

壹、課程說明

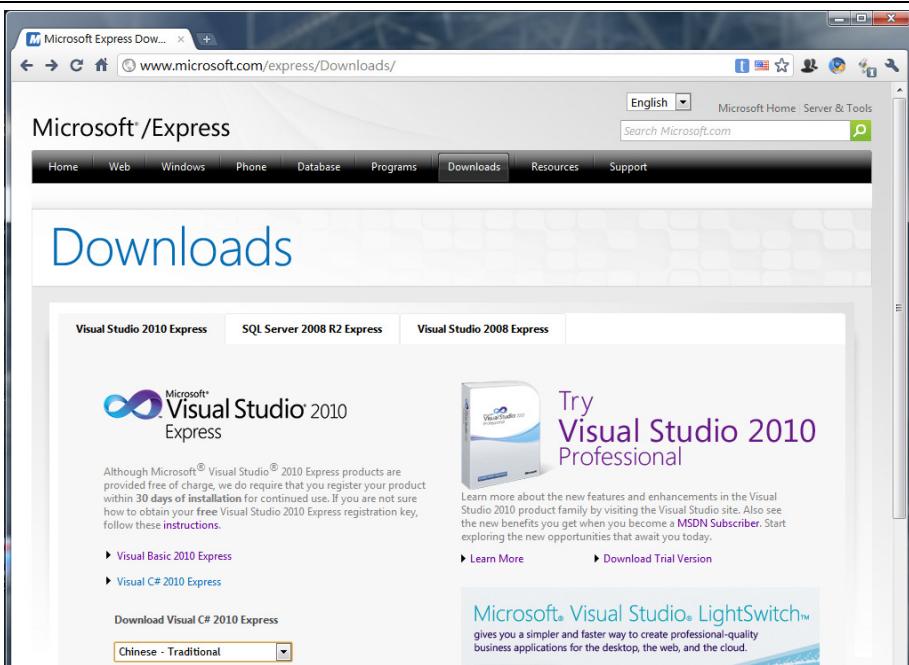
單元名稱	Visual C#程式設計
單元摘要	<ol style="list-style-type: none">1. Visual C#介紹。2. 主控台應用程式介紹。3. 視窗應用程式介紹。4. 程式範例介紹。
設計者	李啟龍 教師 (國立台灣師大附中)
學習目標	<ol style="list-style-type: none">1. 瞭解 Visual C#程式語言。2. 瞭解如何開發主控台應用程式。3. 瞭解如何開發視窗應用程式。4. 學習 Visual C#程式的開發。
課綱範圍	電腦與問題解決 專題製作課程
教學節數	6 節
先備知識	<ol style="list-style-type: none">1. 修畢資訊科技概論。2. 具有程式設計的觀念。
評量方法	課堂觀察。 口頭問答。 上機實作。 紙筆測驗。
參考資源	<p>參考書籍：</p> <ul style="list-style-type: none">■ 位元文化，Visual C++ 2010 Express 入門進階(附光碟)，松崗資訊。■ 李啟龍，Visual C# 2010 程式設計 16 堂課，碁峰資訊。■ 張書源，Visual C# 2010 與 UML 開發實戰，悅知文化。■ 黃嘉輝，Visual C# 2010 網路程式設計之道(附光碟)，碁峰資訊。■ 鄧文淵，Visual C# 2010 程式設計速學對策，碁峰資訊。

貳、教學計畫

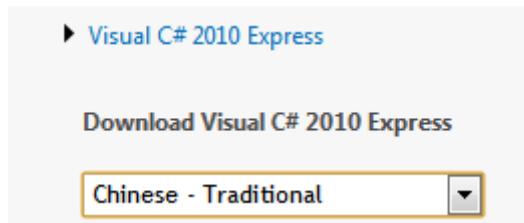
教學活動	時間	說明
第一堂課： C#簡介		
1.程式語言與C#簡介	15 分	<p>程式語言就跟中文、英文這些語言一樣，用來當作與電腦溝通的媒介，例如當我們想與美國人對話時，可能會使用英文，那如果我們想和電腦溝通時，我們就得使用程式語言，讓電腦幫助我們完成想做的事情。</p> <p>程式語言的種類非常多，例如被歸類為低階語言的機器語言、組合語言...等等，或是被歸類為高階語言的：C/C++、Java、Python、C#、Basic...等語言。其中低階語言在電腦中的執行效率比高階語言的效率高，且對於電腦硬體的控制程度也較高，不過由於低階語言的語法結構與人類的語言使用習慣相去較遠，故低階語言</p>

		<p>在開發、閱讀與維護上，都比高階語言困難許多。而高階語言則是較偏敘述性的語言，其語法結構與人類的語法邏輯使用習慣較為接近，因此也比較易於開發、閱讀、除錯與維護，也是一般專案開發時較常使用的程式語言種類。然而高階語言的種類繁多，C#語言又有什麼特色值得我們選擇使用呢？</p> <p>C#語言起源於微軟的.NET架構計畫，當時微軟內部開發.NET架構使用的程式語言，就是C#語言的前身，也可以說C#語言是為了.NET架構而生的產物。相對於一般校園或公司開發使用率較高的C/C++或是Java、Basic語言來說，C#可以說是非常年輕的程式語言。</p> <p>微軟在設計C#時，廣納百川並擷取各既存語言的優點，努力推動C#成為微軟新一代程式設計的標準語言。從C#名稱上就可以看出來，C#語言建立在C/C++語法基礎上，然而C#也兼有Java的精簡、嚴謹以及C++/Java兩種語言都強調的物件導向概念，C#一次包含了幾乎所有C++與Java當中你能想到的特色與功能。另外在.NET架構的推動下，微軟也透過Visual C#程式設計軟體讓C#同時擁有和Visual Basic一樣簡單方便的微軟視窗程式設計能力。</p> <p>我們將C#語言的優點條列如下：</p> <ol style="list-style-type: none"> 1. 程式初學者或指導者常常在C/C++語言以及Visual Basic語言之中猶豫，C#語言正好兼有兩者的優點，學C#可以讓我們透過Visual C#得到與Visual Basic一樣強大的Windows視窗程式開發能力，由於語法與C/C++、Java幾乎相同，也不用擔心以後與主流C/C++語言、Java語言的銜接問題。 2. 對於已經有C/C++或是Java基礎的程式設計師來說，轉換到C#平台更是沒有障礙，程式語法與使用概念幾乎相同，使用Visual C#更大幅簡化了在Windows底下開發視窗程式的難度，由於Windows與C#都出自於微軟之手，使用C#較其他語言，更能得到Windows作業系統的原生支援。 3. C#語言已經成為微軟的新一代開發標準，如微軟新一代網頁設計方案Web Developer或是與Adobe Flash對抗的Silverlight，還有能夠設計PC、Xbox360遊戲的Game Studio軟體套件，C#語言都對這些微軟的新產品，有著全面或甚至獨佔性的支援。
2. 簡介微軟的.NET架構	15分	<p>.NET架構(.NET Framework)是微軟從Visual Studio程式開發平台2002版後，開始為Windows程式加入的軟體開發架構，當時所使用的是「.NET 1.0版」，到了2005版時採用的版本是「.NET 2.0版」，至於目前最新2010版Visual Studio程式開發平台所採用的版本已經是「.NET 4.0版」了。</p> <p>那麼「.NET架構」到底是什麼？我們可以將「.NET架構」大略的比喻成一個精通各種語言的厲害廚師，當饕客從世界各地前來想要品嚐美味，他們可以直接用自己熟悉的母語和他溝通，請廚師做出他們想要的菜。在點菜的過程中，饕客不需要知道廚師的烹調方法、不需要知道使用何種廚具…等瑣碎的事情，饕客只需要告訴廚師想吃的是什麼，「.NET」大廚便會把食物做出來上菜。</p> <p>「饕客」可以透過自己熟悉的「母語」向「廚師」點菜，換句話說，也就是程式設計師可以使用自己熟悉的程式語言透過.NET架構來寫程式。在這個比喻中</p>

