程式設計
(Programming)

真理大學 資訊工程系 吳汶涓老師

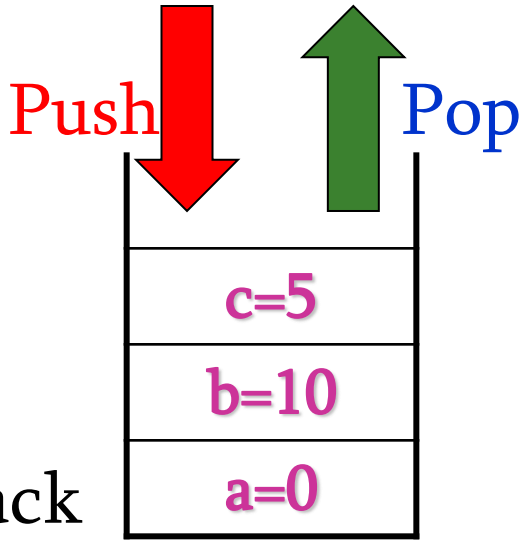
CH05

函式 (function)

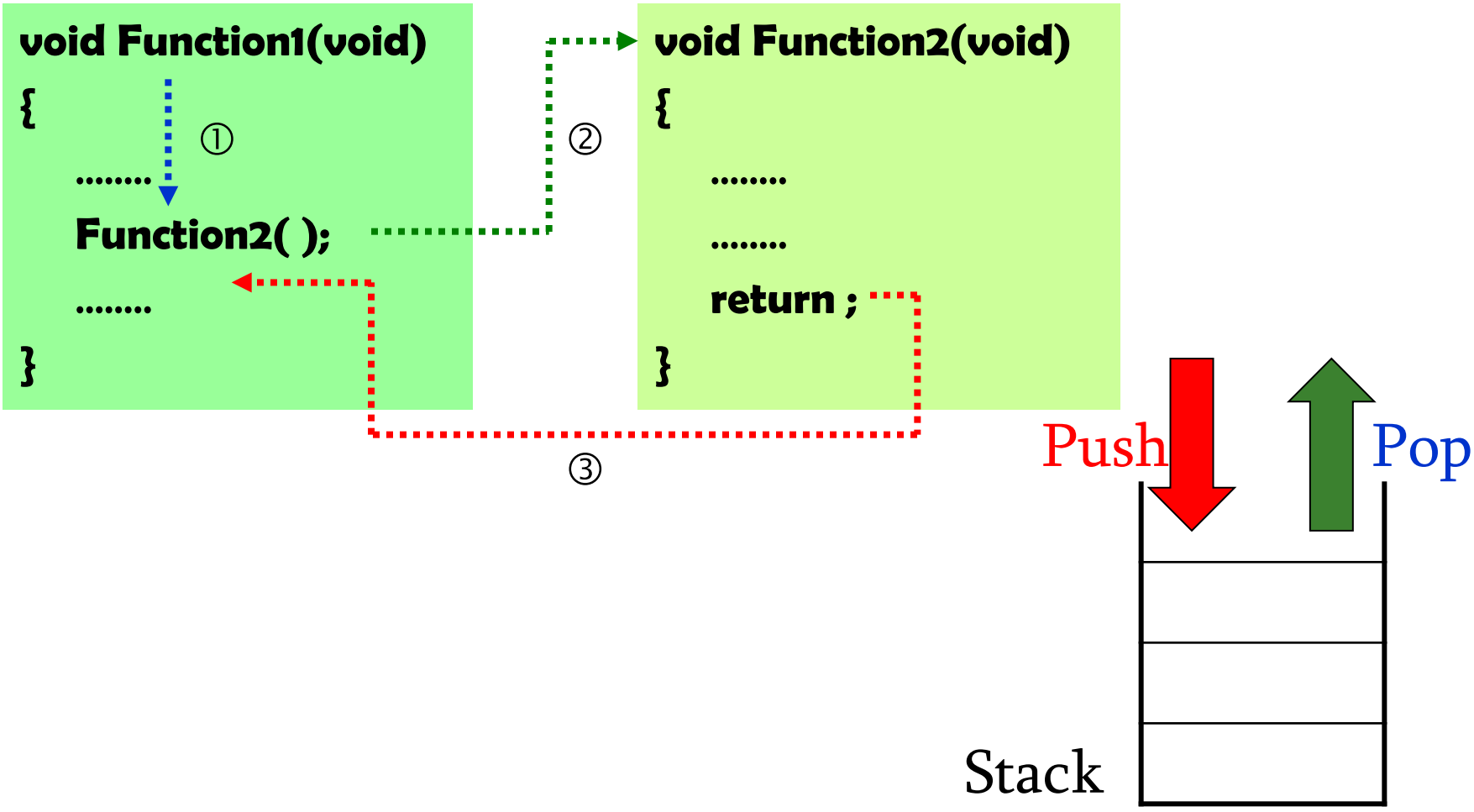
本章綱要

5.7 函式呼叫堆疊與活動紀錄

- 函式呼叫執行**堆疊 (stack)**
  - 堆疊是一種**後進先出(last-in-fist-out, LIFO)**的結構
    - push: 放入資料
    - 任何放入堆疊的東西都會擺在「最頂端」
    - pop: 取出資料
    - 每次只有「最頂端」的資料可以被取出
  - C利用程式執行堆疊來**記錄先前呼叫的函式**
  - 函式呼叫的數量超過C的可處理能力，會產生**堆疊溢位(stack overflow)**的錯誤



- How Function Call Works



5.8 標頭(header)

- 標頭檔
  - 內含函式庫函式的**原型**
  - <stdlib.h>, <math.h> 等等
  - 使用 **#include <filename>** 載入
- 自訂標頭檔
