



第六章

副程式及函數



本章學習目標

1. 讓讀者瞭解主程式與副程式的呼叫方式及如何傳遞參數。
2. 讓讀者瞭解副程式與函數的差異及如何自定函數。



本章內容

6-1 副程式(Subroutine)

6-2 傳值呼叫(Call By Value)

6-3 傳址值呼叫(Call By Address)

6-4 自定函式(Function)

6-5 遞迴函數(Recursive)

6-6 內建函數(Built-in)

6-1 副程式(Subroutine)



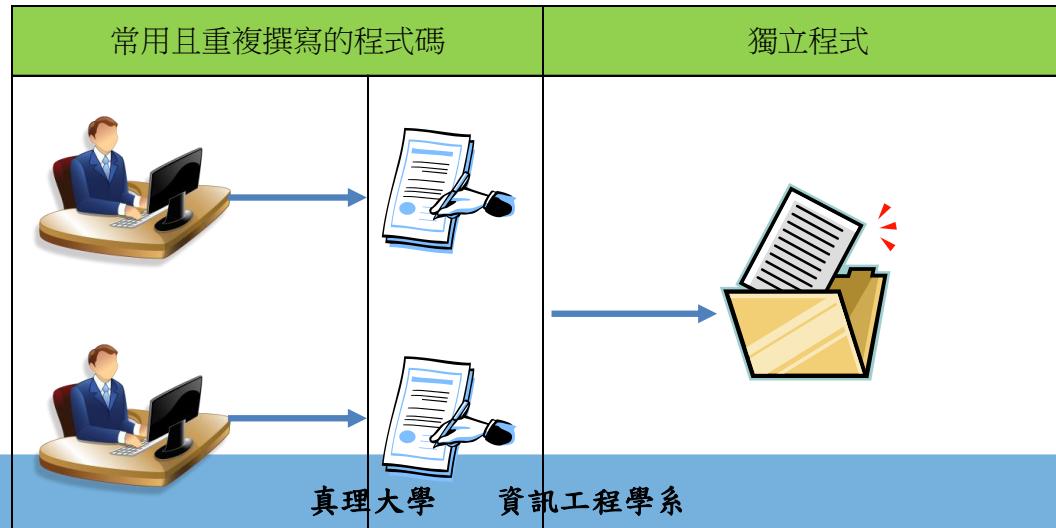
【引言】

當我們在撰寫程式時，都不希望重複撰寫類似的程式。因此，最簡單的作法，就是把某些會「重複的程式」獨立出來，這個獨立出來的程式就稱做副程式(Subroutine)或函數(Function)。

【定義】是指具有獨立功能的程式區塊。

【作法】把一些常用且重複撰寫的程式碼，集中在一個獨立程式中。

【示意圖】

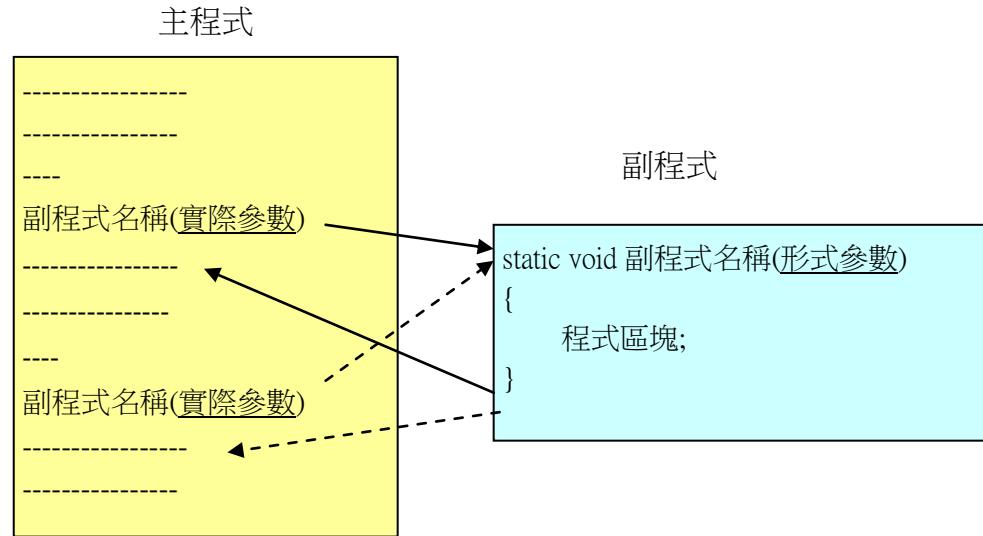


在Java程式語言中稱為「方法(Method)」。這樣可以避免一再重複撰寫相似的程式碼，讓程式看起來更有結構，有利於日後的維護。筆者為了讓讀者更容易閱讀本章節的內容，因此，在本文中所看到的「函數(Function)」也可以透過第八章「類別與物件」中的「方法成員」來建立」，亦即物件導向程式設計中的「方法」。

