Chapter 2  
與程式互動

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。

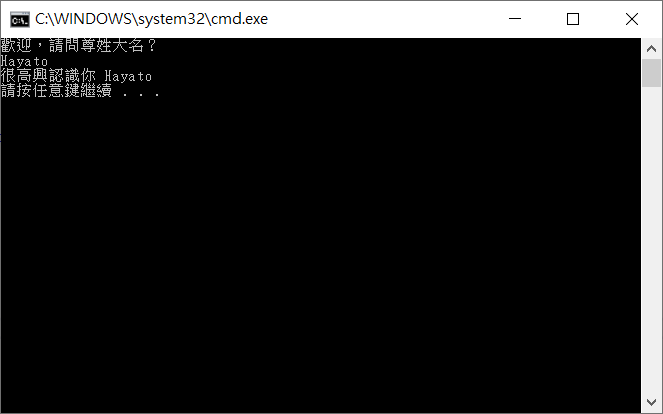
## 2.1 回到我們的Hello World

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：

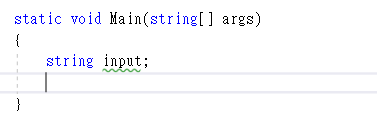


|  |  |
| --- | --- |
| paper&pencil_48 | 要達到這樣的效果，除了讓程式輸出訊息外，也要讓程式能讀入使用者輸入的姓名才可以，不能直接把姓名在程式碼中寫死阿。 |

## 2.2 變數的使用

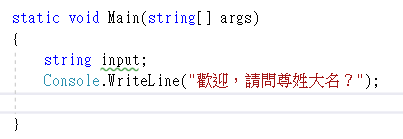
為了達到目的，我們需要使用一個變數來儲存使用者輸入的姓名。

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：

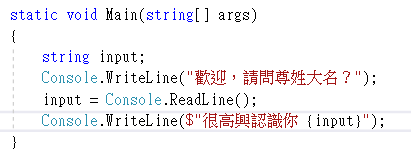


我們宣告了一個變數，型態是string(字串)，名稱是input。

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。 |

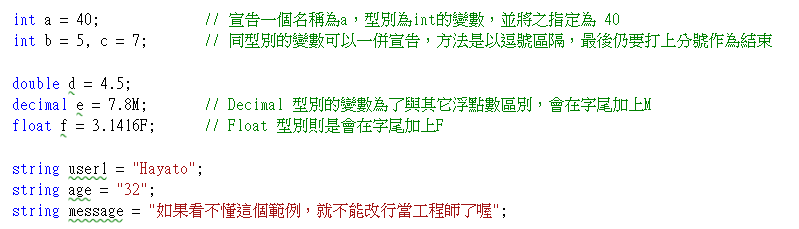
|  |  |
| --- | --- |
| lightbulb_48 | 上述範例中的第二個Console.WriteLine()，筆者使用了C#中字串內插補點的語法，在字串雙引號的前方，加上$符號，在之後的字串中，凡是被大括號包覆起來的區域，會輸出變數或運算結果。輸出結果等同於Console.WriteLine(“很高興認識你 ” + input);，但更簡潔。詳細的使用方法，也可上MSDN查詢：  <https://docs.microsoft.com/zh-tw/dotnet/csharp/language-reference/tokens/interpolated> |

### 2.2.1 變數的型別與變數的運算

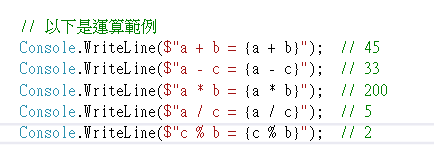
前面的範例，我們示範了如何讀入使用者輸入。使用者輸入的是字串，所以我們必須使用string型別的變數來儲存。事實上除了string，在C#中，定義了數種不同的資料型態。常使用的有：

* 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：

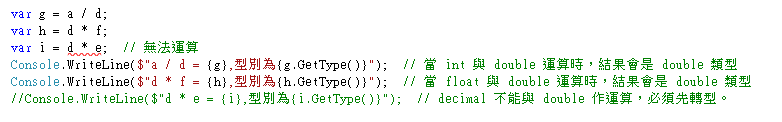


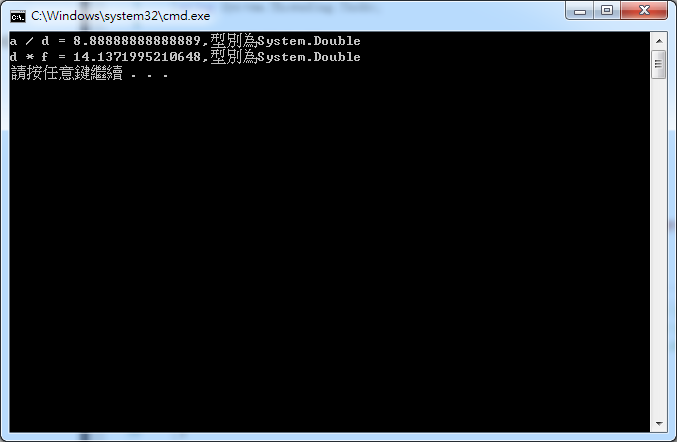
以上只是簡單的加減乘除，以及取餘數。這裡可以看到，由於 a 與 c 都是整數，所以相除後結果也會是整數，小數的部分就無條件捨去了。