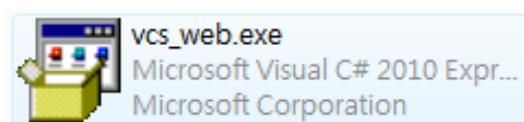
		<p>「.NET 架構」可以想像成廚師，「程式設計師」可以想像成饕客，「母語」可以想像成程式設計師所熟悉的各種程式語言。</p> <p>在.NET 架構下，程式從開發到執行的方式如下：</p> <ol style="list-style-type: none"> 1. 程式設計師可以使用任何熟悉的程式語言來開發.NET 程式，如：Visual C#、Visual C++、Visual Basic ...等等。開發軟體時，無論是使用哪種語言，都可以套用.NET 所提供方法庫中的各種內建功能。 2. 程式開發完成後，在編譯的過程，會把不同語言所寫的程式，轉為相同的中繼語言(Common Intermediate Language)。 3. 程式被執行時，.NET 架構中的共通語言執行環境 CLR(Common Language Runtime)會根據執行環境來進行編譯，將中繼語言一一轉換成電腦上可以執行的機器語言並且執行之。 <p>由於所有程式碼的執行都由 CLR 負責執行，故 CLR 在.NET 架構中同時還扮演著保護使用者電腦，不受不當程式碼破壞的保全角色，以及自動辨識不再使用的記憶體區塊並加以釋放(Garbage Collection)的清理者角色。</p> <p>總結.NET 架構所帶來的好處主要有：</p> <ol style="list-style-type: none"> 1. 不同程式語言之間可以相容，因為不同的程式語言，都會被轉為相同的中繼語言。 2. 提供非常多且實用的方法庫，透過呼叫方法庫即可完成開發軟體時所需的常用功能，不需重覆撰寫。 3. 可以避免撰寫好的程式在不同的電腦中執行時，會因為環境不同(例如使用不同的 CPU 指令...等)而造成錯誤，提高軟體的相容性。 <p>因此透過.NET 架構，程式撰寫者可以更加專注在程式的撰寫上，處理真正核心問題，而不用再費心於開發程式的常用功能或處理程式的相容性問題上。</p>
3.下載與 安裝 Visual C# 2010 Express	20 分	<p>下載 Visual C# 2010 Express 安裝檔與安裝的程序說明如下：</p> <p>Step1：請直接在瀏覽器輸入網址 http://www.microsoft.com/express/Downloads/，即可到達下載頁面。</p>



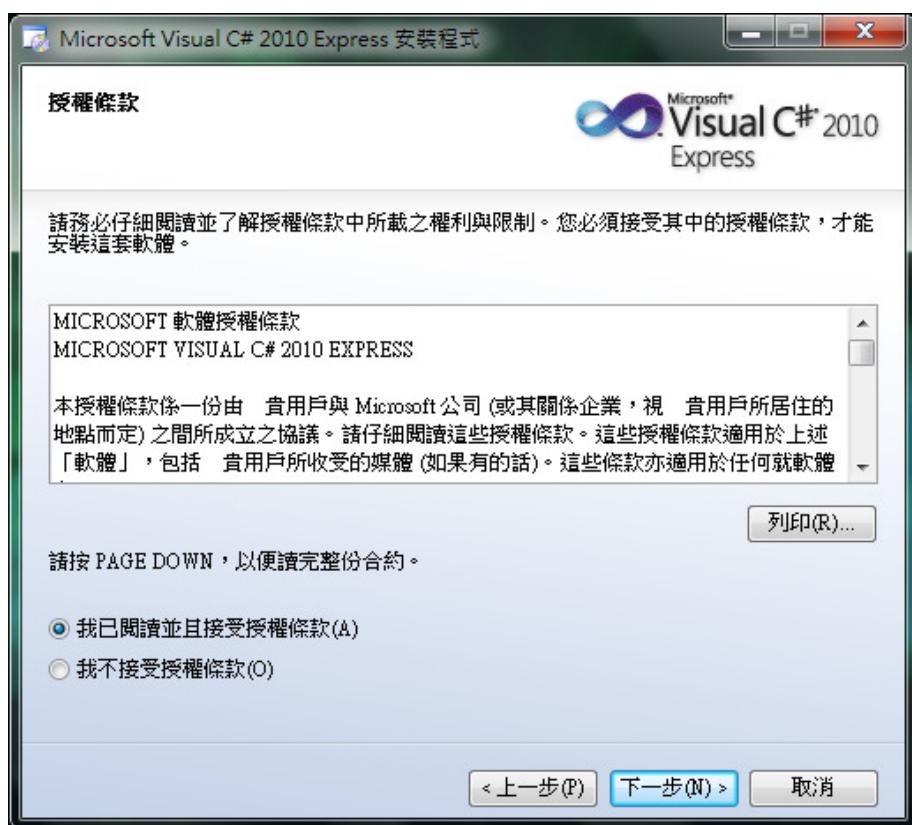
Step2：Visual C# 2010 Express 提供多國語系版本。若要下載繁體中文版，請在下載的頁面上，選擇「Chinese - Traditional」選項，就可以下載繁體中文版。選取好「Chinese - Traditional」選項後，瀏覽器就會自動開始下載安裝檔案。



