

第 4 章

列印在螢幕上

我們為我們的程式製作了優良的流程，但是我們並沒有探討要怎麼把我們的計算結果顯示出來。有許許多多的方法可以把結果顯示在螢幕上。在這本書中，我們將使用一個 Cocoa 所提供的函式：NSLog() 函式。這是個明智之舉，因為現在您將不用再為了要把您的結果 "列印" 到螢幕上而擔心（或者寫任何程式）。

NSLog() 函式主要是設計來顯示錯誤訊息，而不是用來輸出程式結果的。但是因為它太容易使用了，所以我們在這本書中調整了它的用法來顯示我們的結果。一旦您對 Cocoa 更熟悉之後，您將可以用更為精巧的技巧。

讓我們來看看 NSLog() 函式要怎麼使用。

```
[1]
int main()
{
    NSLog(@"Julia is a pretty actress.");
    return 0;
}
```

執行之後，範例 [1] 的述句會讓文字 "Julia is a pretty actress." 顯示出來。介於 "@" 與 " 之間的文字被稱為字串（string）。

除了字串本身之外，NSLog() 函式列印出了許多額外的資訊，例如目前的日期與程式名稱。舉例來說，在我的系統上程式 [1] 的完整輸出如下：

```
2005-12-22 17:39:23.084 test[399] Julia is a pretty actress.
```

一個字串可以包含零或多個字元（character）。

請注意：在接下來的範例中，只有 main() 函式中的重要述句才會被列出。

```
[2]
NSLog(@"");
NSLog(@" ");
```

述句 [2.1] 包含了零個字元，也被稱為空字串（empty string，亦即它的長度為零）。述句 [2.2] 並非空字串，儘管它看起來很像。它包含了一個空白（space），因此這個字串的長度是 1。

在字串中，一些特別的字元序列具有特殊的意義。舉例來說，要強制讓我們句子中的最後一個字列印到新的一行中，一個特殊的代碼必須包含到述句中 [3.1]。這個代碼是 `\n`，代表新行字元。

[3]

```
NSLog(@"Julia is a pretty \nactress.");
```

現在輸出結果看起來如下（只有相關的輸出被列出）：

```
Julia is a pretty  
actress.
```

在 [3.1] 中的反斜線稱為脫逸（escape）字元，它告訴 `NSLog()` 函式下一個字元不是一般用來列印到螢幕上的字元，而是有特殊意義的字元：在本例中 "n" 代表 "開始新行"。

在某些情況下您需要印一個反斜線到螢幕上，這好像會是個問題。如果一個位在反斜線後面的字元有特殊意義，那怎麼可能印出一個反斜線呢？嗯，我們只要在反斜線的前面（事實上是後面）放置另一個反斜線即可。這會告訴 `NSLog()` 函式這個（第二個）反斜線，亦即右邊多出來的這個，是要印出來的，而且它的任何特殊意義都應該要忽略。底下是一個範例：

[4]

```
NSLog(@"Julia is a pretty actress.\n");
```

述句 [4.1] 執行之後的結果會是

```
Julia is a pretty actress.\n
```

到目前為止，我們只有顯示固定的字串。現在讓我們把運算之後的結果印到螢幕上。

[5]

```
int x, integerToDisplay;
```

```
x = 1;
```

```
integerToDisplay = 5 + x;
```

```
NSLog(@"The value of the integer is %d.", integerToDisplay);
```

請注意，在小括號中我們有一個字串、一個逗號以及一個變數名稱。這個字串包含了一些有趣的東西：`%d`。就像反斜線一樣，百分比字元 `%` 有特別的意義。它後面跟著一個 `d`（decimal number 的縮寫），執行之後，輸出值的 `%d` 所在的位置將會被插入逗號之後的東西，亦即變數 `integerToDisplay` 的值。執行範例 [5] 可以得到

```
The value of the integer is 6.
```