|  |  |
| --- | --- |
| 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。  例如，以上述的例子而言，當我們把程式碼改為a / 0(除法不能除以0)，雖然在編譯期不會出現問題，程式執行時就會跳出DivideByZeroException例外而無法正常執行。 |

|  |  |
| --- | --- |
| lightbulb_48 | 加減乘除相信大家都知道，%的運算是取餘數，讀作modulus，例如 10除以7，商數是1，餘數為3，則 10 % 7 = 3。  另外，除法是不能除以0的，所以 % 一樣不能除以0。 |

接著我們來看一下，當不同類型的數值一同運算時，會發生什麼樣的事情。





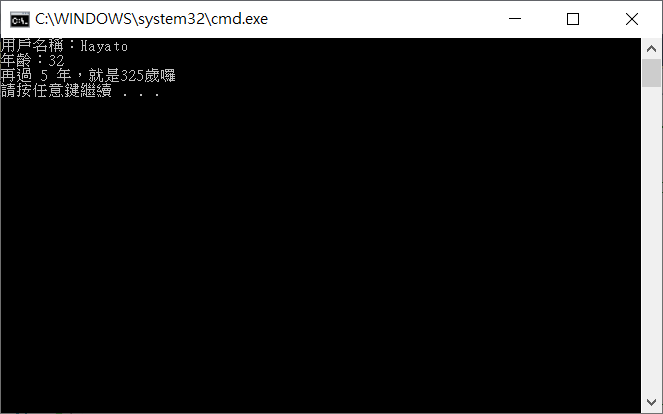
如上圖，當 int 與 float 碰到 double 時，計算出來的結果都會被改為 double 型別；而 decimal 型別是不能直接與 double 作計算的。

|  |  |
| --- | --- |
| paper&pencil_48 | Double和decimal明明都是浮點數，為什麼不能直接和decimal作運算？簡單的說，因為decimal(十進位浮點數)的精確度比double(二進位浮點數)來得高，所以double和decimal間不能直接運算。而float是double的子集合，所以可以直接隱含轉換。  那，什麼時候使用decimal呢？由於decimal的特性，非常適合在財務與金融計算上使用。畢竟，算錢的時候，要使用精確度較高的資料型別，才能坦蕩蕩避免起爭議嘛！ |

除了數值型別，我們也可以來看看字串型別的運算。



當我們試圖印出用戶五年後的年齡時，age + 5印出來的並不是 32 + 5 = 37，而是32和5連在一起的字串325。如下圖：

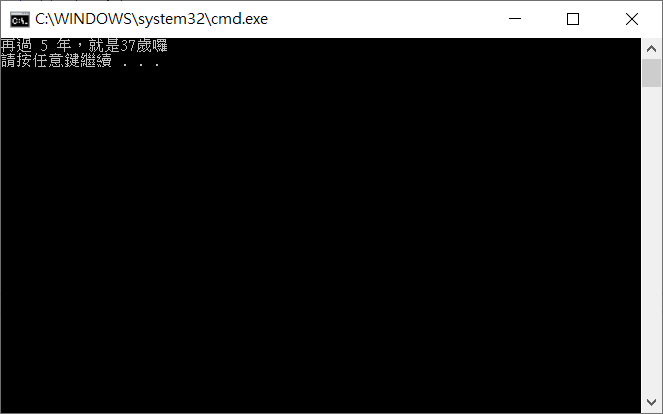


這顯然是有問題的，會造成這樣的原因，是因為age我們在宣告時，宣告為字串(string)型別，當放在Console.WriteLine()中時由於其特殊的機制，程式自動將age + 5 的5轉換為字串類型，而在字串間使用「+」的意思是將字串作拼接，所以輸出結果是325。

