Chapter 2  
與程式互動起來

上一章我們完成了第一個專案，與程式的世界打了一聲招呼。但是，那僅僅是在程式碼中預設的訊息，我們並沒有與程式互動，而是被動地接收程式輸出給我們的訊息。如果程式不能動態地讀入資料，對容易喜新厭舊的人類而言，絕對是看不上眼的。

這一章，我們將會教您如何讓程式接受使用者的輸入，並且針對不同類型的資料，作出不同的運算及反應。

##### 完成本章後，您可以學會:

* 藉由主控台介面傳遞資料給程式。
* 了解C#語言中不同的資料型態。
* 了解如何作邏輯判斷分支。

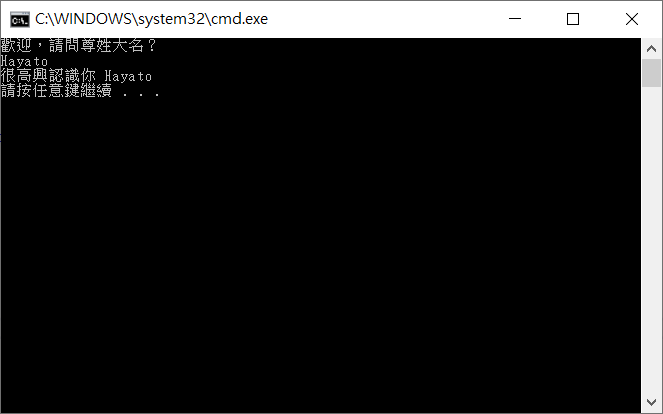
## 1.1 回到我們的Hello World

上一章的Hello World是諸位的第一個專案，我們當然不能這麼快就畫下句點，還沒完呢！

回到我們的Hello World專案，現在，我們希望修改Hello World程式，讓它執行起來能更親切友善，如同下面的流程：

1. 程式問使用者是誰？
2. 讓使用者輸入自己的姓名。
3. 程式問候使用者。

可以參考如下圖的執行結果：

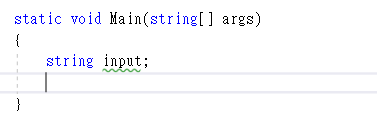


|  |  |
| --- | --- |
| paper&pencil_48 | 要達到這樣的效果，除了讓程式輸出訊息外，也要讓程式能讀入使用者輸入的姓名才可以，不能直接把姓名在程式碼中寫死阿。 |

## 1.2 變數的使用

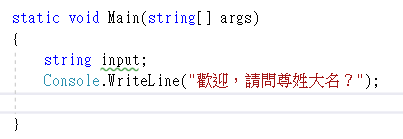
為了達到目的，我們需要使用一個變數來儲存使用者輸入的姓名。

在使用變數前，必須作宣告。由於C#是屬於強型別語言，宣告時必須要定義變數的型態以及變數的名稱，如下圖：

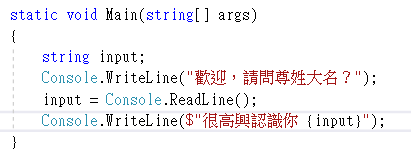


我們宣告了一個變數，型態是string(字串)，名稱是input。

接著，我們需要將詢問訊息輸出至螢幕，使用者才知道要做什麼互動。這部分，一樣使用我們才剛學會的Console.WriteLine()就可以了。



在使用者看到詢問訊息後，我們要請使用者輸入他的姓名，並再顯示出來。這裡我們會用到一個新的函式：Console.ReadLine()。語法如下：



Console.ReadLine()會從主控台中讀取一行使用者的輸入，接著我們把使用者輸入儲存到input變數中，再使用Console.WriteLine()輸出我們定義好的文字加上剛剛使用者輸入的姓名，按下執行就可以得出我們預期的結果了。

|  |  |
| --- | --- |
| lightbulb_48 | 提醒您，程式執行是不是一閃即過？現在除了使用Ctrl+F5執行以外，您也可以在程式的最後一行加上Console.ReadLine()，讓程式等待使用者按下輸入鍵後才結束。 |

