) 運算子來求取某固定範圍的整數亂數

產生夠亂的亂數

- 電腦產生亂數是利用內存的一個亂數表，因此我們必須利用時間函式庫 **<time.h>** 內的 **srand** 函式適當的選取第一個開始的亂數種，否則亂數出現的順序將是固定的。
  - 可以用 srand(time(NULL)) 指令初始化亂數表
    - **void srand(unsigned** seed)：用 seed 初始化亂數表
    - **time_t time**(NULL)：回傳目前時間 [定義於 **time.h**]

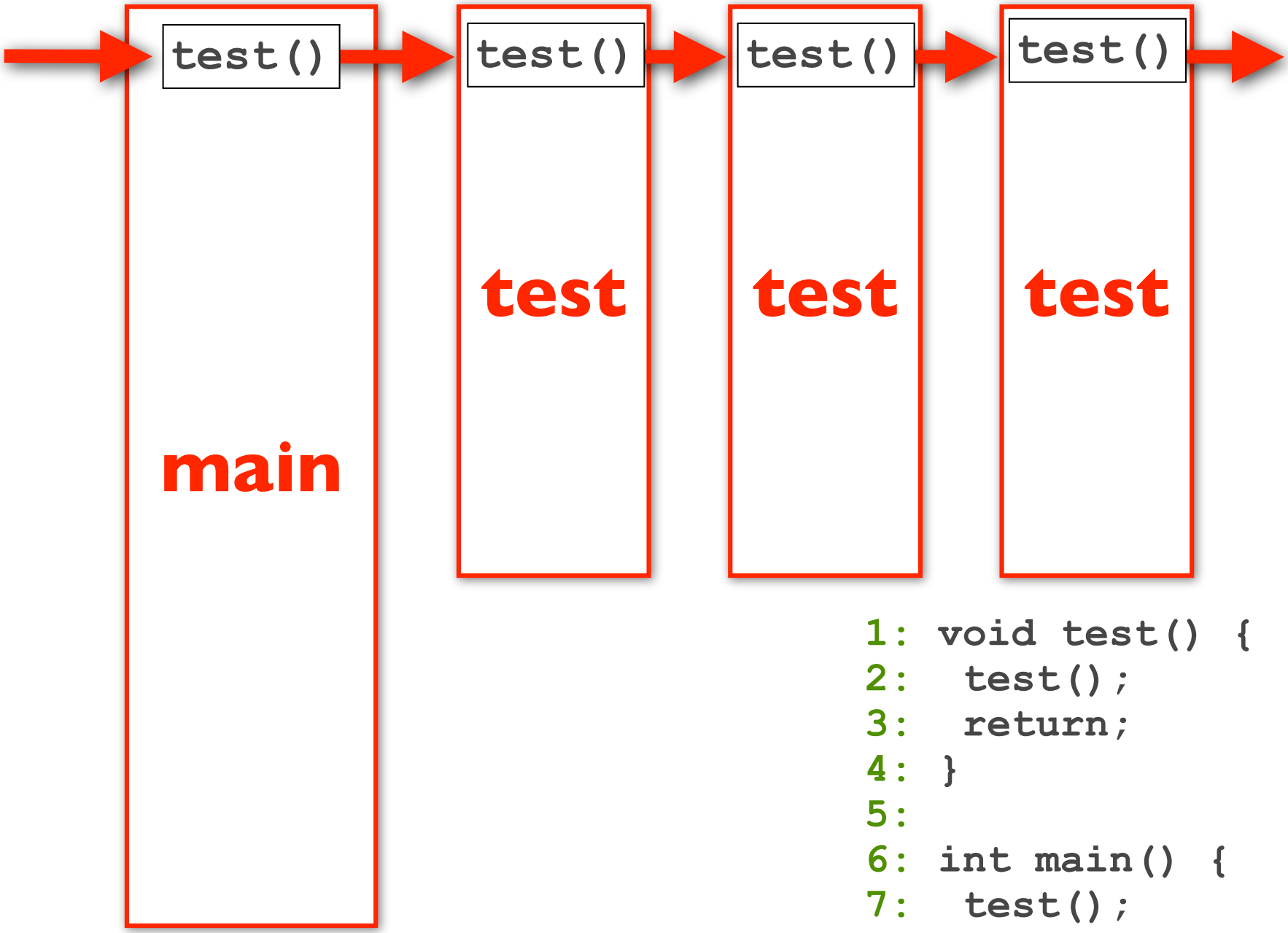
```
#include <stdio.h>
#include <stdlib.h>
#include <time.h>
int main(){
    srand(time(NULL));
    printf("A random number: %d\n", rand());
    system("pause");
    return 0;
}
```