Step3：下載完成後會得到一個安裝檔「vcs_web.exe」，用滑鼠左鍵雙擊即可開始安裝。



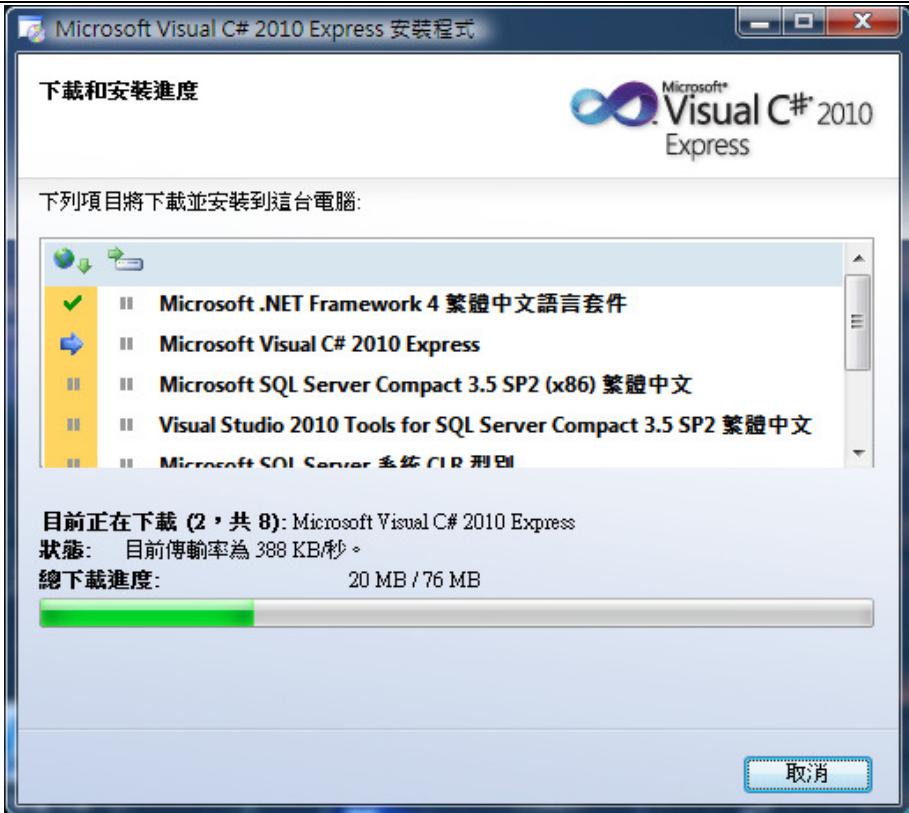
Step4：安裝的過程大致都只需按「下一步」按鈕即可進行，但其中幾個步驟可選擇的安裝細節，必須稍微注意一下。首先，在「使用者授權合約」部分，我們要記得勾選「我已閱讀並且接受授權合約中的條款」選項，才能繼續安裝。



Step5：選擇安裝的目的資料夾，預設的資料夾為「C:\Program Files\Microsoft Visual Studio 10.0\」，使用者也可以依據使用需求，另外設定檔案安裝的資料夾位置。