如果我們想讓字串型別的資料，像數值一樣作運算，就必須先將其轉型。如下圖：



使用Convert.ToInt32()函式將字串型別轉換成整數型別，這樣就可以運算了。



|  |  |
| --- | --- |
| paper&pencil_48 | Console.WriteLine()這個方法會使用複合格式，將物件的值轉換成文字表示，所以即使age是字串，直接與整數運算在Console.WriteLine()中是不會不能執行的，否則正常情況下編譯器會直接顯示錯誤。 |

|  |  |
| --- | --- |
| warning_48 | 轉型的方法很多，在C#語言中可分為隱含轉換、明確轉換、使用者定義轉換及使用協助程式類別轉換。上面的例子為使用協助程式類別轉換的一種。每種轉型的情況雖然不盡相同，但都有可能發生例外或資訊遺失，也就是程式有可能因為錯誤而中斷或轉換後的數值與原數值有誤差。例如我們如果想把一個字串”abc”轉型成整數類型，而使用Convert.ToInt32()，就會發生例外情況，因為原始字串並不能看成數值類型。 |

## 2.3 邏輯分支判斷

本章的最後一節，會提到邏輯分支判斷。

假設，我們設計一個諮詢程式，讓使用者輸入他的年收入(萬元)，並依照年收入的多寡來回應對該使用者增加年收入的建議。我們定義的條件如下：

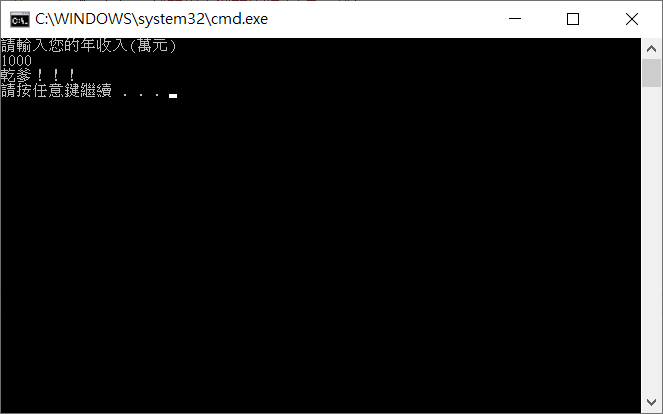
1. 年收入未達50萬：輸出「…我敢肯定這本書對你會有幫助的。」
2. 年收入50萬以上未滿100萬：輸出「…好好把這本書讀完，你就有機會再進一步。」
3. 年收入100萬以上未滿500萬：輸出「恭喜你，人生勝利組。」
4. 年收入500萬以上：輸出「乾爹！！！」

依照上述的規則，我們需要讀取使用者輸入的資料，並作出判斷來回應。本章開頭已經示範過如何讀取使用者輸入，還不熟悉的讀者請回頭看前兩章；而在邏輯分支判斷上，C#常用的語法有兩種，一種是if-else if-else，一種是switch。這邊的情境，我們較適合使用if-else if-else作處理，範例如下：



使用if時，需要使用小括號將敘述式包覆起來。而如果同時有多個條件，則可以用 ==(Equal)、&&(And)、||(Or)、!(NOT)及小括號作連接。當然有的型別也提供判斷是否相等或包含的函式，只要結果可得出一個布林值(真或假)的敘述式都可以。而條件成立要執行的程式碼區塊，則需使用大括號包覆起來。

最後，看看我們的執行結果吧。



|  |  |
| --- | --- |
| paper&pencil_48 | if條件之後不一定要直接接else條件，可以同時多個if或else if 並列，也可以沒有else條件。多個if條件並列，與if-else if條件並列的區別在於，如果是多個if條件並列，則每一個if條件都會判斷，如果同時符合多個if條件，則這些區塊都會被執行；如果是if – else if，程式會由上而下依序判斷，只要其中有一個條件成立，程式在執行過後就不會再執行這組if-else if中的其他條件區塊。  另外，else區塊作為與if相對應的條件區塊，並不需要明確的條件敘述，只要if不成立就會直接走進else區塊。在if-else if-else中，則是上述所有if及else if條件都不成立的話，就會執行else區塊。在設計上應想好是不是有一個預設的機制在所有條件都不成立時要執行，如果有的話，就應該放在else區塊中，反之則應檢查else中是不是有不應放入的程式碼，造成程式邏輯錯誤。 |

|  |  |
| --- | --- |
| paper&pencil_48 | 您沒有看錯，我也沒有打錯，在邏輯分支中，等於就是以兩個等號「==」來作判斷是否相等，這不是網路聊天用的表情符號。其他像「且」則是以兩個「&&」判斷兩個敘述式是否同時成立，「或」則是以兩個「||」(這個鍵通常是跟「\」在同個按鈕上，請按住shift鍵打出)判斷，只要有其中一個條件成立即可。  And與Or存在捷徑運算，對And來說，只要第一個敘述式為假，就不會再判斷第二個；Or則是第一個敘述式為真，就不會判斷第二個。 |