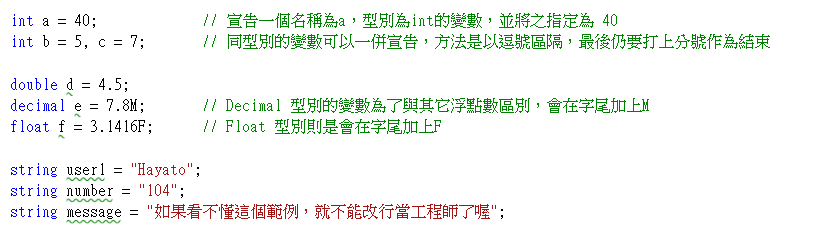
|  |  |
| --- | --- |
| lightbulb_48 | 上述範例中的第二個Console.WriteLine()，筆者使用了C#中字串內插補點的語法，在字串雙引號的前方，加上$符號，在之後的字串中，凡是被大括號包覆起來的區域，會輸出變數或運算結果。輸出結果等同於Console.WriteLine(“很高興認識你 ” + input);，但更簡潔。詳細的使用方法，也可上MSDN查詢：  <https://docs.microsoft.com/zh-tw/dotnet/csharp/language-reference/tokens/interpolated> |

### 1.2.1 變數的型別與變數的運算

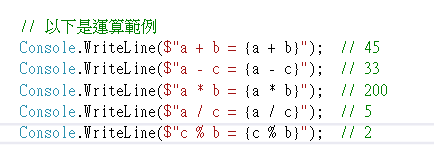
前面的範例，我們示範了如何讀入使用者輸入。使用者輸入的是字串，所以我們必須使用string型別的變數來儲存。事實上除了string，在C#中，定義了數種不同的資料型態。常使用的有：

* byte、int、long：整數
* float、double：IEEE二進位浮點數
* decimal：高精確度十進位浮點數
* char：字元
* bool：布林值(即true和false)
* string：字串

上述筆者只是列出C#型別中的一部分，以初學者而言，這已經很足夠練習了。不同型別的變數之間，不能互相作運算，如要運算，必須先透過轉型才行。以下讓我們看一些練習範例：



我們先宣告一些變數，有數值類(整數、二進位浮點數、十進位浮點數)，也有字串。接著，我們來試著對數值做點運算：



以上只是簡單的加減乘除，以及取餘數。這裡可以看到，由於 a 與 c 都是整數，所以相除後結果也會是整數，小數的部分就無條件捨去了。

|  |  |
| --- | --- |
| warning_48 | C#是編譯式語言，在執行前會先將程式碼透過編譯器編譯成可執行檔案(.exe)，如果程式碼在編寫時有誤，編譯器便可以在執行前將之揪出。然而，也有的情況是只有在執行期才會發生的錯誤(Runtime Error)，即使通過了編譯，也不代表程式必定可以順利執行。 |

|  |  |
| --- | --- |
| paper&pencil_48 | 錯誤清單區塊在編譯期僅列出錯誤訊息，在執行期還可多出動態監看式、呼叫堆疊、即時運算視窗等功能，是偵錯的好幫手。 |