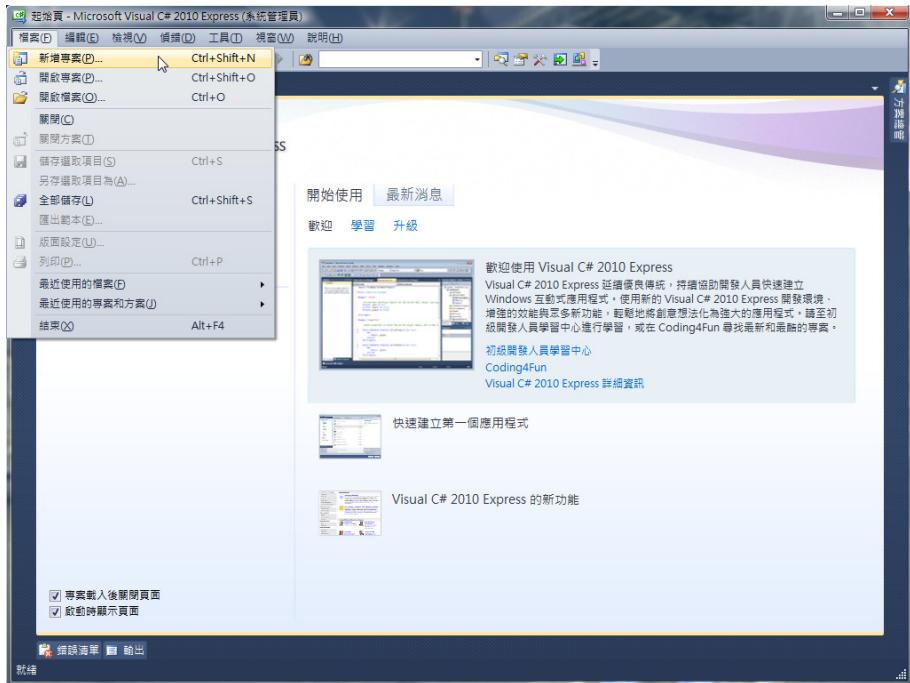
以下就是下載和安裝的進度畫面。

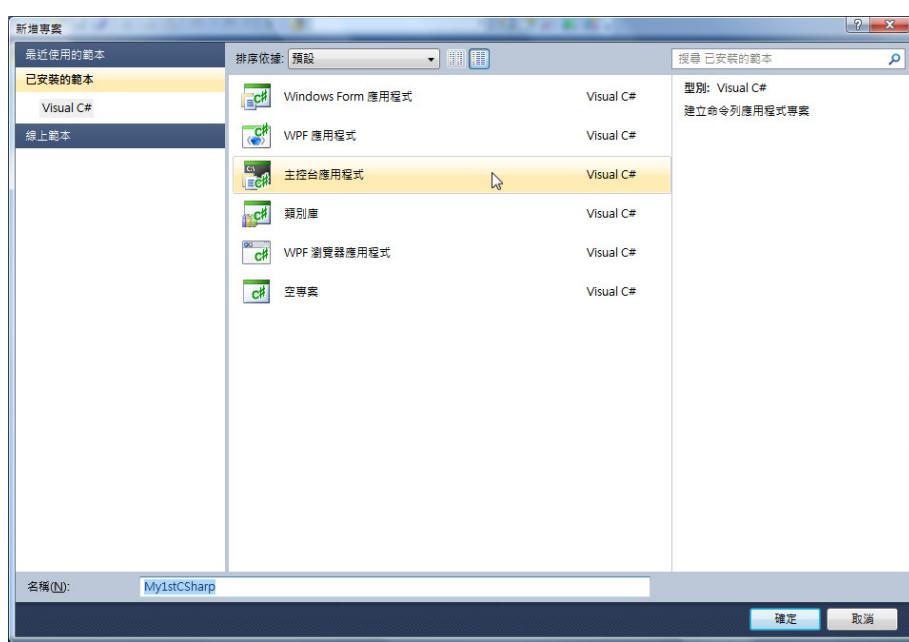


Step6：安裝程式完成後按下「結束」鍵，即可順利完成「Microsoft Visual C# 2010 Express」的安裝。



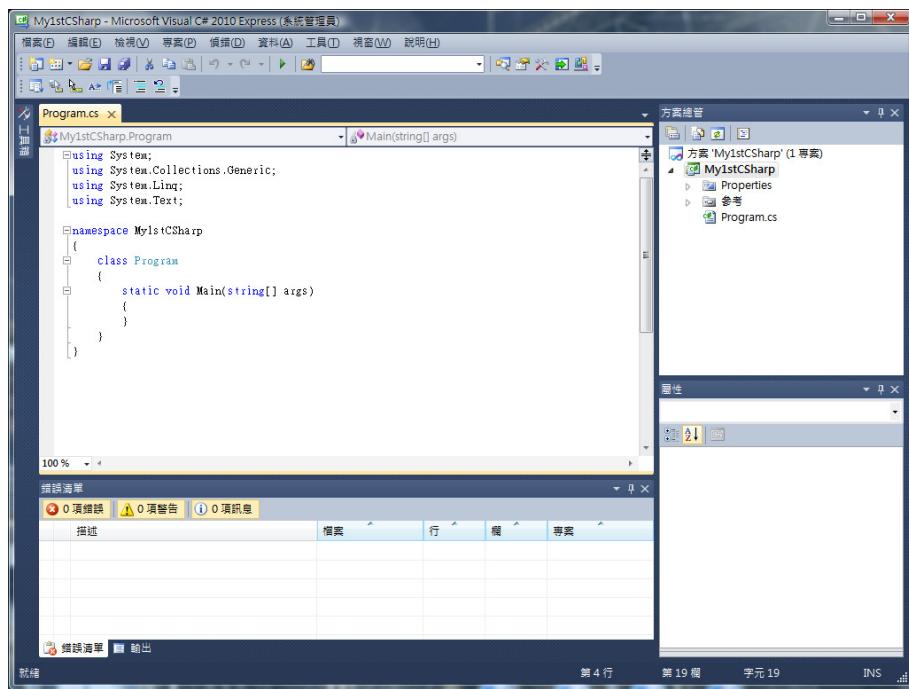
第二堂課：主控台應用程式開發

1.何謂主控台(Console)	5 分	<p>如果讀者用過 MS-DOS 或是 Linux 作業系統的話，想必對於命令行(Command Line)並不陌生，主控台應用程式便是設計在類似命令行底下作業的模式，而主控台(Console)則是 System 命名空間(NameSpace)底下的類別之一，用來處理主控台應用程式裡輸出或輸入等各種動作。</p> <p>相較於我們習慣的視窗圖型介面，主控台模式在顯示上是純粹的文字構成，或許讀者一開始會覺得不是很友善，但要記得「萬丈高樓平地起」，在我們把所學運用到視窗模式之前，觀念的部份最好還是先從基礎開始才比較清晰。</p>
2.主控台工作環境簡介	15 分	<p>從開始功能表找到 Microsoft Visual C# 2010 Express Edition 的捷徑，執行之後就會進入 Visual C# 的開發環境，首先我們從檔案功能表中選擇「新增專案」選項。</p>  <p>在跳出的「新增專案」視窗中，選擇「主控台應用程式」，接著取一個自己可以方便辨識的名稱，最後再按下「確定」鈕，即完成主控台應用程式的新增了。</p>

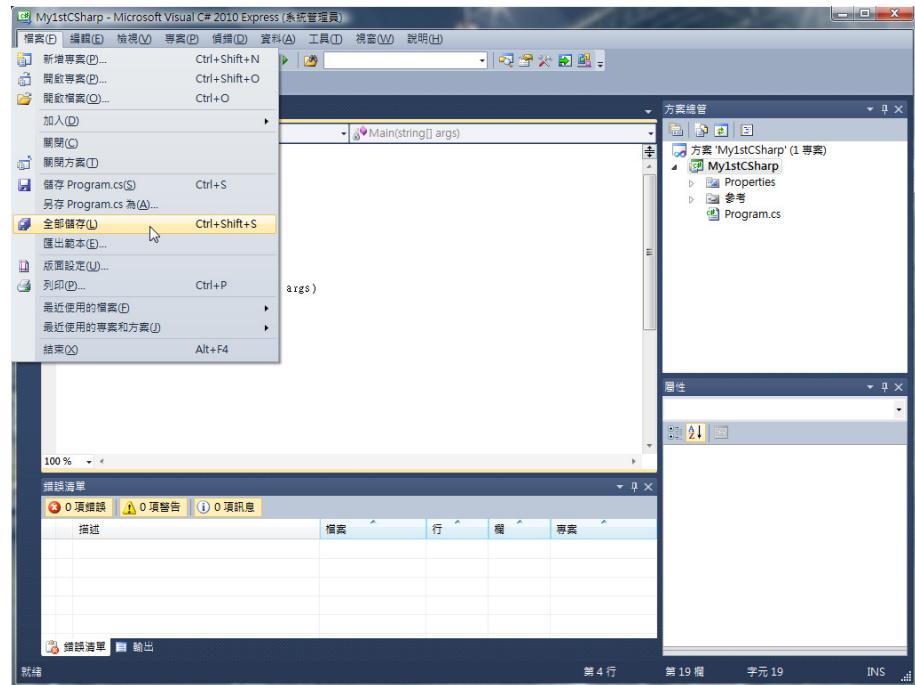


完成上述步驟後，畫面便會變成主控台應用程式的工作環境了。畫面中間有程式碼的區域便是我們要編輯程式碼的地方，除了「程式碼編輯區」以外，另外還有「方案總管」、「錯誤清單」、「工具箱」等各個區塊。每個區塊的右上角都有縮小、隱藏、關閉的三個按鈕。

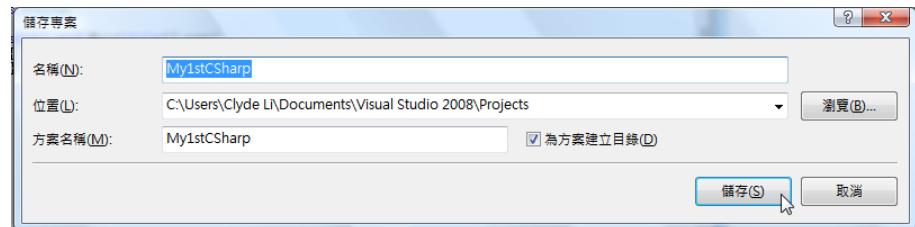
「方案總管」、「錯誤清單」、「工具箱」這三個區塊都有其個別的功能，「方案總管」內會顯示這個專案底下包含的程式碼及資訊，例如：讀者可以看到圖中編輯程式碼的區塊，顯示的便是方案總管下 Program.cs，這個 Visual C# 新專案預設產生的程式碼內容，而「工具箱」雖然在主控台應用程式下不常用到，但我們會在後續的單元中使用到，「錯誤清單」則是在 Visual C# 編譯程式碼出現錯誤時，提示我們那裡出了問題的地方。



