



# 第六章

# 副程式及函數

# 6-4 自定函式(Function)



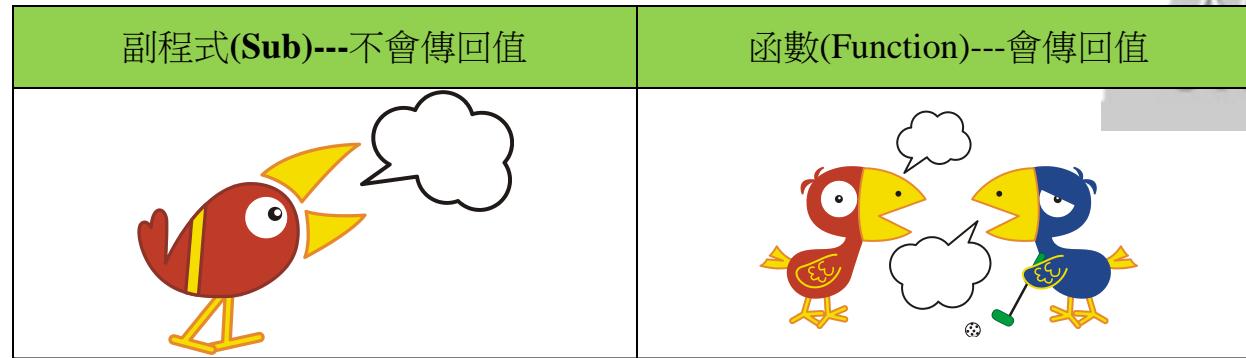
## 【定義】

是指依照解決問題的需要來自行定義的「副程式」，稱為「自定函式」。

## 【副程式(Sub)與函式(Function)之差異】

1. 副程式不會傳回值。
2. 函式可以指定傳回值的資料型態給主程式，若無則傳回值型別以 void 表示。

## 【示意圖】



【適用時機】內建函式沒有提供的功能，我們才有必要自定函式。

### 【函式的運作原理】

當「主程式」呼叫「函式」時，就會將執行控制權跳到該函式，等到函式執行完畢，再跳回主程式繼續未完的指令。

## (一) 主程式呼叫自定函式的語法

第一種寫法：

變數名稱=自定函數名稱(參數 1,參數 2,⋯);

【說明】

1. 將「自定函數名稱」直接指定給某一個「變數名稱」。
2. 「參數之串列」可以是「多個」數。

第二種寫法：

自定函數名稱(參數 1,參數 2,⋯);

## (二)自定函式的語法

```
static 傳回值資料型態 函式名稱(型態1 參數1,型態2 參數2,...)
```

```
{
```

```
-----  
-----
```

③

```
return 運算式的結果; //控制權回到呼叫函式的地方
```

```
}
```

【說明1】自定函式通常它會包含：

- ① 函式名稱
- ② 傳入參數(亦即形式參數)
- ③ 傳回值

其函式內的「程式運作過程」有如「黑盒子」。

【說明2】利用**return**來傳回值時，此函式並不需要等執行到右大括號”}”才會結束函式區塊。

【注意】主程式的「實際參數」名稱不一定要與自定函式的「形式參數」

名稱相同。

# 【實例】利用自定函數來計算 $1+2+\dots+10$ 的程式。

行號	程式檔名：ch6_4.java
01	<code>public class ch6_4 {</code>
02	<code>    public static void main(String[] args) //主程式</code>
03	<code>    { //宣告及設定初值</code>
04	<code>        int Sum, Max = 10;</code>
05	<code>        //處理</code>
06	<code>        Sum = MyFunction(Max); //呼叫自定函數</code>
07	<code>        //輸出</code>
08	<code>        System.out.println("1+2+...+10=" + Sum);</code>
09	<code>    }</code>
10	<code>    static int MyFunction(int X) //被呼叫的副程式</code>
11	<code>    {//宣告</code>
12	<code>        int i, total=0;</code>
13	<code>        //處理</code>
14	<code>        for(i = 1;i&lt;=X;i++)</code>
15	<code>            total = total + i;</code>
16	<code>        return total; //控制權回到呼叫函數的地方</code>
17	<code>    }</code>
18	<code>}</code>