  - 建立函式檔案，如square函式存在**square.h**中
  - 存成 filename.h
  - 使用 **#include <filename>** 載入
  - 可重複使用函式

標準函式庫標頭	說明
<assert.h>	內含一些用來幫助程式偵錯的巨集和資訊。
<ctype.h>	內含一些檢測字元特性及大小寫字元轉換等函式的原型。
<errno.h>	定義了一些用來回報錯誤狀況的巨集。
<float.h>	內含此系統中對浮點數大小的限制。
<limits.h>	內含此系統中對整數大小的限制。
<locale.h>	內含一些能夠使程式區域化的函式原型與資訊。區域化的表示方式讓電腦系統能處理世界各地各種不同的資料（如日期，時間，金額及大的數目）表示習慣。

圖 5.6　標準函式庫標頭檔

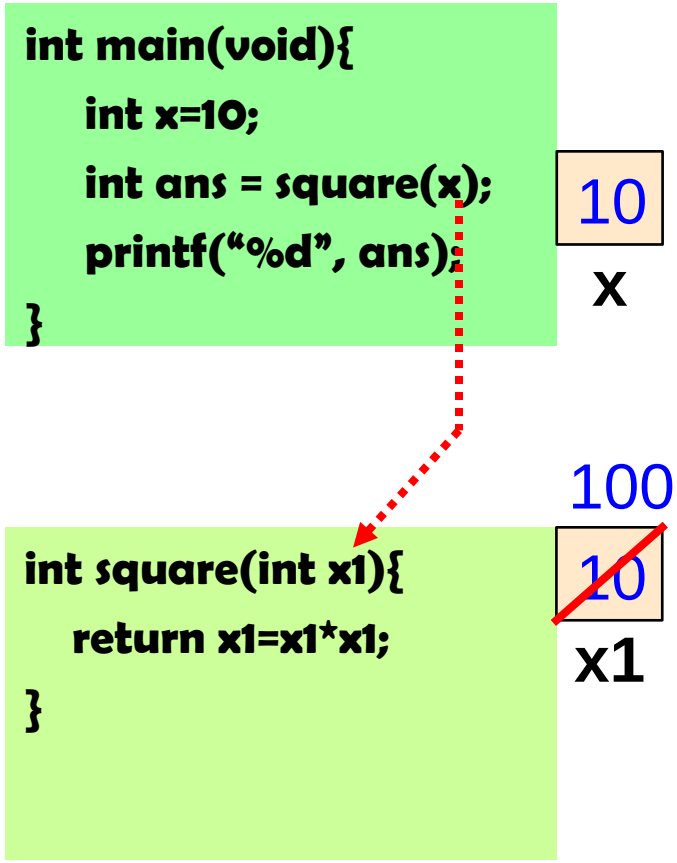
標準函式庫標頭	說明
<math.h>	內含數學函式庫的函式原型。
<setjmp.h>	內含改變正常函式呼叫與回傳順序的函式原型。
<signal.h>	內含處理程式執行中各種狀況的函式原型和巨集。
<stdarg.h>	定義一些處理不確定型別及個數之引數列的函式。
<stddef.h>	內含一些在 C 執行運算時所常用到的型別。