在一切就緒之後，大家可能會迫不及待想要開始寫第一個程式，在寫程式之前，我們還是先練習將檔案存檔，養成先存檔的好習慣，以避免當電腦突然發生狀況時，所有心血付之一炬，我們從檔案功能表中選擇「全部儲存」選項。



專案的名稱由使用者自行命名，此處命名為「My1stCSharp」，存檔的位置也是自由選擇，然後務必勾選「為方案建立目錄」選項，Visual C#便會為該專案建立個別的資料夾，將所有的檔案置於資料夾中。



3. 第一個主控台程式的編譯與執行

30 分

Visual C#會自動幫我們新建立的主控台應用程式，設定好一些基本常用的東西，可以從方案總管中看到詳情，像是剛剛看到的 Program.cs 裡面的內容，便是一個它幫我們預先弄好基本架構的例子。不過在讀者試著去看懂這段程式碼之前，如果讀者您是個程式初學者，筆者想先從更簡單一點的例子開始講起，請看下面這段程式碼。

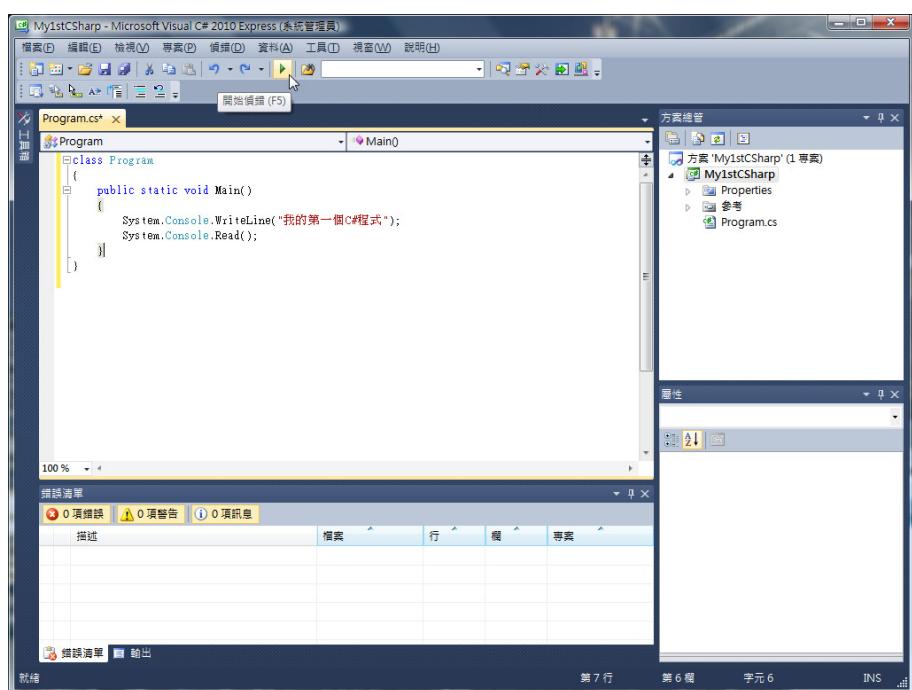
```
class Program
{
    public static void Main()
    {
        System.Console.WriteLine("我的第一個 C# 程式");
        System.Console.Read();
    }
}
```

跟預設的程式碼架構比起來，這段程式碼顯然更加簡單，筆者說明一下這段程式碼的意思。首先在 C# 語言中，**所有的程式碼都要寫在一個類別(Class)或是結構(Structure)內**，此例中我們的程式碼便是在 Program 這個類別當中，接下來 Main() 這個方法(Method)，則是所有 C# 程式的進入點，Main 前面的幾個單詞的意義以及何謂「方法」，我們也會在後面介紹到，注意到這裡在類別與方法中的內容，都要寫在 { } 大括號之間。最後兩行用分號(;)結尾的程式碼敘述，則分別是使用 System 這個命名空間內 Console 類別中的 WriteLine 方法印出一行文字，以及用 Read 方法讀取使用者輸入的資料，此處加入 Read 方法是用來讓畫面停住，否則程式執行畫面會一閃即逝。

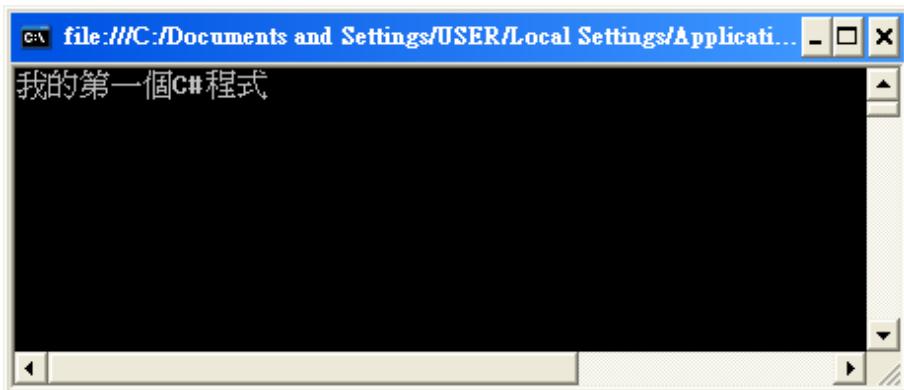
將以上程式碼鍵入程式碼編輯視窗(**注意！在 C# 語言中大小寫是有差別的**)，在輸入的過程中，讀者應該會發現 Visual C# 會在您輸入的同時列出建議的清單，這項方便的功能微軟稱之為 IntelliSense，相信讀者在之後的使用中，會發現它非常方便！



完成輸入之後，我們便可以開始編譯並執行這個程式了！只要按下工具列上三角形播放符號或是按鍵盤的「F5」鍵，Visual C# 便會開始編譯並執行程式！



如下圖所示，程式執行結果簡單的列出一行文字「我的第一個 C# 程式」，此時我們就完成了第一個 C# 程式！



接下來我們把程式碼稍微修改一下，來看看跟剛剛的程式有什麼不一樣。

```
using System;

namespace MyLib
{
    class Program
    {
        public static void Main()
        {
            Console.WriteLine("我的第一個 C# 程式");
            Console.Read();
        }
    }
}
```

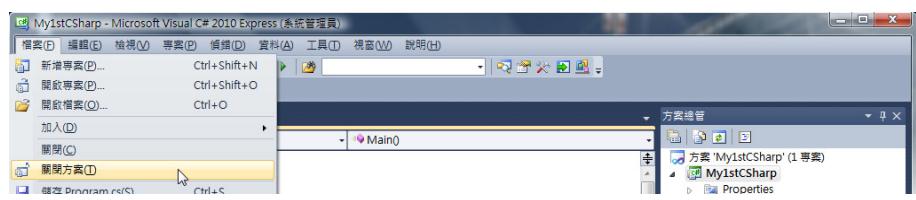
相信聰明的讀者應該會發現這個程式內容與剛剛的程式其實相差無幾，它的執行結果事實上也與剛剛的程式一模一樣，所以我們這邊要看的就是新增的幾個語法，一個是 `using`，另一個則是 `namespace`。

