

# CHAPTER 3

Functions(ch5)



- **Functions**是模組化的一大特色，將一個大的應用程式切割為數**Functions**(副程式或函式)，每個**Function**都有特定的功能。

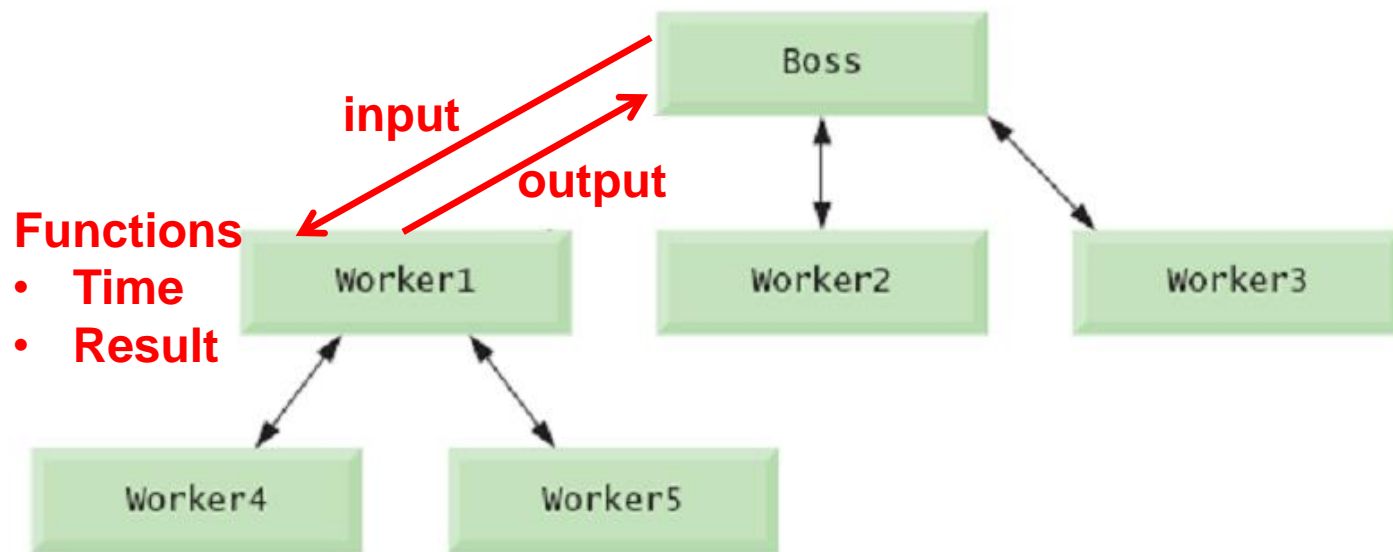
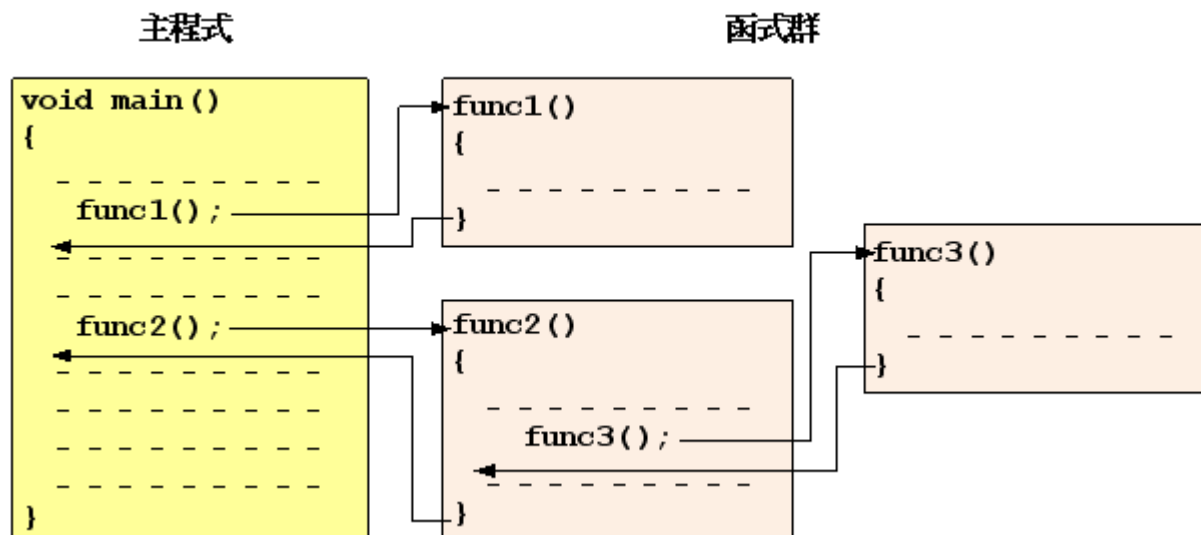


圖 5.1 階層式的老闆函式／員工函式關係圖

- **Functions(函式與副程式)的特性**
  - 將程式結構化(**structure**)
    - 使程式碼更容易閱讀
  - 將程式碼模組化(**module**)與物件化(**object**)
    - 提高程式碼重複使用率
  - 將程式碼抽象化 (**abstraction** )
    - 設計師只需知道如何使用程式碼, 不需要知道程式內容, 可快速完成程式
  - 提高程式設計效率, 容易設計、維修、擴充、交流

- 將一個大的主程式切割為數**Functions**，當主程式呼叫**Functions**時
  - **Step 1:** 程式的控制權將會轉移到**相對應的函式開頭處**，然後執行函式中的程式碼
  - **Step 2:** 函式的程式碼執行完畢後，程式控制權將重新回到主程式碼（呼叫敘述）的下一個敘述，繼續往下執行
  - 這之間的控制權轉換必須記憶**下一個敘述在記憶體中的位址**，而編譯器在編譯函式呼叫時，會以**系統堆疊**來存放該位址，**程式設計師不用為此特別撰寫程式碼**。