標準函式庫標頭	說明
<stdio.h>	內含標準輸出 / 入函式庫的函式原型以及所需的資訊。
<stdlib.h>	內含一些數字與文字間轉換，記憶體配置，亂數，及其他公用函式的原型。
<string.h>	內含字串處理函式的原型。
<time.h>	內含處理時間與日期的函式原型。

圖 5.6　標準函式庫標頭檔

5.9 呼叫函式：傳值呼叫、傳參考呼叫

- 傳值呼叫 (call by value)
  - 將傳遞給函式的引數複製一份值
  - 函式值的改變不會影響原來的引數值
- 傳參考呼叫 (call by reference)
  - 傳遞原來引數
  - 函式值的改變會影響原來的引數值
- 目前我們只關心傳值呼叫



練習

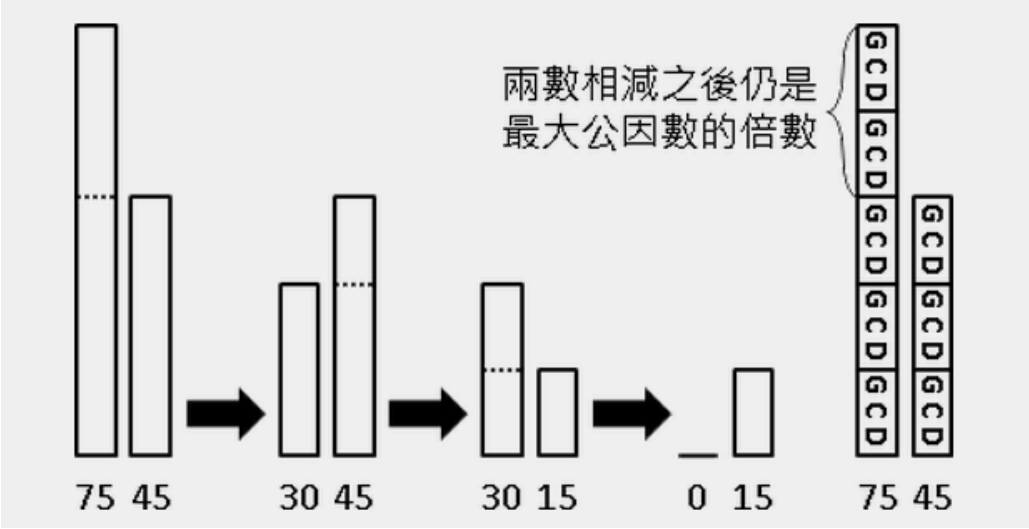
- 建立一標頭檔，將函式撰寫於此，此函式能傳回整數參數值的反數，若參數值為7631，則傳回1367。並且，撰寫一程式能重複地輸入整數值，透過載入標頭檔得到反數，直到輸入-1為止。

```
input a value: 7631
output: 1367
input a value: 4323
output: 3234
input a value: -1
End…
```

類似課本pp. 5-59, Ex. 5.22

練習

- 撰寫一函式gcd，傳回兩個整數的最大公因數(GCD)，最大公因數是指能**同時整除**這兩個數的最大整數值。



- 撰寫一函式lcm，傳回兩個整數的最小公倍數(LCM)，最小公倍數是指能**同時被這兩個數整除**的最小整數值。
  - LCM結果 = 兩數相乘 / GCD結果

課本pp. 5-60, Ex. 5.29, 5.39

5.10 亂數產生

- rand函式
  - 載入 **<stdlib.h>**
  - 回傳 0 ~ RAND_MAX 之間的隨機數字 (RAND_MAX至少為2147483647)
    如 **i = rand( );**
  - 虛擬亂數 (pseudorandom numbers)
    - 「亂」數順序是固定的
    - 每次函式呼叫都會產生同樣的序列

int 所能表示的最大整數值

如何產生固定區間的隨機數字呢?