與上一個程式碼比較，讀者可以注意到一旦我們使用了 `using System` 之後，原本我們寫成 `System.Console` 的部份，現在就只要寫 `Console` 就可以了！**using System** 的功能便是告訴 Visual C# 「當你找不到東西的時候，去 `System` 這個命名空間裡面找就對了！」。

而 `namespace`(名稱空間)則是將我們寫的類別都包裝進去，就像是一個管理容器，可以將所定義的類別含括到不同的名稱空間之下，比較不會產生相互衝突的情況，例如：我們定義了一個名稱為 `3D` 和一個名稱為 `4D` 的名稱空間，在這兩個名稱空間之下都有一個 `Arrow` 類別，由於 `Arrow` 類別屬於不同的名稱空間，所以這兩個類別名稱，並不會有所衝突。

完成並存檔後，只要點選檔案功能表中的「關閉方案」選項，便能將這個方

案給關閉了！相同的，以後要再開啟專案，只要選擇檔案功能表中的「開啟專案」選項，然後找到當初存檔所選擇的相應專案檔(.sln/.csproj)即可。



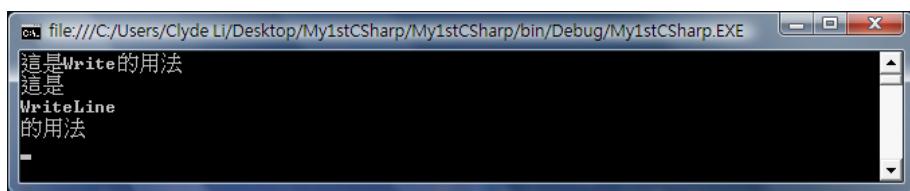
第三堂課：主控台輸出入

1. Write 與 Read 30 分 在前面的例子中，我們用到了 Console 類別當中的 WriteLine 和 Read 兩個方法，因為在主控台應用程式當中，使用者與程式互動的唯一方式就是透過文字的顯示與讀取，所以在主控台專案當中，與 Write 和 Read 相關方法的使用就顯得相當重要，前者用來將資訊輸出顯示在螢幕上，後者則是用來將使用者輸入的資訊接收到程式當中。我們舉一個簡單的例子，介紹 Write 與 WriteLine 這兩種方法的用法：

```
using System;

namespace MyLib
{
    class Program
    {
        public static void Main()
        {
            Console.Write("這是");
            Console.Write("Write");
            Console.Write("的用法\n");
            Console.WriteLine("這是");
            Console.WriteLine("WriteLine ");
            Console.WriteLine("的用法");
            Console.Read();
        }
    }
}
```

這段程式碼執行的結果如下：

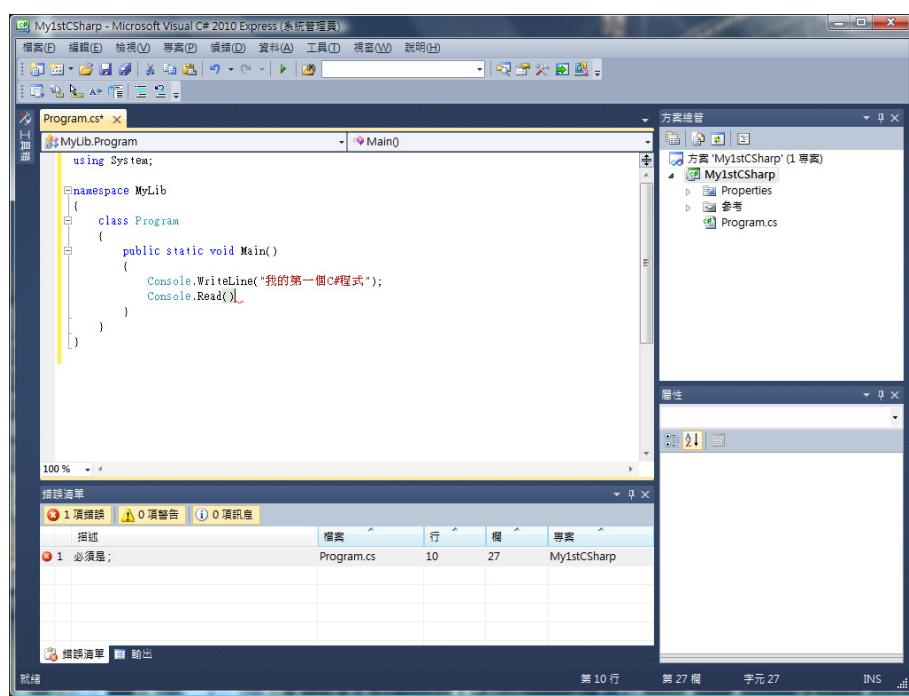


Write 方法與 WriteLine 方法的用法類似，我們只要把要印出來的字串擺在後面括號內並在前後加上雙引號，告訴程式這是一段字串即可，而讀者可以發現其實兩者的差別，就是差在有沒有換行而已。用 Write 方法的話，程式就不會進行換行的動作，而是會接著前一段字串印出來，用 WriteLine 方法的話，就會印出文字後換行。不過讀者或許也發現了，最後一個 Write 方法其實也有換行，原因就是筆者在字串中加上了「\n」，「\n」是一種特殊字元，而這個特殊字元的作用就相當於換行，以下列出我們常用的特殊字元給讀者參考：