【說明】主程式呼叫副程式時，會把實際參數X傳遞給副程式的形式參數，等到副程式全部執行完之後(計算出結果)，才會回到主程式”。

# 6-5 遞迴函數(Recursive)



## 【定義】

撰寫程式時，將程式模組化為一支獨立的函數，如果該函數可以反覆地自己呼叫自己，我們稱這個函數為遞迴函數。

## 【作法】

1. 由上而下將一個大問題切割成若干小問題。
2. 小問題的資料量是大問題的縮小版。

## 【最典型的例子】

在遞迴函數中，最典型的例子就是計算 $n$ 階乘的程式。

## 一、數學上：

### n階乘的概念如下：

題目： $n! = n \times (n-1) \times (n-2) \times (n-3) \times \dots \times 1$	舉例： $5! = 5 \times 4 \times 3 \times \dots \times 1$
$\begin{aligned} n! &= n \times (n-1)! \\ &= n \times [(n-1) \times (n-2)!] \\ &= n \times (n-1) \times [(n-2) \times (n-3)!] \\ &\quad \vdots \\ \therefore n! &= n \times (n-1) \times (n-2) \times (n-3) \times \dots \times 1 \end{aligned}$	$\begin{aligned} 5! &= 5 \times 4! \\ &= 5 \times (4 \times 3!) \\ &= 5 \times 4 \times (3 \times 2!) \\ &= 5 \times 4 \times 3 \times (2 \times 1!) \\ &= 5 \times 4 \times 3 \times 2 \times (1!) \\ &\quad \because 1! = 1 \\ \therefore 5! &= 5 \times 4 \times 3 \times 2 \times 1 \end{aligned}$

說明：在撰寫一個n階乘的遞迴函數程式時，則該函數將具備兩個主要特徵：

1. 該遞迴函數可以自己反覆地呼叫自己

(1) 第一次呼叫時的參數為n

(2) 第二次呼叫時的參數為n-1

(3) 第三次呼叫時的參數為n-2

(4) 參數的值會逐次遞減。

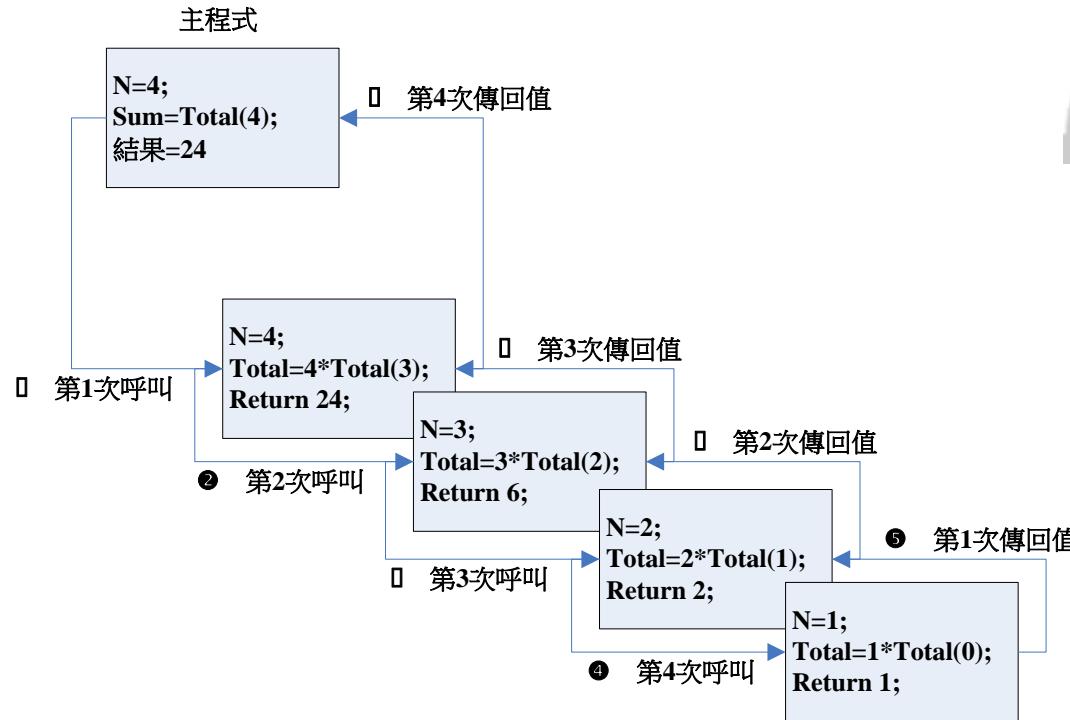
2.當參數值等於1時，必須停止遞迴呼叫。

## 二、演算法上：n階乘的概念如下：



```
Procedure fact(int n)
Begin
    int result;
    if(n == 1) then result = 1;
    else
        result = n * fact(n-1);
    return result ;
End
End Procedure
```