【副程式的運作原理】

一般而言，「原呼叫的程式」稱之為「主程式」，而「被呼叫的程式」稱之為「副程式」。當主程式在呼叫副程式的時候，會把「實際參數」傳遞給副程式的「形式參數」，而當副程式執行完成之後，又會回到主程式呼叫副程式的「下一行程式」開始執行下去。

【圖解說明】



【說明】

1. 實際參數→實際參數1, 實際參數2,....., 實際參數N
2. 形式參數→形式參數1, 形式參數2,....., 形式參數N
3. 定義副程式時，可分為兩種：一種是「有傳回值」，另一種則是「無傳回值」。

如果使用「無傳回值」時，就必須要在副程式名稱的前面加上「`void`」。



4. 主程式呼叫副程式時，不一定要傳遞參數。

5. 結束副程式的方式：

(1) 副程式執行到右大括號“ }”

(2) 副程式執行到return

【優點】

1. 可以使程式更簡化，因為把重覆的程式模組化。
2. 增加程式可讀性。
3. 提高程式維護性。
4. 節省程式所佔用的記憶體空間。
5. 節省重覆撰寫程式的時間。

【缺點】降低執行效率，因為程式會Call來Call去。

6-1.1 沒有傳遞參數的副程式



【定義】主程式呼叫副程式時，並沒有傳遞參數給副程式。

【優點】

1. 不需要額外宣告「實際參數」與「形式參數」。
2. 節省記憶體的堆疊空間。

【缺點】實用性與彈性較低。

【語法】

(一) 主程式呼叫副程式的語法

```
public static void main(String[] args)
{
    副程式名稱();
}
```

(二) 副程式的語法

```
static void 副程式名稱()
{
    程式區塊;
}
```



【實作】

請設計一個主程式呼叫一支副程式，如果成功的話，顯示「TestSub()
ok！」

【程式】

行號	程式檔名：ch6_1_1.java
01	<code>public class ch6_1_1 {</code>
02	<code> public static void main(String[] args)</code>
03	<code> { //主程式</code>
04	<code> MytestSub(); //呼叫副程式</code>
05	<code> }</code>
06	<code> static void MytestSub() //被呼叫的副程式</code>
07	<code> { //副程式</code>
08	<code> System.out.println("TestSub() ok!");</code>
09	<code> }</code>
10	<code>}</code>

【注意】主程式呼叫副程式時，不一定要傳遞參數，如上面的例子中，
主程式中的MytestSub()中並沒有參數的傳遞。



6-1.2 具有傳遞參數的副程式

【定義】主程式呼叫副程式的同時，「主程式」會傳遞參數給「副程式」。

【優點】提高副程式的實用性與彈性。

【缺點】

1. 需要額外宣告「實際參數」與「形式參數」。
2. 會佔用到記憶體的堆疊空間。

【語法】

(一) 主程式呼叫副程式的語法

```
public static void main(String[] args)
{
    副程式名稱(參數 1,參數 2,...);
}
```

的語法

```
static void 副程式名稱()(資料型態 參數 1, 資料型態 參數 2,⋯)  
{  
    程式區塊;  
}
```

【實例】請設計一個主程式呼叫一支副程式時，將主程式的實際參數傳遞給副程式的形式參數，並且副程式計算形式參數的總合。

【撰寫程式】

行號	程式檔名：ch6_1_2.java
01	<code>public class ch6_1_2 {</code>
02	<code> public static void main(String[] args)</code>
03	<code> { //主程式</code>
04	<code> int a=10;int b=20;</code>
05	<code> MyAdd(a,b); //呼叫副程式</code>
06	<code> }</code>
07	<code> static void MyAdd(int x,int y)</code>
08	<code> { //副程式</code>
09	<code> int sum=0;</code>
10	<code> sum = x + y;</code>
11	<code> System.out.println("x+y=" + sum);</code>
12	<code> }</code>
13	<code>}</code>