- 在C語言中，呼叫函式的語法如下
  - 語法1（函式無回傳值）：

函式名稱(傳入引數串列);

```
void compute_area(float r, float pi)
```

- 語法2（函式有回傳值）：

變數=函式名稱(傳入引數串列);

```
float compute_area(float r, float pi)
```

```
int compute_area(float r, float pi)
```

- Functions(函式與副程式)的宣告語法如下：

函式回傳值型態      函式名稱(資料型態 [參數1],資料型態 [參數2],.....);

- Ex: `int square( int y );`

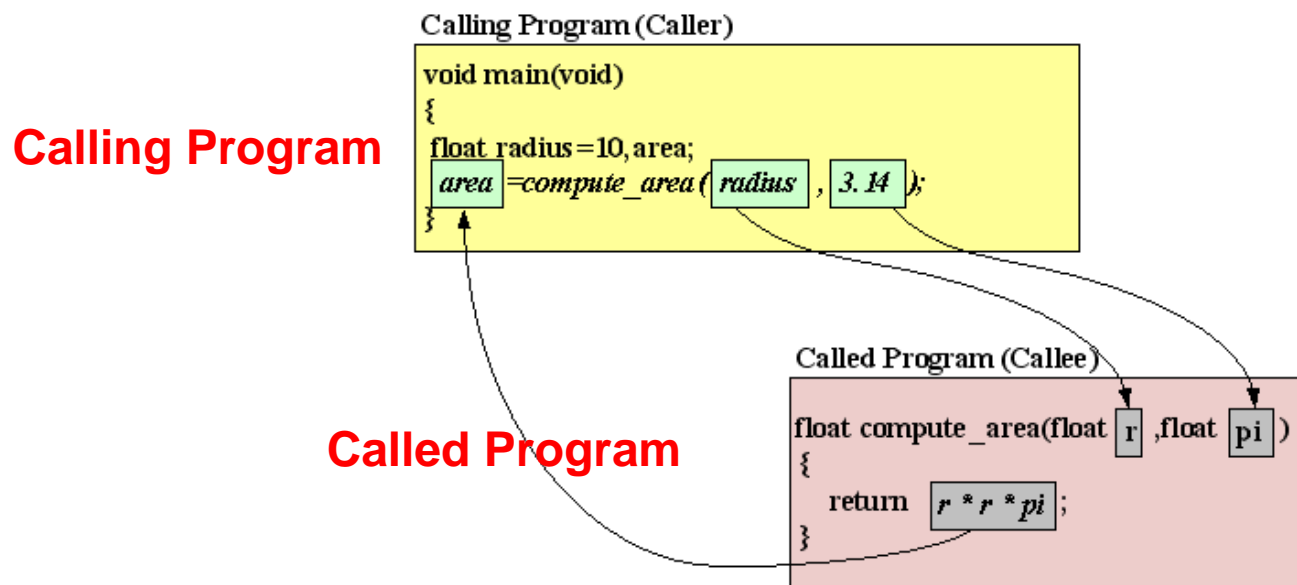
- 輸入整數y 透過square副程式運算,得到函式輸出結果為整數

- Functions(函式與副程式)經過宣告後，代表編譯器得知該程式中存在一個這樣的函式，如果我們是使用使用者自訂的模組函式，我們必須再定義函式的內容（也就是該函式要執行的程式碼），如此才能成為程式中一個完整可用的函式。

```
函式回傳值型態 函式名稱(資料型態 參數1,資料型態 參數2,.....)
{
    .....函式主體（程式碼） .....
    [return ... ;]
}
```

- Ex: `int square( int y ) /* y is a copy of argument to function */`  
`{`  
 `return y * y; /* returns square of y as an int */`  
`} /* end function square */`

- 函式呼叫者(Calling Program)必須與函式(Called Program)名稱相同，但兩者之引數與參數名稱可以不同。
- 若呼叫者(Calling Program)有資料要傳遞給被呼叫者(Called Program)，則必須藉由傳入引數串列將資料傳遞給函式的參數，並且『傳入引數串列』的傳入變數會由『函式定義的參數串列』的相對參數來接收，其順序、個數必須相同，但引數名稱可以與參數名稱不同。如下圖示意：



- Functions(函式與副程式)宣告範例

宣告函式範例	解說
<code>void func1( );</code>	func1函式無回傳值，呼叫時也不必輸入引數。
<code>float func2(int a);</code>	func2函式的回傳值為float資料型態，並且呼叫時需要一個整數型態的輸入引數。
<code>int func3(int a,char b);</code>	func3函式的回傳值為int資料型態，並且呼叫時需要有兩個輸入引數，分別傳送給整數型態的a，和字元型態的b。
<code>int func4(int,char);</code>	同func3，但省略宣告參數名稱（參數資料型態不可省略）。

- 參數在函式主體內屬於合法的資料變數，也就是說，我們不用在函式主體內重複宣告這些參數，就可以直接將這些參數當作已宣告的變數使用。
- 具有回傳值的函式，在函式主體內應該包含至少一個**return**敘述，以便傳回資料。**不具回傳值的函式**則可以沒有**return**敘述