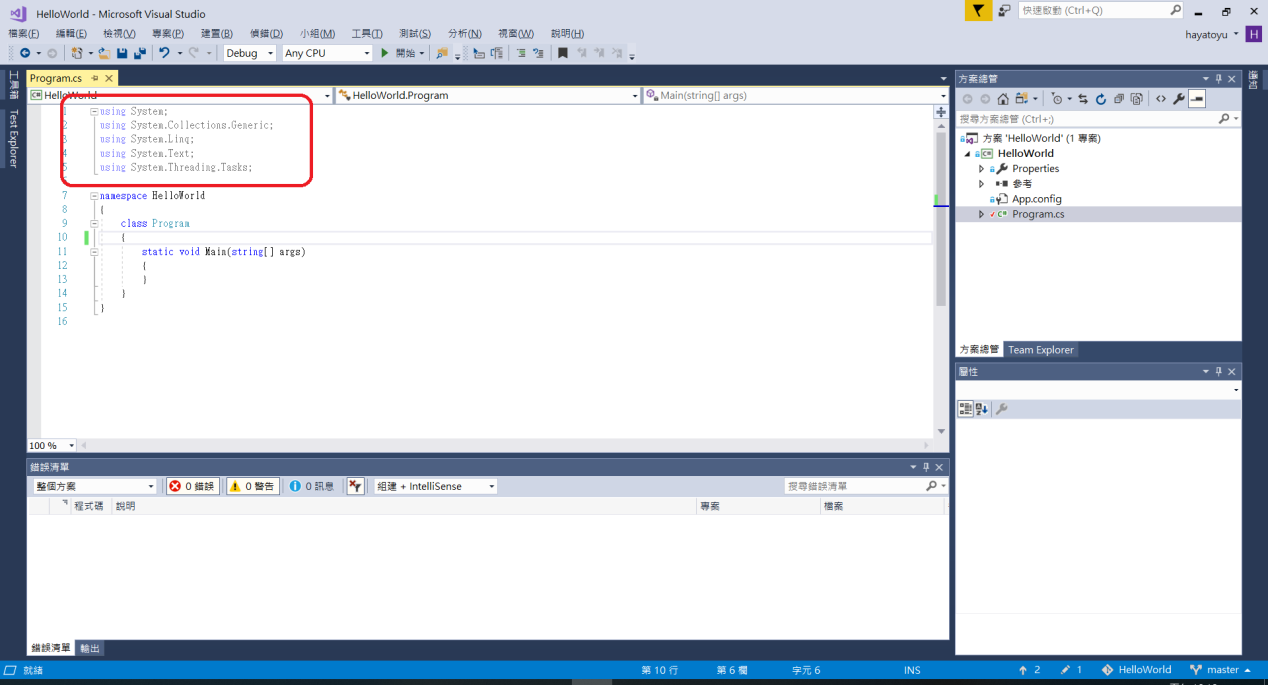
## 1.3 程式設計入門

稍微了解Visual Studio的功能後，我們還是要回到主題，此書旨在教導基礎程式設計，不能光介紹工具而已。現在讓我們來看看Program.cs吧。

### 1.3.1 引用函式庫與基本語法

.Net Framework內建相當多的函式庫，可供使用者在開發上直接引用。在程式碼編輯器的上方，可以看到有一堆using…的敘述句。這表示在此程式碼檔案中，我們將會引用這些函式庫協助我們進行開發，我們可以直接呼叫函式庫中所定義的類別及方法而不需要重新撰寫。

與許多著名的語言相同，在每一句敘述的結尾都會打上分號(;)，以表示該敘述句結束。

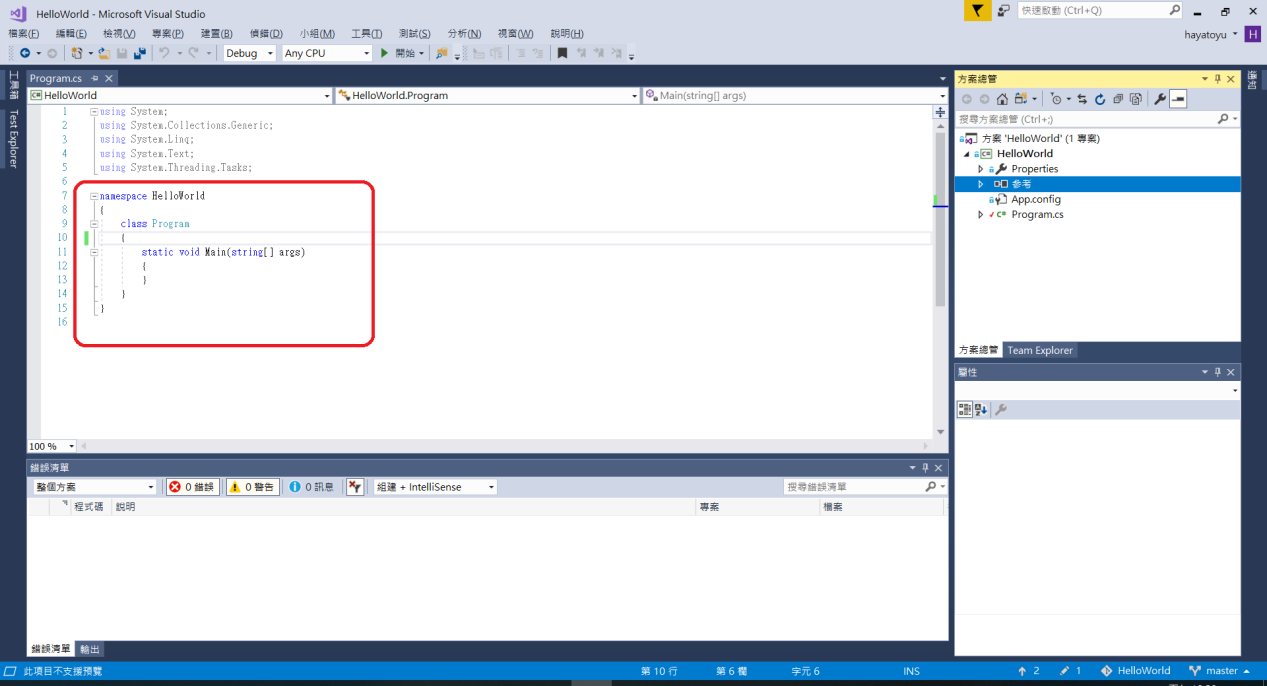


|  |  |
| --- | --- |
| warning_48 | 引用函式庫有可能造成函式庫名稱重複，此時編譯器便會提示。解決方法是可以另外命名函式庫的別名，或者辛苦一點直接以全名呼叫。 |

|  |  |
| --- | --- |
| paper&pencil_48 | 除了.Net Framework內建的函式庫外，也可以引用第三方函式庫或自己撰寫的函式庫(如Dll檔等)，方法是在方案總管中的參考上點滑鼠右鍵，選擇加入參考。 |