要顯示浮點數，您必須用 `%f` 取代 `%d`。

[6]

```
float x, floatToDisplay;
x = 12345.09876;
floatToDisplay = x/3.1416;
NSLog(@"The value of the float is %f.", floatToDisplay);
```

要顯示多少位的有效數字（小數點後的部份）全取決於您。要顯示兩位的有效數字，您應該要把 .2 放到 % 與 f 之間，就像下面這樣：

```
[7]
float x, floatToDisplay;
x = 12345.09876;
floatToDisplay = x/3.1416;
NSLog(@"The value of the float is %.2f.", floatToDisplay);
```

稍後，當您知道如何重複計算之後，您也須會想要建立一個填了數值的表格。想像一個費氏與攝氏溫度轉換表。如果您要漂亮的顯示這些數值，您會希望兩欄資料的數值有著固定的寬度。您可以在 % 與 f 之間（或者 % 與 d 之間）用整數來指定這個寬度。然而，如果您指定的寬度比數值的寬度小時，將會採用數值的寬度。

```
[8]
int x = 123456;
NSLog(@"%2d", x);
NSLog(@"%4d", x);
NSLog(@"%6d", x);
NSLog(@"%8d", x);
```

範例 [8] 有著如下的輸出：

```
123456
123456
123456
 123456
```

在頭兩個述句 [8.1, 8.2] 中，我們宣告給要顯示的數值使用的空間太小了，於是空間就被自動擴充。只有述句 [8.4] 指定的寬度比這個數值寬，因此我們看到了額外的空白出現，表達出的是保留給這個數值的空間。

同時指定要顯示的寬度與小數的位數也是可行的。

```
[9]
float x = 1234.5678
NSLog(@"Reserve a space of 10, and show 2 significant digits.");
NSLog(@"%10.2d", x);
```

當然，要顯示多於一個，或任何混合多種型別的數值也是可以的 [10.3]。但是您必須確保有適當的使用 %d 與 %f 來標明資料型別 (int、float)。

[10]

```
int x = 8;
float pi = 3.1416;
NSLog(@"The integer value is %d, whereas the float value is %f.", x, pi);
```

對變數型別使用正確的符號是很重要的。如果您弄錯的第一個，第二個也可能無法正常的顯示！例如，

[10b]

```
int x = 8;
float pi = 3.1416;
NSLog(@"The integer value is %f, whereas the float value is %f.", x, pi);
```

給了下列的輸出：

```
The integer value is 0.000000, whereas the float value is 0.000000.
```

要執行我們的第一個程式現在只有一個疑問與一個解答。我們的程式怎麼認識這個函式 NSLog() 呢？嗯，它是不認識直到我們告訴它。要做到它，我們的程式必須告訴編譯器去匯入 (import) 包含 NSLog() 函示的適當函式庫，請使用下面的述句：

```
#import <Foundation/Foundation.h>
```

這個述句必須是我們程式的第一句述句。當我們把在本章中學到的所有東西放在一起之後，我們得到了下面的程式碼，我們將會在下一章執行它。

[11]

```
#import <Foundation/Foundation.h>
float circleArea(float theRadius);
float rectangleArea(float width, float height);

int main()
{
    float pictureWidth, pictureHeight, pictureSurfaceArea,
          circleRadius, circleSurfaceArea;
    pictureWidth = 8.0;
    pictureHeight = 4.5;
    circleRadius = 5.0;
    pictureSurfaceArea = rectangleArea(pictureWidth, pictureHeight);
    circleSurfaceArea = circleArea(circleRadius);
```

```
NSLog(@"Area of circle: %10.2f.", circleSurfaceArea);
NSLog(@"Area of picture: %f. ", pictureSurfaceArea);
return 0;
}

float circleArea(float theRadius) // 第一個自定函式
{
    float theArea;
    theArea = 3.1416 * theRadius * theRadius;
    return theArea;
}

float rectangleArea(float width, float height) // 第二個自定函式
{
    return width * height;
}
```