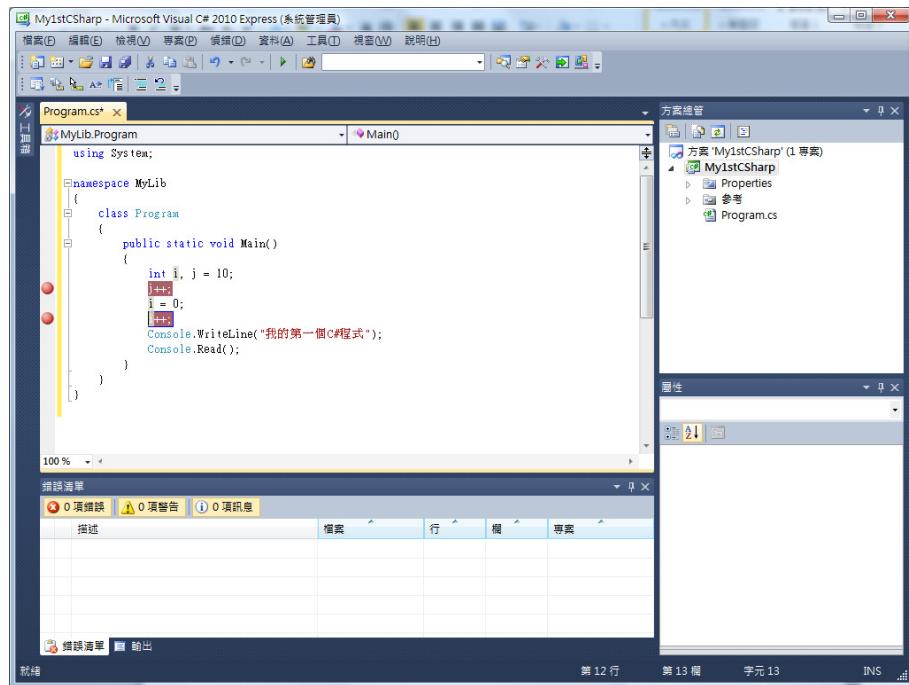
特殊 字元	意義	解說
\a	Alert(beep)	讓電腦發出警笛音(嗶聲)
\b	Backspace	往回退一格
\t	Tab	相當於鍵盤上 Tab 鍵的功能
\n	New Line	換行
\"	Double Quote	因為在程式碼當中「"」「'」「\」都有被賦予其他程式上的意義，所以當我們想要"真的"印出這幾個字元時，必須先加上「\」。
\'	Quote	
\\\	Backslash	
\x0ABC	Unicode Character	印出編碼為 0ABC 的 Unicode 字元

相對的，Read 和 ReadLine 方法所做的事就和 Write 相反，他們的工作是把使用者輸入的資訊讀到程式當中，詳細的用法我們留待後面章節再說明。我們現在用 Read 方法的目的，就只是單純為了讓程式等待使用者輸入資料，連帶的把程式畫面停住，讓我們觀察執行結果而已。

2.Visual C#的偵錯 技巧	20 分	<p>記得筆者學程式設計時，教授就常常說「沒有程式是沒有錯誤的」。的確，隨著我們寫的程式越來越複雜，錯誤總是難以避免，有可能是語法上有不符合規定的地方，也有可能是程式跑出來的結果不符合自己的預期，這個時候我們就需要 Visual C#除錯功能的幫助了！</p> <p>如果是語法的錯誤，Visual C#在我們輸入時就會在錯誤清單區塊中即時提醒我們，例如下圖中，筆者在一行敘述的結尾忘了加上分號(;)，在我們還沒編譯程式之前，Visual C#便會在錯誤清單中立刻提醒我們在某一行，發生了什麼樣的錯誤！</p>
----------------------------------	---------	---

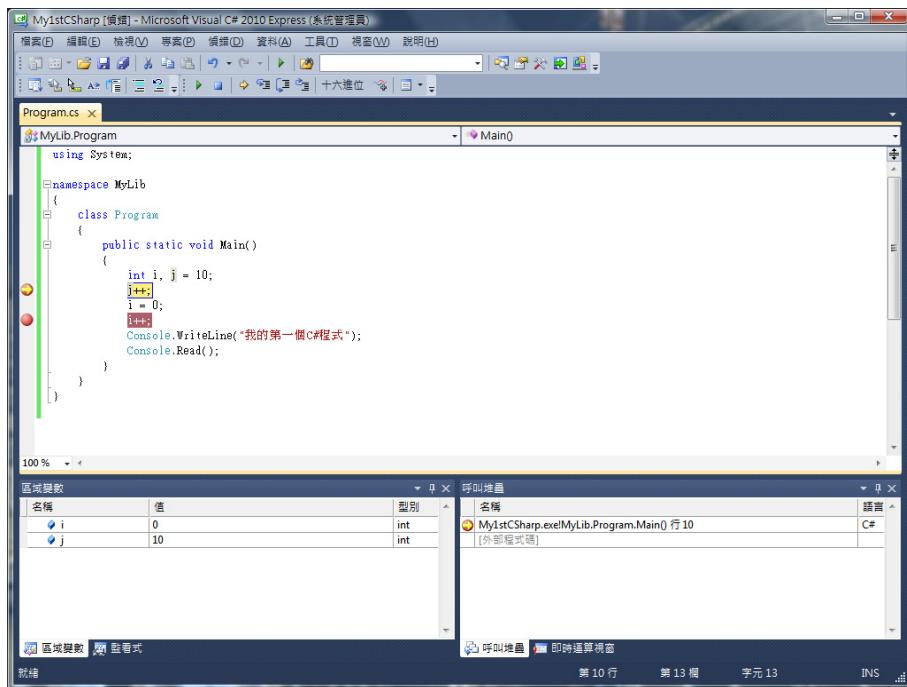


如果不是語法的錯誤，我們通常就會想要瞭解是不是程式在執行的過程中，有哪些部分是不如預期的，這個時候我們就會設程式中斷點，也就是要暫停程式執行的位置。設定中斷點的方法很簡單，我們只要用滑鼠在想要暫停的程式碼位置，在該行前面灰色的區域點擊滑鼠左鍵，畫面就會出現一個紅色的點，提示該處為程式執行時要暫停的地方，如下圖所示。