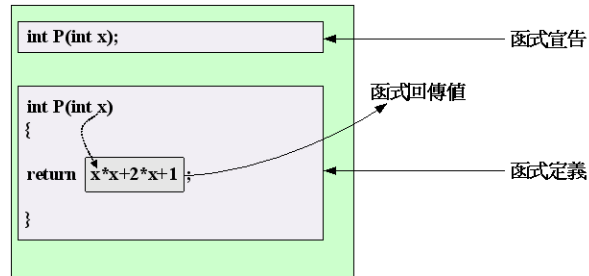
函式定義範例	解說
<pre>void ShowWelcome(int print_times) {     int a;     for(a=1;a&lt;=print_times;a++)         printf("您好,歡迎光臨\n"); }</pre>	<p>(1)您可以在函式主體內使用參數print_times。</p> <p>(2)<b>函式無回傳值。//沒有 return</b></p>



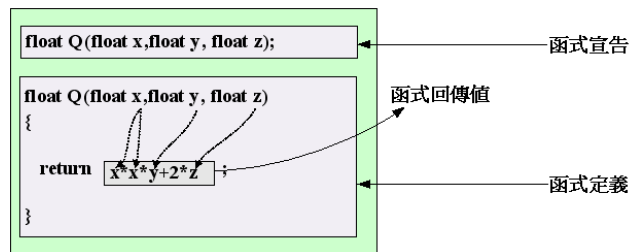
- 使用者自訂的模組函式 (function)宣告範例

函式定義範例	解說
<pre>int Mul(int a,int b) {     int result;     result = a*b;     return result; }</pre>	<p>(1)您可以在函式主體內使用參數a,b。</p> <p>(2)函式回傳值的資料型態為int。</p> <p>(3)使用return回傳資料，result為Mul函式的回傳值。</p> <p>(4)return敘述執行完畢，控制權將立刻返回原呼叫函式的下一個敘述。</p>
<pre>double Add(double a,double b) {     return (a+b); }</pre>	<p>(1)您可以在函式主體內使用參數a,b。</p> <p>(2)函式回傳值的資料型態為double。</p> <p>(3)使用return回傳資料，(a+b)運算式的結果為Add函式的回傳值。</p> <p>(4)執行完畢return敘述，控制權將立刻返回原呼叫函式的下一個敘述。</p>

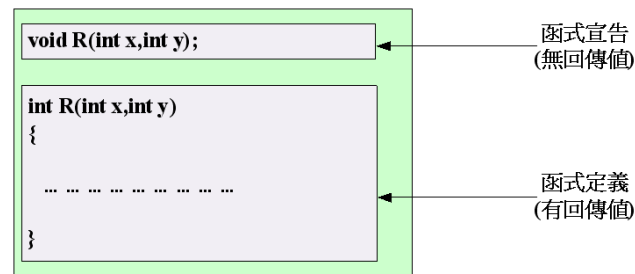
- [正確] 函式P可計算 $x^2+2x+1$ ，其中 $x$ 屬於整數。



- [正確] 函式Q可計算 $x^2y+2z$ ，其中 $x,y,z$ 屬於浮點數。

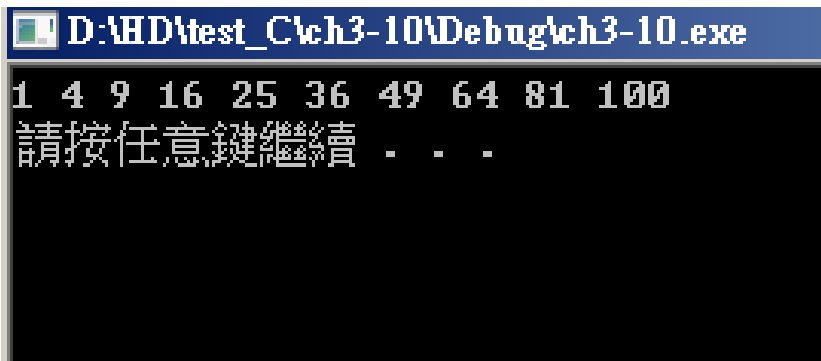


- [錯誤] 在R函式宣告中，已經宣告該函式無回傳值(void)，但在R函式定義中卻指定為整數回傳值型態(int)，因此，編譯器會找不到符合宣告格式的函式定義，而發生錯誤。



- **Functions(函式與副程式)**主要有兩種
  1. 使用者自訂的模組函式 (**function**)
  2. **C標準函式庫 (C standard library)** 事先寫好的函式程式
    - ，**C標準函式庫**提供了包羅萬象的函式，包括以下幾類
      - 輸入/輸出(**#include<stdio.h>**) : **printf, scanf**
      - 數學運算(**#include<math.h>**) : **sqrt(x), log(x)**
      - 字串處理
      - 字元處理
      - 以及許多其他有用的功能
      - <http://www.cplusplus.com/reference/clibrary/>

- 使用者自訂的模組函式 (function)
- 問題: 設計平方副程式
  - `int square(int y);`
    - 輸入整數 $y$
    - 透過`square`副程式運算
    - 得到輸出結果為整數 $x$



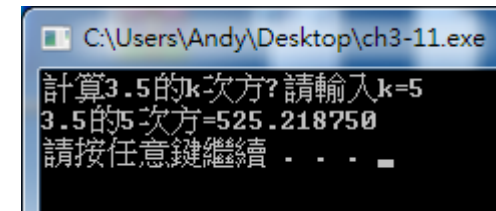
```
D:\HD\test_C\ch3-10\Debug\ch3-10.exe
1 4 9 16 25 36 49 64 81 100
請按任意鍵繼續 . . .
```

```
1 #include <stdio.h>
2 #include <stdlib.h>
3
4 int square(int y);
5
6 int main(void)
7 {
8     int x;
9
10    for(x=1;x<=10;x++)
11    {
12        printf("%d ",square(x));
13    }
14    printf("\n");
15    system("pause");
16    return 0;
17 }
18
19 int square(int y)
20 {
21     return y*y;
22 }
```