函式遞迴

- **遞迴是函式應用上最迷人而容易困惑的一種觀念**
  - 遞迴是將一個大問題化成一些相似的小問題的方法
- **函式遞迴指的是在函式定義內又呼叫同名函式：**
  - 在 test 函式定義內又呼叫 test，試想這樣會產生怎樣的結果 ？

```
void test() {
  test();
  return;
}

int main() {
  test();
  return 0;
}
```




```
1: void test() {
2:   test();
3:   return;
4: }
5:
6: int main() {
7:   test();
8:   return 0;
9: }
```

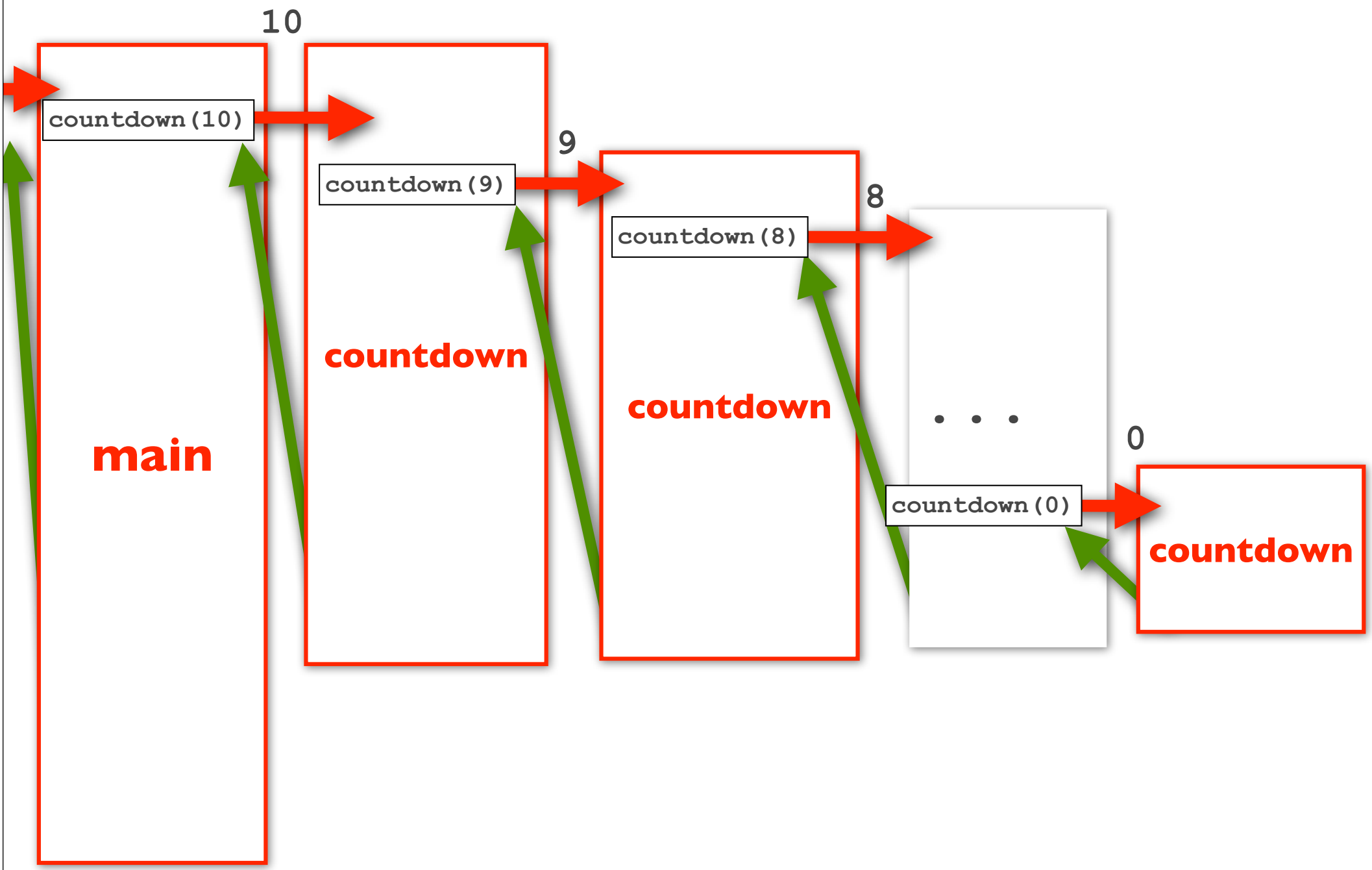
《範例》倒數計時 1

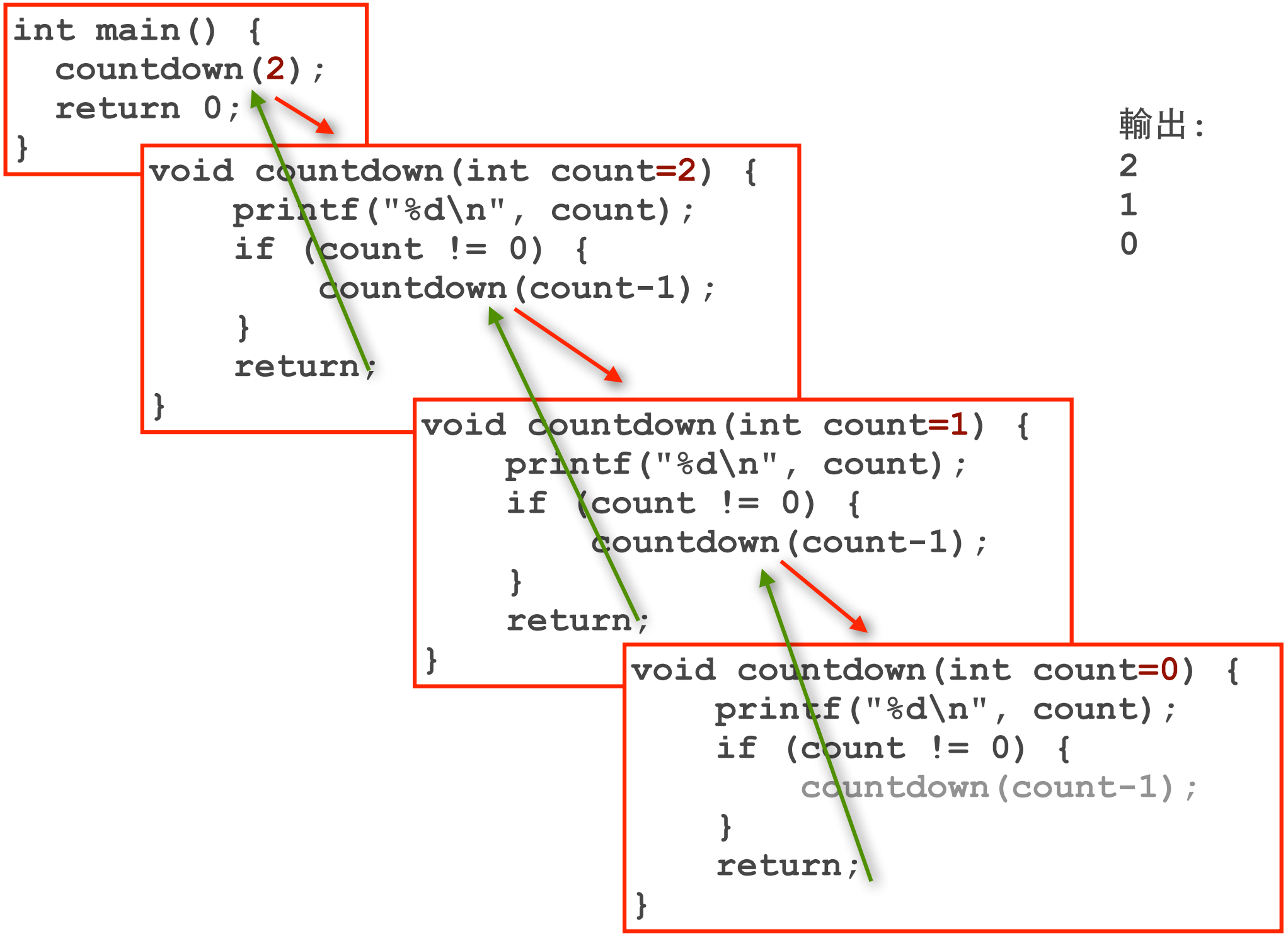
- 試想一下下列程式碼 **(countdown_1.cpp)** 會產生怎樣的結果？