### 1.3.2 程式碼的撰寫

在引用的下方，從namespace開始，就是我們程式碼邏輯的區域。



每一個區域範圍，會使用大括號({})包覆。在此我們可以看到分為數個層級：命名空間(namespace)、類別(class)以及類別內的Main函式。

在此筆者先簡略的介紹一下，類別是物件導向語言的靈魂，也是最基本的操作單元。類別中可能有變數、屬性、建/解構子、函式等等，而類別需要實例化(Initialize)，才可在記憶體中配置空間並生成物件(object)。類別與物件的關係，就像你拿到了一台汽車的設計圖，它告訴你汽車該怎麼製造，有哪些零件及功能，但你光有汽車設計圖依然是當不了老司機的，你必須依照設計圖把這台車造出來，才有車可以開。設計圖就像是程式語言中的類別(class)，而你依照設計圖造出來的車，則是物件(object)。透過實例化，我們可以在記憶體中配置物件。想當然，你可以依照一張設計圖造出好多台車，所以一個類別也可以實例化出許多個物件，只是這些物件都必須遵守類別定義的規範而已。

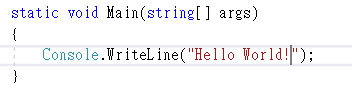
命名空間(namespace)中，可能有很多個類別，在同一個命名空間下的類別，可以互相參考。如果想參考不同命名空間下的類別，就必須先引用。

最後，這個專案的主角，Main函式。Main函式是程式的進入點，也就是程式在啟動時，會從Main函式開始執行。所以現在我們想完成這個專案，就必須把程式碼加入到Main函式中。

|  |  |
| --- | --- |
| paper&pencil_48 | 事實上類別、函式等在宣告時是可以定義存取範圍的，不一定都會被外部存取。筆者會在之後的章節說明。 |

### 1.3.3 Hello World！

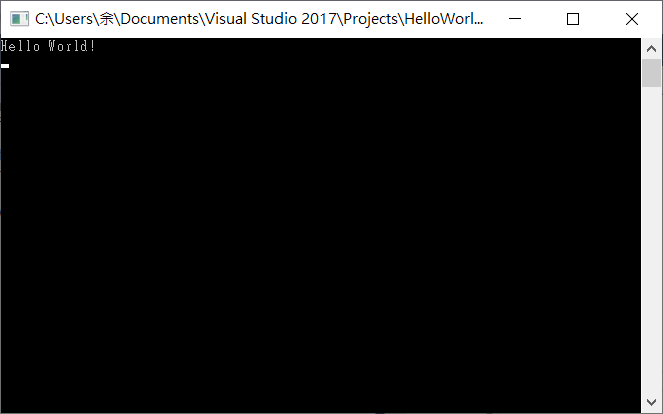
在Main函式中加入以下程式碼：



這行程式碼，呼叫了Console類別中的WriteLine()函式。Console類別是針對主控台應用程式的操作類，而其WriteLine()函式的功能，為將參數輸出至主控台，並在結束時換行。參數即為括號中以雙引號包覆的字串，也就是說這行程式碼的功能是在主控台上輸出Hello World字樣。

在編寫完成後，我們可以按下Visual Studio畫面上方的開始來執行程式。也可以直接點鍵盤上的F5來執行。

執行後的畫面應該如下：



|  |  |
| --- | --- |
| lightbulb_48 | 如果你發現主控台熟悉的黑色畫面一閃即逝，根本來不及看執行結果，那是因為程式一執行完就會自動關閉。您可以使用Ctrl + F5，讓程式執行完後點擊任意鍵再結束，不過這方法就不適用偵錯模式了。 |

|  |  |
| --- | --- |
| lightbulb_48 | 請注意，放在Console.WriteLine()中的字串一定要使用雙引號包覆。這樣編譯器才會判定它是字串資料。 |

至此我們的第一個 Hello World 專案便大功告成了，距離您學會程式設計，也就邁進了一大步了。在下一個章節，我們會介紹資料型態與變數的宣告，這會是極其無聊的一個章節，但我們還是希望您能耐著性子看完，我們相信這對您是會有幫助的。