【執行結果】

x+y=30

【註】即使x與y參數的資料型態相同，也不可以合併宣告。否則會產生錯誤。

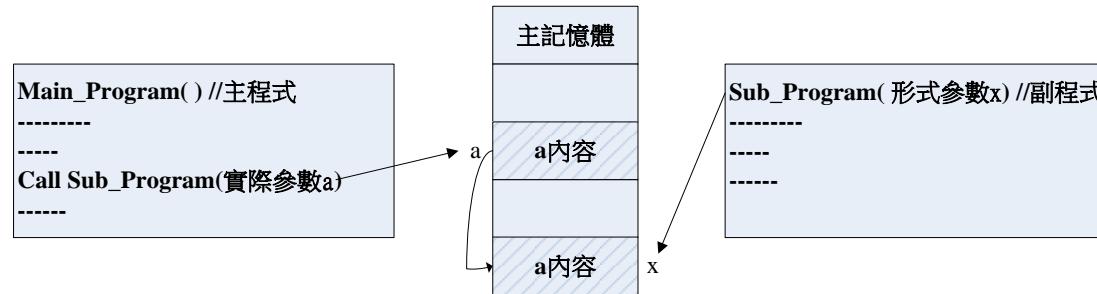
6-2 傳值呼叫(Call By Value)



【定義】

是指主程式呼叫副程式時，主程式會將實際參數的「值」傳給副程式的形參數，而不是傳送位址。

【運作原理】



【說明】在副程式中改變了形參數的值(內容)時，也不會影響到主程式的實際參數值(內容)。



【語法】

(一) 主程式呼叫副程式的語法

```
副程式名稱(參數 1, 參數 2, ...)
```

(二) 副程式的語法

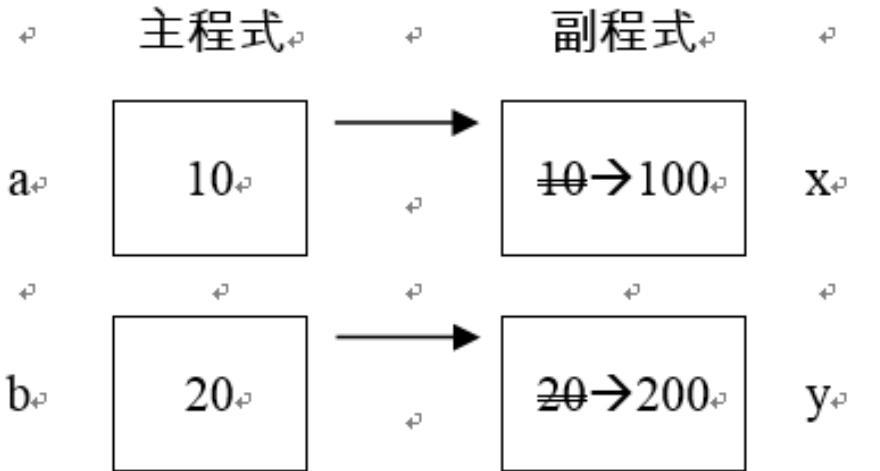
```
static void 副程式名稱(資料型態 1 參數 1, 資料型態 2 參數 2, ...)  
{  
    程式區塊;  
}
```

【說明】在副程式中的形式參數，可以是變數及陣列。

【實例】傳值呼叫

行號	程式檔名：ch6_2.java
01	<code>public class ch6_2 {</code>
02	<code> public static void main(String[] args)</code>
03	<code> { //主程式</code>
04	<code> int a=10;int b=20;</code>
05	<code> System.out.println("==Call Before==");</code>
06	<code> System.out.println("a=" + a + " " + "b=" + b);</code>
07	<code> CallByValue(a, b); //呼叫副程式</code>
08	<code> System.out.println("==Call After==");</code>
09	<code> System.out.println("a=" + a + " " + "b=" + b);</code>
10	<code> }</code>
11	<code> static void CallByValue(int x,int y) //被呼叫的副程式</code>
12	<code> { //副程式</code>
13	<code> x = 100;</code>
14	<code> y = 200;</code>
15	<code> }</code>
16	<code> }</code>

【執行過程】



【說明】當主程式呼叫副程式時，實際參數a所佔用的記憶體位址中的內容(值)會傳遞給副程式中的形式參數x，而實際參數b傳送給形式參數y，因為是傳值呼叫，所以主程式的實際參數與副程式的形
式參數不會佔用相同的記憶體位址。因此，副程式中的形式參數值改
變，也不會影響到主程式中的實際參數值。