- 使用者自訂的模組函式 (function)
- 問題:計算x的n次方, `double Power (double x ,int n);`

```
1 #include <stdio.h>
2 #include <stdlib.h>
3
4 double Power(double, int);
5 void main(void)
6 {
7     int k; double Ans;
8     printf("計算3.5的k次方?請輸入k=");
9     scanf("%d",&k); Ans=Power(3.5,k);
10    printf("3.5的%d次方=%f\n",k,Ans);
11    system("pause");
12 }
13
14 double Power(double X,int n)
15 {
16     int i; double PowerXn=1;
17     for(i=1;i<=n;i++)
18         PowerXn=PowerXn*X;
19     return PowerXn;
20 }
```

## 執行結果



Ex: k=5

Power(3.5, 5)

X=3.5, n=5

PowerXn = 1 x 3.5

PowerXn = 3.5 x 3.5

PowerXn = 3.5 x 3.5 x 3.5

PowerXn = 3.5 x 3.5 x 3.5 x 3.5

PowerXn = 3.5 x 3.5 x 3.5 x 3.5 x 3.5

- 使用者自訂的模組函式 (function)
- 計算 $x$ 的 $n$ 次方,  $\text{Power}(x, n)$ 函式呼叫之引數傳遞與回傳值如下圖

## Calling Program

```
void main(void)
{
    int k;
    double Ans;
    Ans = Power(3.5, k);
}
```

## Called Program

```
double Power(double X, int n)
{
    int i;
    double PowerXn=1;

    for(i=1;i<=n;i++)
        PowerXn=PowerXn*X;
    return PowerXn ;
}
```

- 使用者自訂的模組函式 (function)
- 使用者輸入三個整數，輸出三個整數的最大值

```
1 #include <stdio.h>
2 #include <stdlib.h>
3
4 int maximum(int x,int y,int z);
5
6 int main(void)
7 {
8     int number1;
9     int number2;
10    int number3;
11    printf("Enter three integers:");
12    scanf("%d %d %d",&number1,&number2,&number3);
13    printf("Maximum is: %d\n",maximum(number1,number2,number3));
14    system("pause");
15    return 0;
16 }
```

```
17
18 int maximum(int x,int y,int z)
19 {
20     int max=x;
21
22     if(y>max)
23     {
24         max=y;
25     }
26
27     if(z>max)
28     {
29         max=z;
30     }
31
32     return max;
33 }
```

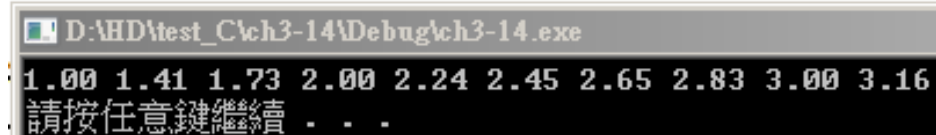
```
C:\Users\Andy\Desktop\ch3-13.exe
Enter three integers:22 85 17
Maximum is: 85
請按任意鍵繼續 . . .
```

```
C:\Users\Andy\Desktop\ch3-13.exe
Enter three integers:85 22 17
Maximum is: 85
請按任意鍵繼續 . . .
```

```
C:\Users\Andy\Desktop\ch3-13.exe
Enter three integers:22 17 85
Maximum is: 85
請按任意鍵繼續 . . .
```

- 使用C標準函式庫 (C standard library) 事先寫好的函式
- 問題: 從數學函式庫(`#include <math.h>`), 使用內建的開平方根(`sqrt(x)`)副程式
  - `double a = sqrt( 900); // a =30.0`

```
1 #include <stdio.h>
2 #include <stdlib.h>
3 #include <math.h>
4
5 int main(void)
6 {
7     int x;
8
9     for(x=1; x<=10; x++)
10    {
11        printf("%.2f ", sqrt(x));
12    }
13    printf("\n");
14    system("pause");
15    return 0;
16 }
```



D:\HD\test\_C\ch3-14\Debug\ch3-14.exe  
1.00 1.41 1.73 2.00 2.24 2.45 2.65 2.83 3.00 3.16  
請按任意鍵繼續 . . .



