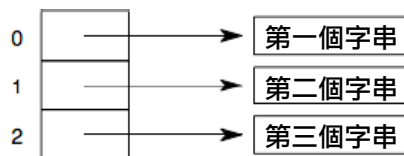


# 第 13 章

## 陣列

有時候您會需要控制一系列的資料。舉例來說，您可能需要維護一串的字串，不過假如每一個字串都各自使用一個變數將會有點累贅。當然，有個比較方便的解決之道：陣列（array）。陣列是一個有序的物件串列（或者更精確的說，它是一個物件指標的串列）。您可以把物件加入到陣列中、由陣列中移除物件，或者要求這個陣列讓您知道儲存在給定的索引位置（index，亦即給定的位置）中的是哪個物件，此外您也可以要求陣列讓您知道它含有多少個元素（element）。

當您數東西的時候，您通常由 1 開始。在陣列中，第一個項目的索引是零，第二個的索引是 1，依此類推。



範例：一個含有三個字串的陣列

我們在本章稍後會給您一些程式碼範例，讓您了解由零開始計數的影響。

陣列是由兩個類別所提供：NSArray 及 NSMutableArray。就像字串一樣，有不能修改與可以修改的版本。在本章中我們將考慮可以修改的這個版本。

一個建立陣列的方法是執行像下面這樣的算式：

```
[NSMutableArray array]
```

執行之後，這份程式碼會建立並傳回一個空的陣列。但是...等一下...這個程式碼好像怪怪的不是嗎？的確，在這個例子中我們使用了這個 NSMutableArray 類別的名稱來作為訊息的接收者。但是我們應該要傳送訊息給實體，而非類別，沒錯吧？

嗯，我們剛剛學到了一個新東西：事實上，在 Objective-C 中，我們也可以傳送訊息給類別（而原因是類別也是物件，它們是被我們稱為 *meta-classes* 的實體，不過我們不會在這篇導覽文章中更深入的探討這個觀念）。在 Cocoa 文件中，我們可以在類別上呼叫的方法是以 "+" 號來表示，而非我們平常會在方法名稱前面看到的 "-" 號（例如第 8 章 [4.5]）。舉例來說，在文件中我們看到對這個 *array* 方法的敘述：

**array**

+ (id)array

Creates and returns an empty array. This method is used by mutable subclasses of NSArray.

**See Also:** [+ arrayWithObject:](#), [+ arrayWithObjects:](#)

讓我們回頭來寫程式碼。下面的程式會建立一個空的陣列，然後在它裡面儲存三個字串，接著印出在這個陣列中的元素個數。

[1]

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

int main (int argc, const char * argv[])
{
    NSAutoreleasePool *pool = [[NSAutoreleasePool alloc] init];

    NSMutableArray *myArray = [NSMutableArray array];

    [myArray addObject:@"first string"];
    [myArray addObject:@"second string"];
    [myArray addObject:@"third string"];

    int count = [myArray count];

    NSLog(@"There are %d elements in my array", count);

    [pool release];
    return 0;
}
```

執行之後，這個程式會印出：

```
There are 3 elements in my array
```

下面的程式與前一個一模一樣，但是它印出來的是陣列中儲存在索引位置為 0 的字串。要取得這個字串，它使用了 *objectAtIndex:* 這個方法 [2.13]。

[2]

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

int main (int argc, const char * argv[])
{
```

```
NSAutoreleasePool *pool = [[NSAutoreleasePool alloc] init];

NSMutableArray *myArray = [NSMutableArray array];

[myArray addObject:@"first string"];
[myArray addObject:@"second string"];
[myArray addObject:@"third string"];

NSString *element = [myArray objectAtIndex:0]; // [2.13]

NSLog(@"The element at index 0 in the array is: %@", element);

[pool release];
return 0;
}
```

執行之後，這個程式會印出：

```
The element at index 0 in the array is: first string
```

您會時常需要走訪一個陣列來對陣列中的每一個元素做些事情。要做到這個效果，您可以像下面這個依照索引依序印出陣列元素的程式一樣使用迴圈來達成：

```
[3]
#import <Foundation/Foundation.h>

int main (int argc, const char * argv[])
{
    NSAutoreleasePool *pool = [[NSAutoreleasePool alloc] init];

    NSMutableArray *myArray = [NSMutableArray array];

    [myArray addObject:@"first string"];
    [myArray addObject:@"second string"];
    [myArray addObject:@"third string"];

    int i;
    int count;
```

```
for (i = 0, count = [myArray count]; i < count; i = i + 1)
{
    NSString *element = [myArray objectAtIndex:i];
    NSLog(@"The element at index %d in the array is: %@", i, element);
}

[pool release];
return 0;
}
```

執行之後，這個程式會印出：

```
The element at index 0 in the array is: first string
The element at index 1 in the array is: second string
The element at index 2 in the array is: third string
```

請注意陣列並不只限定於持有字串，它們可以持有任何您想要的物件。

`NSArray` 與 `NSMutableArray` 類別提供了許多其他的方法，而我們也鼓勵您去查詢這些類別的說明文件來學習更多關於陣列的知識。我們將討論一個可以讓您用一個物件來置換（replace）另一個位於給定索引的物件的方法，並以此為本章劃下句點。這個方法被命名為 `replaceObjectAtIndex:withObject:`。到目前為止，我們只處理過最多僅有一個引數的方法。但是這個不同，而這也是為何我們要在這裡看它的原因：它接受兩個引數。您可以由它名稱中所包含的兩個冒號而分辨出來。在 Objective-C 中，方法可以有任意數量的引數。以下是您要如何使用這個方法的做法：

```
[4]
[myArray replaceObjectAtIndex:1 withObject:@"Hello"];
```

執行這個方法後，位於索引 1 的物件會是字串 `@ "Hello"`。當然，這個方法只應該隨著一個有效（valid）的索引值來喚起。也就是說，必須有個物件已經儲存在我們給予這個方法的索引位置中，如此一來這個方法才能用我們傳遞給它的物件在陣列中進行置換的動作。如您所見，在 Objective-C 中的方法名稱就像一個有著空格（以冒號開頭）的英文句子一樣。當您要喚起一個方法時，您需要把真正的值填入這些空格中而形成一個具有意義的 "句子"。這個用來表示方法名稱與實行方法的做法是承襲自 Smalltalk，它也是 Objective-C 最大的優點，因為它讓程式碼非常具有表達力（expressive）。當您創造您自己的方法時，您也應該努力的以 "當它們被呼叫時會形成一個有表達力的句子" 為目標來為它們命名。這有助於讓 Objective-C 程式碼具有可讀性，對於將您的程式保持在容易維護的狀態而言是非常重要的。