# 其遞迴函數呼叫的過程。如下圖所示：



## 【說明】

當主程式呼叫Total(4)時，第一次進入Total函數中，N=4，則執行Total=N\*Total(N-1)部份，因此，又是呼叫Total函數，所以N=3，則又執行Total=N\*Total(N-1)部份，直到N=0時，Total=1，才開始Return回到上一層，逐一的回到原先呼叫它的副程式，並且將值傳回去。

# 【實例一】遞迴函數呼叫 $10! = 1 \times 2 \times 3 \times 4 \times 5 \times 6 \times \dots \times 10$

【假設】階乘的公式如下：

$$f(n) = \begin{cases} 1 & , if \quad n = 1 \\ n \times f(n-1) & , if \quad n \geq 2 \end{cases}$$



## 【演算法】

演算法：n 階乘	
01	Procedure fact(int n)
02	Begin
03	int Result;
04	if(n == 1) then result = 1;
05	else
06	result = n * fact(n-1);
07	return Result ;
08	End
09	End Procedure

## 【撰寫程式】

行號	程式檔名：ch6_5A.java
01	<b>public class</b> ch6_5A {
02	<b>public static void</b> main(String[] args) //主程式
03	{ <b>//宣告及設定初值</b>
04	<b>int</b> Result, Max = 10;
05	<b>//處理</b>
06	Result = MyFunction(Max); //呼叫自定函數
07	<b>//輸出</b>
08	System.out.println("10!= " + Result);
09	}
10	<b>static int</b> MyFunction( <b>int</b> N) //遞迴函數名稱
11	{
12	<b>if</b> (N ==0) //遞迴函數的終值
13	<b>return</b> 1;
14	<b>else</b>
15	<b>return</b> N * MyFunction(N - 1); //函數自己又可以呼叫自己
16	}
17	}

## 【執行結果】

10!=3628800

由以上的例子中，我們可以清楚的得知，遞迴函數的呼叫過程是不能無限次的呼叫本身，否則會產生無窮迴圈。因此，基本上一個合乎演算法的遞迴函數必須要有下列條件：

### 1. 遞迴函數必須要設定「初值」與「終值」。

例如：以上例子中，行號04中的`Max=10`就是「初值」設定。

行號12中的`if (N == 0)`就是「終值」設定。

### 2. 遞迴函數必須要有更新值。

例如：以上例子中，行號15中的`MyFunction(N - 1)`。變數N的值每次都會遞減。不可以是常數。

### 3. 遞迴函數必須要自己呼叫自己。

例如：以上例子中，行號10的遞迴函數名稱(`MyFunction`)必須要與行號15的函數呼叫相同。

〔實例二〕利用遞迴方式來撰寫Fibonacci Number(費氏數)的程式。

【提示】Fibonacci Number(費氏數)

是指某一數列的第零項為0，第1項為1，其他每一個數列中項目的值是由本身前面兩項的值之和。

【假設】費氏數列的公式如下：

$$\text{Fib}(n)=\begin{cases} 0 & , \text{if } n=0 \\ 1 & , \text{if } n=1 \\ \text{Fib}(n-1)+\text{Fib}(n-2) & , \text{if } n \geq 2 \end{cases}$$

【說明】某一數為其前二個數的和。

【舉例】

假設我們現在想求出「費氏數列」第8項的費氏數列值。因此，我們就必須要先從第零項及第1項開始來推算，也就是第零項為0，第1項為1，開始計算。

## 【圖解說明】

n	0	1	2	3	4	5	6	7	8	.....
Fib(n)	0	1	1	2	3	5	8	13	21	.....

►數學上：Fibonacci(費氏數)的概念如下：

假設  $n_0=1$ ，  $n_1=1$ ， 則

$$n_2 = n_1 + n_0 = 1 + 1 = 2$$

$$n_3 = n_2 + n_1 = 2 + 1 = 3$$

⋮

⋮

$$\therefore n_i = n_{i-1} + n_{i-2}$$

►演算法上：Fibonacci(費氏數)的概念如下：

演算法：費氏數

```
01 Procedure Fib(int n)
02   Begin
03     if(n=0) return 0;
04     if(n=1) return 1;
05     if(n>=2) return Fib(n-1)+Fib(n-2);
06
07   End
08 End Procedure
```