- **C**的常用數學函式庫(`#include <math.h>`)函式，在此表格中變數**x**和**y**的型別都是**double**。

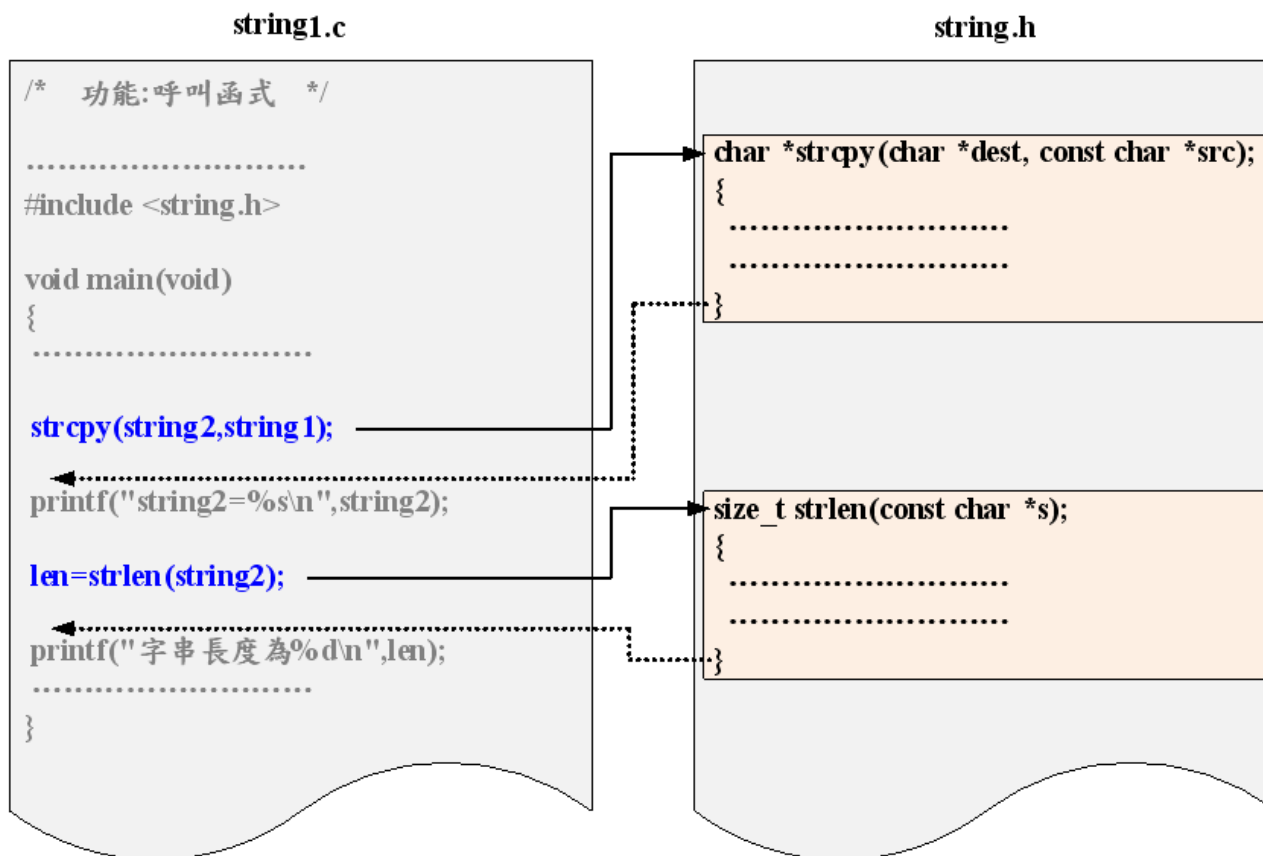
函式	說明	範例
<code>sqrt( x )</code>	$x$ 的平方根	<code>sqrt( 900.0 )</code> is 30.0 <code>sqrt( 9.0 )</code> is 3.0
<code>exp( x )</code>	指數函式 $e^x$	<code>exp( 1.0 )</code> is 2.718282 <code>exp( 2.0 )</code> is 7.389056
<code>log( x )</code>	$x$ 的自然對數 (底為 $e$ )	<code>log( 2.718282 )</code> is 1.0 <code>log( 7.389056 )</code> is 2.0
<code>log10( x )</code>	$x$ 的對數 (底為 10)	<code>log10( 1.0 )</code> is 0.0 <code>log10( 10.0 )</code> is 1.0 <code>log10( 100.0 )</code> is 2.0
<code>fabs( x )</code>	$x$ 的絕對值	<code>fabs( 13.5 )</code> is 13.5 <code>fabs( 0.0 )</code> is 0.0 <code>fabs( -13.5 )</code> is 13.5
<code>ceil( x )</code>	不小於 $x$ 的最小整數	<code>ceil( 9.2 )</code> is 10.0 <code>ceil( -9.8 )</code> is -9.0
<code>floor( x )</code>	不大於 $x$ 的最大整數	<code>floor( 9.2 )</code> is 9.0 <code>floor( -9.8 )</code> is -10.0
<code>pow( x, y )</code>	$x$ 的 $y$ 次方 ( $x^y$ )	<code>pow( 2, 7 )</code> is 128.0 <code>pow( 9, .5 )</code> is 3.0
<code>fmod( x, y )</code>	$x/y$ 的浮點餘數	<code>fmod( 13.657, 2.333 )</code> is 1.992
<code>sin( x )</code>	$x$ 的正弦值 ( $x$ 的單位為弧度)	<code>sin( 0.0 )</code> is 0.0
<code>cos( x )</code>	$x$ 的餘弦值 ( $x$ 的單位為弧度)	<code>cos( 0.0 )</code> is 1.0
<code>tan( x )</code>	$x$ 的正切值 ( $x$ 的單位為弧度)	<code>tan( 0.0 )</code> is 0.0

- 使用C標準函式庫 (C standard library) 事先寫好的函式
- 問題: 從字串處理函式庫(`#include <string.h>`), 使用內建`strcpy(string2,string1)`和`strlen(string2)`副程式
  - `strcpy(string2,string1);` //呼叫`strcpy`函式, 將`string1`內容複製到`string2`之中
  - `len=strlen(string2);` //呼叫`strlen`函式,計算`string2`的長度

```
1 #include <stdio.h>
2 #include <stdlib.h>
3 #include <string.h>
4
5 void main(void)
6 {
7     char string1[60]="Welcome";
8     char string2[60];
9
10    int len;
11
12    strcpy(string2,string1);
13    printf("string2=%s\n",string2);
14
15    len=strlen(string2);
16    printf("字串長度為%d\n",len);
17
18    system("pause");
19    return 0;
20 }
```

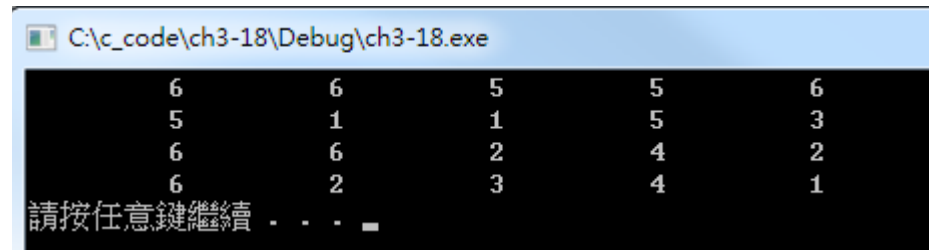


- 使用呼叫**strcpy( )**其實有一個**char \***資料型態的回傳值，但我們並未指定變數來接收這個回傳值。
- 使用呼叫**strlen( )**的回傳值為字串長度（**size\_t**的回傳值是整數資料**int**型態），所以我們利用**len**整數資料變數來儲存這個回傳值。