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

void countdown(int count) {
    printf("%d\n", count);
    if (count != 0) {
        countdown(count-1);
    }
    return;
}

int main() {
  countdown(10);
  system("pause");
  return 0;
}
```


10
countdown(10)
9
countdown(9)
8
countdown(8)
countdown(0)
0
main
countdown
countdown
countdown


int main() {
countdown(2);
return 0;
}
void countdown(int count=2) {
printf("%d\n", count);
if (count != 0) {
countdown(count-1);
}
return;
}
void countdown(int count=1) {
printf("%d\n", count);
if (count != 0) {
countdown(count-1);
}
return;
}
void countdown(int count=0) {
printf("%d\n", count);
if (count != 0) {
countdown(count-1);
}
return;
}
輸出:
2
1
0

《範例》倒數計時 2

- 試想一下下列程式碼 **(countdown_2.cpp)** 會產生怎樣的結果？

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

void countdown(int count) {
    if (count != 0) {
        countdown(count-1);
    }
    printf("%d\n", count);
    return;
}

int main() {
  countdown(10);
  system("pause");
  return 0;
}
```

```
int main() {
  countdown(2);
  return 0;
}
```

```
void countdown(int count=2) {
    if (count != 0) {
        countdown(count-1);
    }
    printf("%d\n", count);
    return;
}
```

```
void countdown(int count=1) {
    if (count != 0) {
        countdown(count-1);
    }
    printf("%d\n", count);
    return;
}
```

```
void countdown(int count=0) {
    if (count != 0) {
        countdown(count-1);
    }
    printf("%d\n", count);
    return;
}
```

輸出:

```
0
1
2
```

《範例》求數字和

- 試寫一 **sum** 的函式 **(sum.cpp)** 去計算 **1+2+...+N** 的值
  - sum(N) == N + sum(N-1) 且 sum(1) == 1

```
int sum(int N) {
    if (N == 1) {
        return 1;
    }
    return N + sum(N-1);
}
```

- 可以用迴圈 **(while** 或 **for)** 來寫嗎 **?**

《補充》C++ 的函式重載

- 在 C 語言裡面，同樣名字的函式，只能有一個
- 在 C++ 語言裡面，同樣名字的函式，只要參數個數或型別不同就可以有很多個，稱為函式重載 **(function overloading)**

```
void print(int v) { printf("%d\n", v); }
void print(double v) { printf("%f\n", v); }

print(3); // 呼叫 print(int)
print(3.0); // 呼叫 print(double)
```

習題 (1)

- **[E0501]** 試寫一函式 **max4** 讓使用者輸入四個整數，並回傳四個數字中最大的
  - 請試著寫出上面兩例在執行時呼叫函式的歷程？
- **[E0502]** 試寫一個 **round** 函式，傳回輸入參數四捨五入至整數位後的值
  - `int round(float num) { ... }`
  - 數學函式庫 **(math.h)** 中有一 **floor** 函式可以將輸入參數無條件捨去
- **[E0503]** 試寫一個程式讓電腦擲 **6000** 次骰子，並統計一到六各個數字分別出現了幾次

習題 (2)

- **[E0504]** 試寫一程式，產生一 **0** 至 **9** 之間整數亂數，讓使用者猜測該數字為何，直到使用者猜對。
- **[E0505]** 試寫一程式，輸入一元二次方程式的三個參數 **(a, b, c)**，求該一元二次方程式的實數解
  - 無實數解時請印出警告訊息
  - $a\mathbf{x}^2+b\mathbf{x}+c = 0$
  - <math.h> 內有 sqrt 函式可以用來平方根值

習題 (3)

(遞迴)

- **[E0506]** 試用遞迴函式寫一程式，讓使用者輸入一正整數 **N** 後顯示 **N!** 的值
  - N! = 1 * 2 * 3 * ... * N = (N-1)! * N
- **[E0507] Fibonacci 數列：**
  - 0, 1, 1, 2, 3, 5, 8, 13, 21 ...
  - 規則：
    - 數列中的前兩個數是 0 和 1
    - 數列中的其他的數是前兩個數字的和
  - 試寫一遞迴程式去計算第 N 個的 Fibonacci 數是多少