## 【撰寫程式】

行號	程式檔名：ch6_5B.java
01	<code>public class ch6_5B {</code>
02	<code>    public static void main(String[] args) //主程式</code>
03	<code>    { //宣告及設定初值</code>
04	<code>        int N = 6,Sum;</code>
05	<code>        //處理</code>
06	<code>        Sum = Total(N); //呼叫自定函數</code>
07	<code>        //輸出</code>
08	<code>        System.out.println("Sum=" + Sum);</code>
09	<code>    }</code>
10	<code>    static int Total(int N) //函數名稱</code>
11	<code>    {</code>
12	<code>        if (N &lt;= 2) //遞迴函數的終值</code>
13	<code>            return 1;</code>
14	<code>        else</code>
15	<code>            return Total(N-2) + Total(N - 1); //函數自己又可以呼叫自己</code>
16	<code>    }</code>
17	<code>}</code>

## 【執行結果】

Sum=8

〔實例三〕利用遞迴方式來撰寫18與15的「最大公因數」之程式

【提示】一般在數學上找出最大公因數(Great Command Divisor)，大多是利用尤拉的輾轉相除法，而輾轉相除法，又名「歐幾里德演算法」( Euclidean algorithm )乃求兩數之最大公因數演算法。

### 【作法】

利用兩數反覆相除，直到餘數為0時，再取其不為0的除數，即為最大公因數。

### 【數學式】

$$GCD(A,B) = \begin{cases} B & , \text{若 } A \% B = 0 \\ GCD(B, A \% B) & , \text{若其他情況} \end{cases}$$

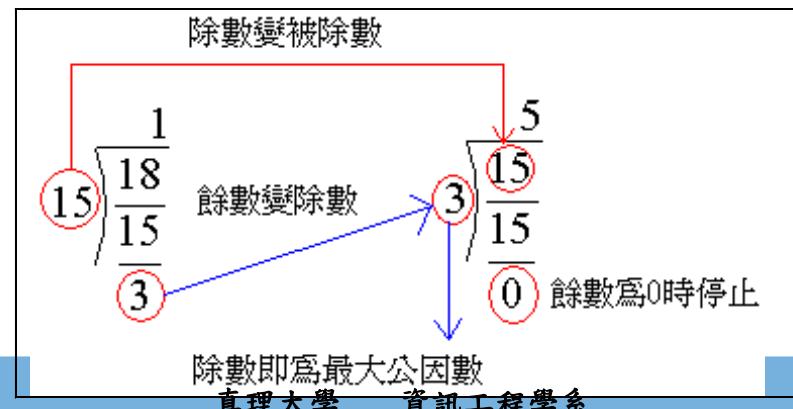
其中%代表取餘數

## 【演算法】

演算法：最大公因數

```
01 Procedure GCD (int a,int b)
02 Begin
03   c = a % b;
04   if (c == 0)           //判斷餘數是否為 0
05     return b;          //將取其不為 0 的除數，即為最大公因數
06   else
07     return GCD(b, c); //函式自己又可以呼叫自己
08   End
09
10 End Procedure
```

以下範例說明輾轉相除法過程：以18及15為例，必須要利用「輾轉相除法」，求最大公因數。其概念如下所示：



## 【撰寫程式】

行號	程式檔名：ch6_5C.java
01	<code>public class ch6_5C {</code>
02	<code>    public static void main(String[] args) //主程式</code>
03	<code>    { //宣告及設定初值</code>
04	<code>        int a=18,b=15;</code>
05	<code>        //處理及輸出</code>
06	<code>        System.out.println("MaxFactor=" + MaxFactor(a, b));</code>
07	<code>    }</code>
08	<code>    static int MaxFactor(int a,int b) //函數名稱</code>
09	<code>    {</code>
10	<code>        int c;</code>
11	<code>        c = a % b; //取餘數</code>
12	<code>        if (c == 0)</code>
13	<code>            return b;</code>
14	<code>        else</code>
15	<code>            return MaxFactor(b, c); //函數自己又可以呼叫自己</code>
16	<code>    }</code>
17	<code>}</code>

## 【執行結果】

MaxFactor=3



# 6-6 內建函數(Built-in)

【定義】內建函數就是在LeJOS程式中「預先寫好的程式」。

【目的】可以讓我們很方便的直接使用。

【例如】利用Math.random()亂數類別來取得某一特定範圍的亂數，以設計一台行駛方向或速度不規則的機器人。

【常見的內建函數】

1.字串函數

2.數學函數

3.亂數函數

4.聲音函數

5.繪圖函數

6.陣列函數

7.系統時間函數

以上各種內建函數的詳細介紹，請參閱第八章內容。

# 【課後評量】



1. 利用函數的方式，來設計圓的面積與周長。
2. 利用函數的方式，來設計 $X^2 + 2X + 1$ 的方程式。
3. 利用副程式的方式，來設計九九乘法表。
4. 利用函數的方式，將攝氏轉換成華氏。
5. 利用函數的方式，判斷是否為閏年。