- 設計擲有六面的骰子
  - 使用rand()的函式原型在<stdlib.h>
  - rand() 會產生一個介於0 to RAND\_MAX的整數
  - rand()%6 會產生一個介於0 to 5的整數
  - 1 + (rand()%6)會產生一個介於1 to 6的整數

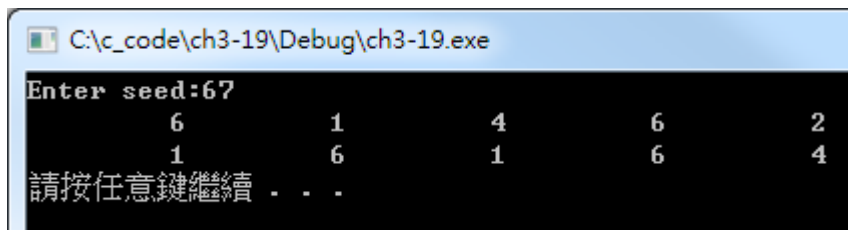
```
1 #include <stdio.h>
2 #include <stdlib.h>
3
4 int main(void)
5 {
6     int i;
7
8     for (i=1; i<=20; i++)
9     {
10         printf("%10d", 1+(rand()%6));
11         if(i%5==0)
12         {
13             printf("\n");
14         }
15     }
16
17     system("pause");
18     return 0;
19 }
```



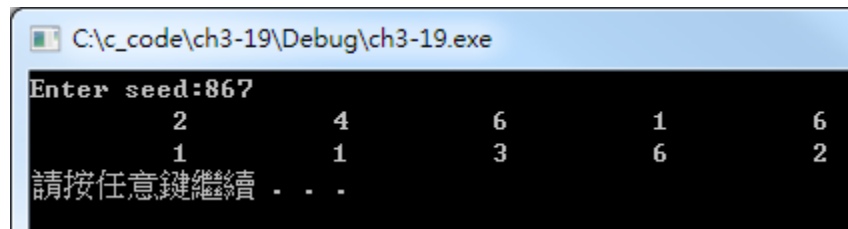
```
C:\c_code\ch3-18\Debug\ch3-18.exe
6      6      5      5      6
5      1      1      5      3
6      6      2      4      2
6      2      3      4      1
請按任意鍵繼續 . . .
```

- 設計擲有六面的骰子
  - 呼叫 **rand()** 函式所產的數列，看起來也是隨機的，但事實上每次都會出現相同的數列
  - 需要事先使用 **srand()** 函式並且 **unsigned** 一個整數引數，提供 **rand()** 函式一個種子 (**seed**)，讓 **rand()** 函式能夠在每次執行時產生不同順序的亂數。

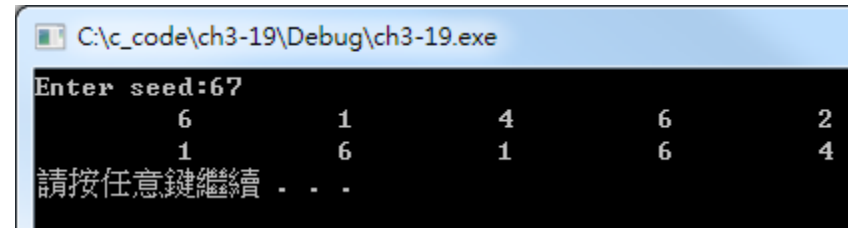
```
1 #include <stdio.h>
2 #include <stdlib.h>
3 int main(void)
4 {
5     int i;
6     unsigned seed;
7
8     printf("Enter seed:");
9     scanf("%u",&seed);
10
11     srand(seed);
12
13     for(i=1;i<=10;i++)
14     {
15         printf("%10d",1+(rand()%6));
16         if(i%5==0)
17         {
18             printf("\n");
19         }
20     }
21     system("pause");
22     return 0;
23 }
```



```
C:\c_code\ch3-19\Debug\ch3-19.exe
Enter seed:67
      6      1      4      6      2
      1      6      1      6      4
請按任意鍵繼續 . . .
```



```
C:\c_code\ch3-19\Debug\ch3-19.exe
Enter seed:867
      2      4      6      1      6
      1      1      3      6      2
請按任意鍵繼續 . . .
```



```
C:\c_code\ch3-19\Debug\ch3-19.exe
Enter seed:67
      6      1      4      6      2
      1      6      1      6      4
請按任意鍵繼續 . . .
```