【執行結果】



```
====呼叫之前====
```

```
a=10  b=20
```

```
====呼叫之後====
```

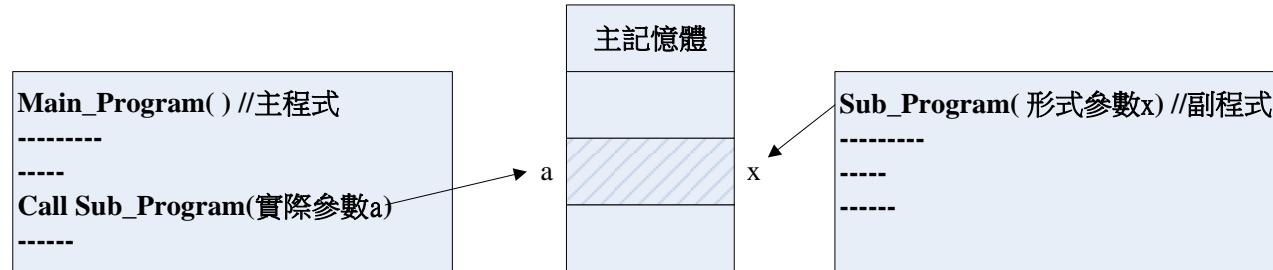
```
a=10  b=20
```

6-3 參考呼叫(Call By Reference)



【定義】是指主程式呼叫副程式時，主程式會將實際參數的「位址」傳給副程式的形式參數，使得主程式與副程式共用相同的記憶體位址。其中「位址」若宣告為陣列或物件時，則代表此種呼叫稱為「參考呼叫」。而在C語言中，稱此方式為「傳址呼叫(Call By Address)」。

【運作原理】



【說明】

一旦副程式中改變了形式參數的值(內容)時，也將隨之影響到主程式的實際參數值(內容)。因此，在使用參考呼叫時不能使用常數做為參數。

【語法】

(一) 主程式呼叫副程式的語法

```
副程式名稱(陣列名稱 1, 陣列名稱 2,⋯⋯)
```

(二) 副程式的語法

```
static void 副程式名稱(資料型態 1 陣列名稱 1, 資料型態 2 陣列名稱 2,⋯⋯)
{
    程式區塊;
}
```

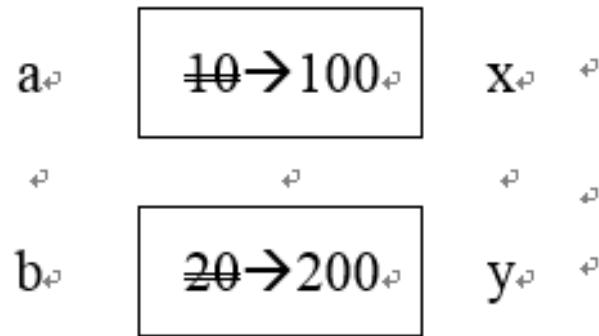
【實例】參考呼叫



行號	程式檔名：ch6_3.java
01	public class ch6_3 {
02	public static void main(String[] args)
03	{ //主程式
04	int []a = new int [1];
05	int []b = new int [1];
06	a[0]=10; b[0]=20;
07	System.out.println("==Call Before==");
08	System.out.println("a=" + a[0] + " " + "b=" + b[0]);
09	CallByAddress (a, b); //呼叫副程式
10	System.out.println("==Call After==");
11	System.out.println("a=" + a[0] + " " + "b=" + b[0]);
12	}
13	static void CallByAddress(int x[], int y[]) //被呼叫的副程式
14	{ //副程式
15	x[0] = 100;
16	y[0] = 200;
17	}
18	}

【執行過程】

主程式與副程式(佔用相同的記憶體位址)



【說明】

當主程式呼叫副程式時，實際參數a所佔用的記憶體位址中的內容(位址)會傳遞給副程式中的形式參數x，而實際參數b傳送給形式參數y，因為是參考呼叫，所以主程式的實際參數與副程式的形式參數會佔用相同的記憶體位址。因此，副程式中的形式參數值改變，也會影響到主程式中的實際參數值。

【執行結果】

```
====呼叫之前====
```

```
a=10  b=20
```

```
====呼叫之後====
```

```
a=100  b=200
```