- 比例化 (scaling)
  - 取得 1~ n 之間的亂數
    - 先回傳 0 ~ (n-1) 的數字 ← 使用餘數運算子 %
    - 加1調整到 1 ~ n 的區間內

i = rand( )%n + 1;

  - 例如，丟六面的骰子
    - i = rand( )%6 + 1
    - 會產生 1~6 的亂數值

```
12    for ( i = 1; i <= 20; i++ ) {
15       printf( "%10d", 1 + ( rand() % 6 ) );
18       if ( i % 5 == 0 ) {
19          printf( "\n" );
20       }
21    }
```

```
6     6     5     5     6
5     1     1     5     3
6     6     2     4     2
6     2     3     4     1
```

圖 5.7　移位過且比例化過的整數

```c
/* Fig. 5.8: fig05_08.c  */
#include <stdio.h>
#include <stdlib.h>

int main( void )
{
   int frequency1 = 0;
   int frequency2 = 0;
   int frequency3 = 0;
   int frequency4 = 0;
   int frequency5 = 0;
   int frequency6 = 0;

   int roll;
   int face;
   for ( roll = 1; roll <= 6000; roll++ ) {
      face = 1 + rand() % 6; /* random number from 1 to 6 */
      switch ( face ) {
         case 1:
            ++frequency1; break;
         case 2:
            ++frequency2; break;
         case 3:
            ++frequency3; break;
         case 4:
            ++frequency4; break;
         case 5:
            ++frequency5; break;
         case 6:
            ++frequency6; break;
      }
   }
```

比例化

Face	Frequency
1	1003
2	1017
3	983
4	994
5	1004
6	999

課本pp. 5-21, pp. 5-60, Ex. 5.31

圖 5.8　投擲一個 6 面的骰子 6000 次

- srand函式
  - 載入 **<stdlib.h>**
  - **隨機化(randomizing)** 則可產生不同的亂數序列
    - 提供一個unsigned型別的種子(seed)，可使用 **%u** 讀到變數中

```
unsigned seed; /* number used to seed random number generator */
scanf( "%u", &seed );
srand( seed );
i = rand();
```

  - 不需要自行輸入種子
    - seed可使用秒數，也就是 **srand(time(NULL));**
    - time(NULL) 會回傳從1970/1/1到目前所經過的秒數
    - 需載入 **<time.h>**

常見的程式設計錯誤 5.9

用 srand 來產生亂數，而不是用 rand。這是一項常見的錯誤。

```c
/* Fig. 5.9: fig05_09.c */
#include <stdlib.h>
#include <stdio.h>

int main( void )
{
   int i;
   unsigned seed;

   printf( "Enter seed: " );
   scanf( "%u", &seed ); /* note %u for unsigned */

   srand( seed );
   for ( i = 1; i <= 10; i++ ) {
      printf( "%10d", 1 + ( rand() % 6 ) );
      if ( i % 5 == 0 ) {
         printf( "\n" );
      }
   }
   return 0;
}
```

替rand函式提供seed

```
Enter seed: 67
         6      1      4      6      2
         1      6      1      6      4
```

```
Enter seed: 867
         2      4      6      1      6
         1      1      3      6      2
```

```
Enter seed: 67
         6      1      4      6      2
         1      6      1      6      4
```