- 設計"crap"的擲骰子遊戲，遊戲規則如下：
  - 1) 玩家獨自投擲兩顆骰子，計算兩顆骰子總和(sum)。
  - 2) 如果玩家第一次投擲總和(sum)為7點或11點，那麼判定**玩家贏**。
  - 3) 如果玩家第一次投擲總和(sum)為2點、3點或12點 (這些點數稱為**crap**)，那麼則是**玩家輸**。
  - 4) 如果玩家第一次投擲總和(sum) 4點、5點、6點、8點、9點或10點，則這個點數成為玩家的**目標點數(my point)**。玩家需要一直繼續投擲這兩顆骰子，直到出現**目標點數(my point)**或是7點，若先出現**目標點數**，則是**玩家贏**。但先出現7點，則判定**玩家輸**。
- 演算法
- if (sum == 7 or 11), then won
- if (sum == 2 or 3 or 12) , then lost
- if (sum == 4 or 5 or 6 or 8 or 9 or 10),
  - then (mypoint = sum) and **continue until sum = mypoint or 7**
  - if (sum == mypoint), then won
  - if (sum == 7), then lost

if (sum == 7 or 11), then won  
 if (sum == 2 or 3 or 12) , then lost  
 if (sum == 4 or 5 or 6 or 8 or 9 or 10),  
 then (mypoint = sum) and  
 continue until sum = mypoint or 7  
 if (sum == mypoint), then won  
 if (sum =7), then lost

```

1 #include <stdio.h>
2 #include <stdlib.h>
3 #include <time.h>
4
5 enum Status {CONTINUE,WON,LOST};
6 int rollDice (void);
7
8 int main(void)
9 {
10     int sum;
11     int myPoint;
12
13     enum Status gameStatus;
14
15     srand( time(NULL));
16     sum=rollDice();
17
18     switch (sum)
19     {
20     case 7:
21     case 11:
22         gameStatus=WON;
23         break;
24
25     case 2:
26     case 3:
27     case 12:
28         gameStatus=LOST;
29         break;
30
31     default:
32         gameStatus=CONTINUE;
33         myPoint=sum;
34         printf("Point is %d\n",myPoint);
35         break;
36     }
37
38     while (gameStatus == CONTINUE)
39     {
40         sum=rollDice();
    
```

```

41
42     if (sum == myPoint)
43     {
44         gameStatus=WON;
45     }
46     else
47     {
48         if (sum == 7)
49         {
50             gameStatus=LOST;
51         }
52     }
53 }
54
55 if (gameStatus == WON)
56 {
57     printf("Player wins\n");
58 }
59 else
60 {
61     printf("Player loses\n");
62 }
63
64 system("pause");
65 return 0;
66 }
67
68 int rollDice(void)
69 {
70     int die1;
71     int die2;
72     int workSum;
73
74     die1=1+(rand()%6);
75     die2=1+(rand()%6);
76     workSum=die1+die2;
77
78     printf("Player rolled %d + %d = %d\n",die1,die2,workSum);
79     return workSum;
80 }
    
```

```
C:\c_code\ch3-21\Debug\ch3-21.exe
Player rolled 5 + 6 = 11
Player wins
請按任意鍵繼續 . . .
```

```
C:\c_code\ch3-21\Debug\ch3-21.exe
Player rolled 4 + 1 = 5
Point is 5
Player rolled 4 + 4 = 8
Player rolled 3 + 6 = 9
Player rolled 4 + 3 = 7
Player loses
請按任意鍵繼續 . . .
```

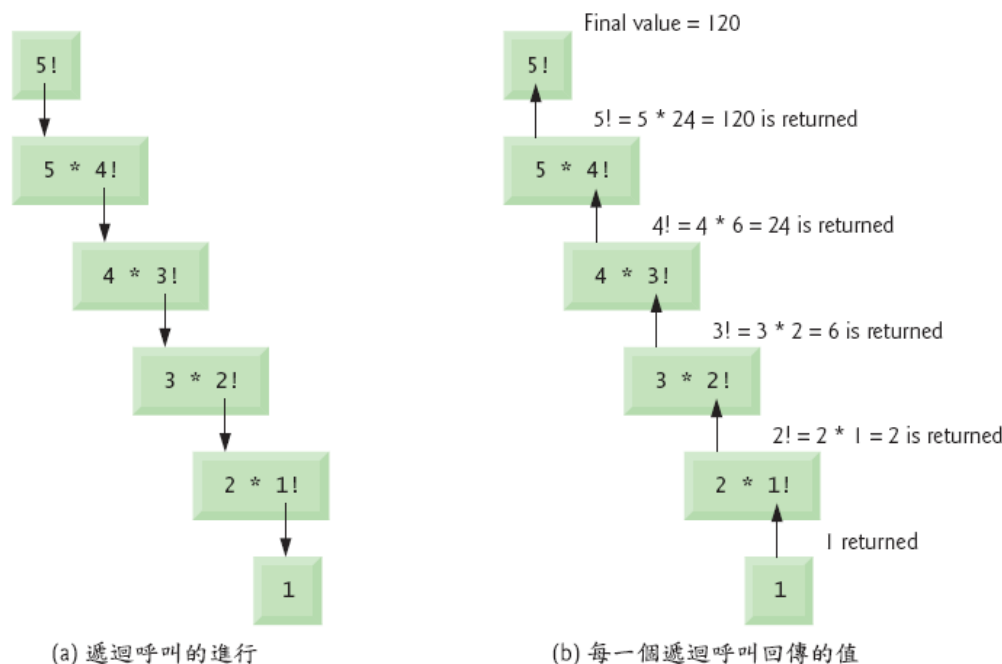
```
C:\c_code\ch3-21\Debug\ch3-21.exe
Player rolled 1 + 1 = 2
Player loses
請按任意鍵繼續 . . .
```

```
C:\c_code\ch3-21\Debug\ch3-21.exe
Player rolled 5 + 5 = 10
Point is 10
Player rolled 6 + 1 = 7
Player loses
請按任意鍵繼續 . . .
```