圖 5.9　將擲骰子程式隨機化

練習

- 請寫出敘述式，將亂數按照下列各範圍的規定，指定給變數n。

(a) $1 \le n \le 2$ ← `n = rand( ) % 2 + 1;`

(b) $1 \le n \le 100$ ← `n = rand( ) % 100 + 1;`

(c) $0 \le n \le 9$

(d) $1000 \le n \le 1112$ ← `n = rand( ) % 113 + 1000;`

(e) $-1 \le n \le 1$

(f) $-3 \le n \le 11$

(g) $n \in \{6,12,18,24,30\}$ ← `n = (rand( )%5+1)*6 ;`

(h) $n \in \{4,9,16,24,36\}$

(i) $n \in \{5,10,15,20,25\}$

課本pp. 5-57, Ex. 5.13

5.11 實例：機會遊戲

- 模擬 Craps 擲骰子遊戲
- 規則
  - 投擲兩顆骰子，辨別輸贏

  - 若第一次為 7點或11點：玩家贏
  - 若第一次為 2點、3點或12點：玩家輸
  - 若第一次為 4點、5點、6點、8點、9點或10點，為「目標點數」，玩家必須在莊家擲出7點贏之前，投擲出與目標點數一樣，才算贏

- 函式：rollDice( ) 用來投擲兩顆骰子，回傳骰子總點數

課本pp. 5-25

- 函式：rollDice( ) 用來投擲兩顆骰子，回傳骰子總點數

```
75  /* roll dice, calculate sum and display results */
76  int rollDice( void )
77  {
78     int die1;
79     int die2;
80     int workSum;
81
82     die1 = 1 + ( rand() % 6 );
83     die2 = 1 + ( rand() % 6 );
84     workSum = die1 + die2;
85
87     printf( "Player rolled %d + %d = %d\n", die1, die2, workSum );
88     return workSum;
89  }
```

圖 5.11　crap 遊戲的執行範例

```
1  /* Fig. 5.10: fig05_10.c Craps */
3  #include <stdio.h>
4  #include <stdlib.h>
5  #include <time.h>
8  enum Status { CONTINUE, WON, LOST };
9
10 int rollDice( void ); /* function prototype */
13 int main( void )
14 {
15    int sum;
16    int myPoint;
18    enum Status gameStatus;
21    srand( time( NULL ) );
22
23    sum = rollDice();
25    /* determine game status based on sum of dice */
26    switch( sum ) {
29       case 7:
30       case 11: gameStatus = WON;  break;
35       case 2:
36       case 3:
37       case 12: gameStatus = LOST; break;
42       default:
43          gameStatus = CONTINUE;
44          myPoint = sum;
45          printf( "Point is %d\n", myPoint );
46          break;
47    }
```

enum 列舉常數：表示識別字的整數集合
0 為CONTINUE
1 為WON
2 為LOST

gameStatus 為列舉型別的變數

```
Player rolled 5 + 6 = 11
Player wins
```

```
Player rolled 1 + 1 = 2
Player loses
```

```
50      while ( gameStatus == CONTINUE ) {
51         sum = rollDice();
54         if ( sum == myPoint ) {
55            gameStatus = WON;
56         }
57         else {
58            if ( sum == 7 ) {
59               gameStatus = LOST;
60            }
61         }
62      }
63
65      if ( gameStatus == WON ) {
66         printf( "Player wins\n" );
67      }
68      else {
69         printf( "Player loses\n" );
70      }
```

```
Player rolled 4 + 1 = 5
Point is 5
Player rolled 6 + 2 = 8
Player rolled 2 + 1 = 3
Player rolled 3 + 2 = 5
Player wins
```

```
Player rolled 6 + 4 = 10
Point is 10
Player rolled 3 + 4 = 7
Player loses
```

圖 5.11　crap 遊戲的執行範例

練習

- 投擲兩顆骰子 (計算兩點數之差)
  - 若第一次為 0點：玩家贏
  - 若第一次為 3點：玩家輸
  - 若第一次為 1點、2點、4點或5點，為「目標點數」，玩家必須在莊家擲0點贏之前，投擲出目標點數才算贏

常見的程式設計錯誤 5.11

只使用大寫字母當作列舉常數的名稱，讓人可以注意到這些常數，並且可以指出這些列舉常數不是變數。

練習

- 撰寫一個程式，利用亂數產生五個分數(0~100分)，並使用函式技巧來計算這五個分數的最高分、最低分、平均分數以及標準差 $\sigma = \sqrt{\frac{1}{N}\sum_{i=1}^{N}(x_i - \overline{x})^2}$ 。

finish the generation
random numbers: 87, 63, 54, 91, 34
max: 91
min: 34
average: 65.80
standard: XXXX.XX