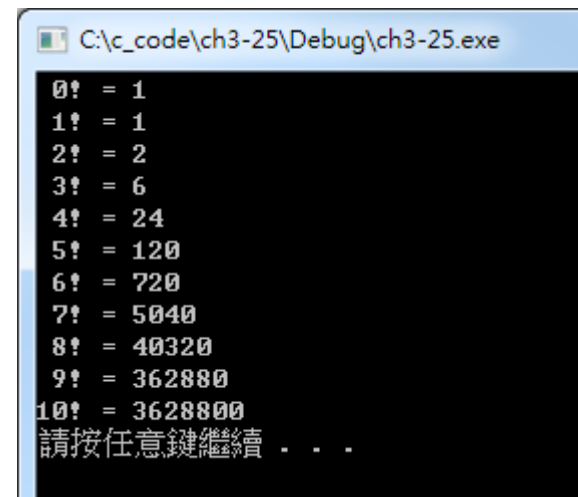
- `int rollDice (void)`
  - **Function:** 自己設計 `rollDice function` 來投擲骰子，並計算印出骰子的點數和回傳總和
- `enum Status {CONTINUS, WON, LOST}`
  - 為增加程式的 **可讀性**，使用列舉 (`enum`) 自己定義型別 (`CONTINUS, WON, LOST`)，目前定義常數 **CONTINUE** 的值是 **0**，**WON** 的值是 **1**，以及 **LOST** 的值是 **2**
- `enum Status gameStatus;`
  - `gameStatus = WON` (程式容易閱讀)
  - `gameStatus = 1` (程式不容易閱讀)
- `srand (time(NULL)) and rand()`
  - 隨機產生變數

- recursive function
  - 自己呼叫自己，一直做重複的事情，當結果已完成即可結束
- 階乘函式的遞迴定義，可以經由觀察下列關係而得：  
$$n! = n \cdot (n - 1)!$$

圖 5.13  $5!$  的遞迴式求值法

- 階乘函式的 $n!$ 遞迴

```
1 #include <stdio.h>
2 #include <stdlib.h>
3
4 long factorial(long number);
5
6 int main(void)
7 {
8     int i;
9     for (i=0; i<=10; i++)
10     {
11         printf("%2d! = %1d\n", i, factorial(i));
12     }
13     system("pause");
14     return 0;
15 }
16
17 long factorial (long number)
18 {
19     if (number <= 1)
20     {
21         return 1;
22     }
23     else
24     {
25         return (number * factorial(number-1));
26     }
27 }
```



```
C:\c_code\ch3-25\Debug\ch3-25.exe
0! = 1
1! = 1
2! = 2
3! = 6
4! = 24
5! = 120
6! = 720
7! = 5040
8! = 40320
9! = 362880
10! = 3628800
請按任意鍵繼續 . . .
```

- 製作三個函式（Odd、Even、TotalSum函式），功能分別為計算奇數和、偶數和、整數和。（其中的整數和請使用奇數和與偶數和之函式）

```
1 #include <stdio.h>
2 #include <stdlib.h>
3
4 int Odd(int U);
5 int Even(int U);
6 int TotalSum(int U);
7
8 int main(void)
9 {
10     int n, Sum;
11     char AddChoice;
12     printf("1+2+...+n=?請輸入n=");
13     scanf("%d",&n);
14     fflush(stdin);
15     printf("請問要做奇數和(O),偶數和(E),還是整數和(I)?請選擇:");
16     scanf("%c",&AddChoice);
17
18     switch(AddChoice)
19     {
```

1+2+...+n=?請輸入n=10  
請問要做奇數和(O),偶數和(E),還是整數和(I)?請選擇:I  
總合為55

```
20     case 'O':
21         Sum=Odd(n);
22         break;
23     case 'E':
24         Sum=Even(n);
25         break;
26     case 'I':
27         Sum=TotalSum(n);
28         break;
29     default:
30         printf("選擇錯誤\n");
31         return -1;
32
33     }
34     printf("總合為%d\n", Sum);
35     system("pause");
36     return 0;
37 }
38
39 int Odd(int U)
40 {
41     int i, total=0;
42     for (i=1; i<=U; i++)
43         if (i%2 == 1)
44             total=total+i;
45     return total;
46 }
```

1+2+...+n=? 請輸入n=10  
請問要做奇數和(O), 偶數和(E), 還是整數和(I)? 請選擇:I  
總合為55

Odd函式的定義，用來計算1+3+...+U  
的奇數和。(total是回傳值)

```
47
48 int Even(int U)
49 {
50     int i, total=0;
51     for (i=1; i<=U; i++)
52         if (i%2 == 0)
53             total=total+i;
54     return total;
55 }
56
57 int TotalSum (int U)
58 {
59     return Odd(U)+Even(U);
60 }
```

Even函式的定義，用來計算2+4+...+U的偶數和。  
(total是回傳值)

- 執行結果：

```
1+2+...+n=? 請輸入n=10
請問要做奇數和(O), 偶數和(E), 還是整數和(I)? 請選擇:I
總合為55
```

- 製作階層函式（**factorial**函式），功能為計算某一正整數的階層**k!**。並利用該函式求出 的值，**m**、**n**為任意正整數  $C_n^m = \frac{m!}{n!(m-n)!}$

```

1  #include <stdio.h>
2  #include <stdlib.h>
3
4  long int factorial(int p);
5
6  void main(void)
7  {
8      int m,n;
9      long int ans;
10     long int a, b, c;
11
12     printf("求排列組合C(m,n)\n");
13     printf("m=");
14     scanf("%d",&m);
15     printf("n=");
16     scanf("%d",&n);
17
18     a=factorial(m);
19     b=factorial(n);
20     c=factorial(m-n);
21

```

```

22     ans=a/(b*c);
23     printf("C(%d,%d)=%d\n",m,n,ans);
24
25     system("pause");
26 }
27
28 long int factorial(int p)
29 {
30     int count;
31     long int result=1;
32
33     for (count=1;count<=p;count++)
34     {
35         result=result*count;
36     }
37     return result;
38 }

```

求排列組合C(m,n)  
m=10  
n=8  
C<10,8>=45