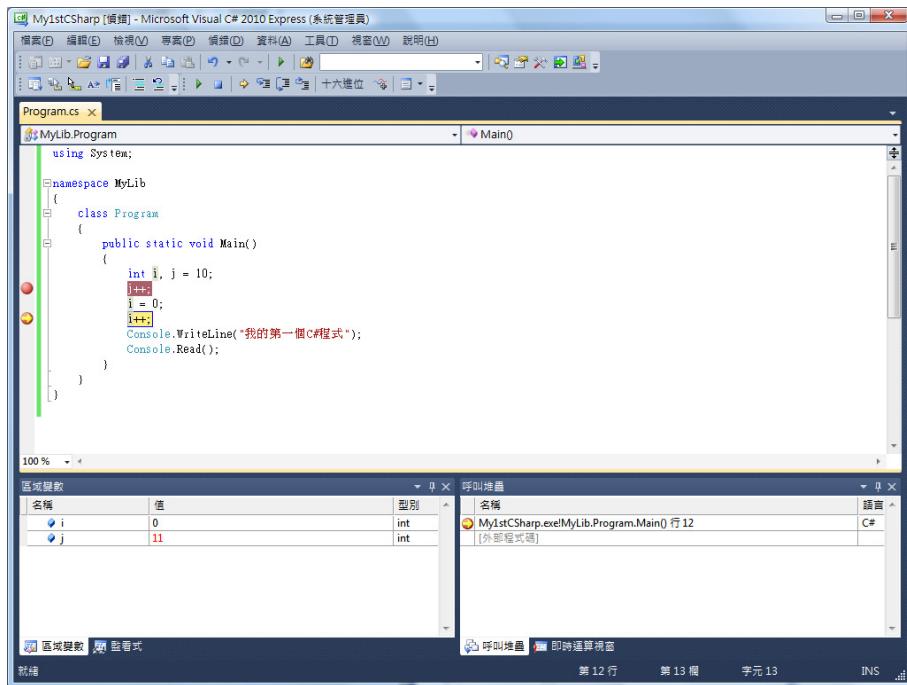


我們用來示範的程式是一個有兩個變數的程式，還不知道變數概念的讀者也沒關係，我們在後面章節會加以說明。設了中斷點後我們開始執行程式，當程式執行

到中斷點時會停住，並在左下角的區塊中顯示現在有那些變數以及他們的數值等資料；讀者也可以把滑鼠游標移到程式碼中的變數上，一樣會有相關資料的顯示。

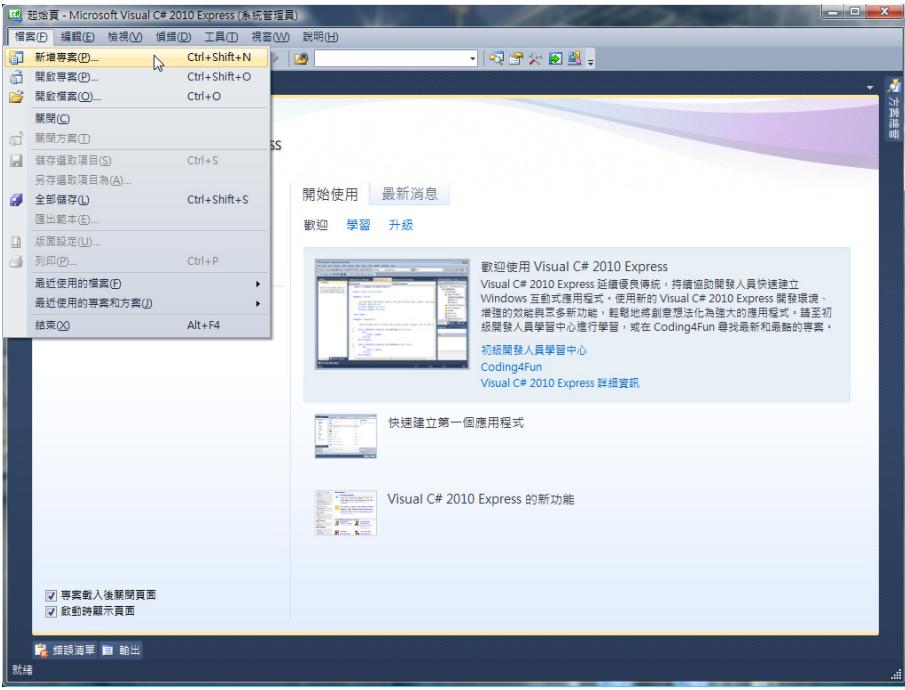
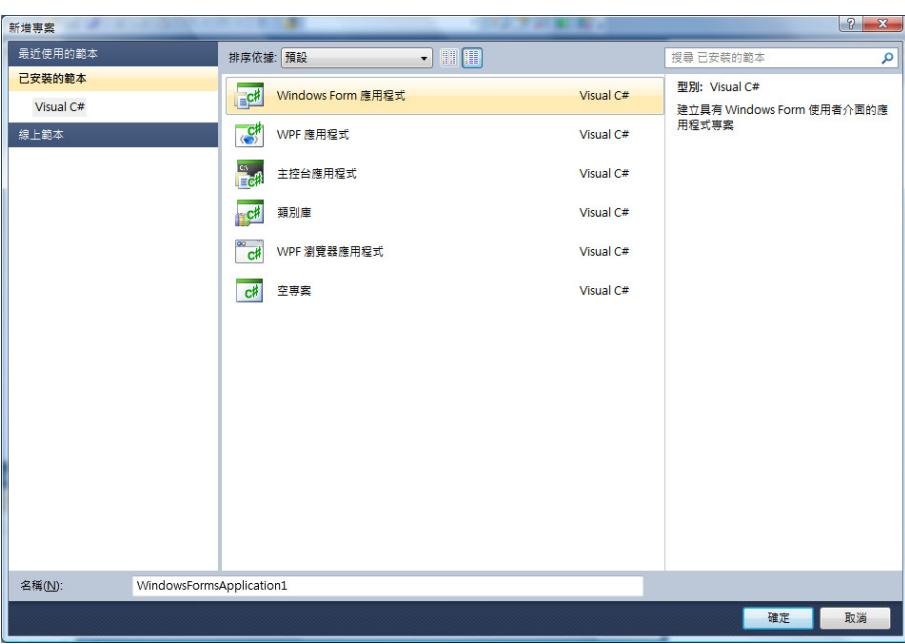


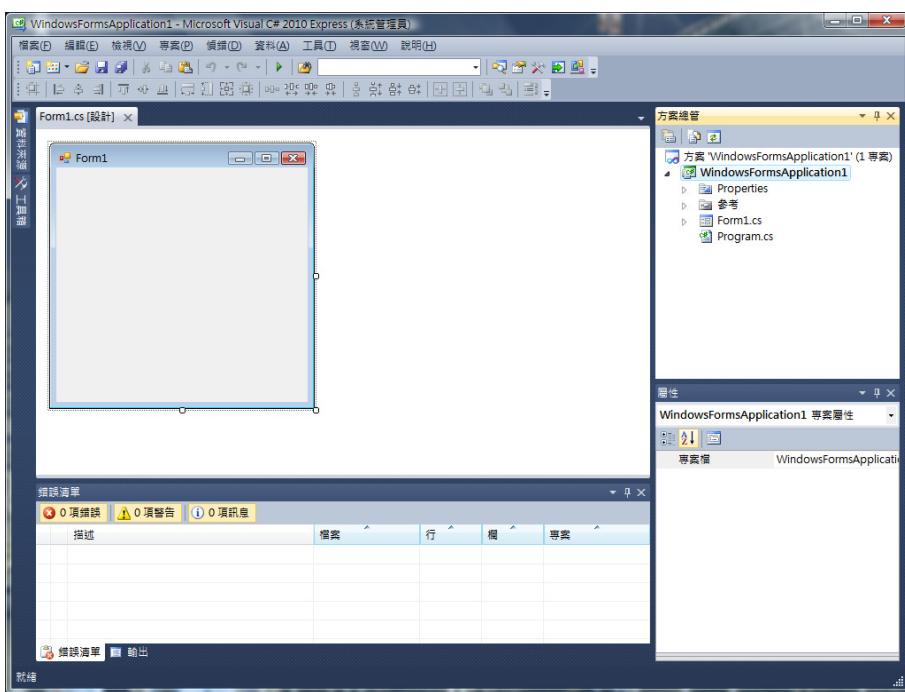
這樣一來，程式在這個中斷點時程式內各個變數的狀態便非常清楚了，有利於我們掌握程式的運作狀況，接著我們可以點選左上角工具列上，一個類似播放符號的按鈕或是按鍵盤的「F5」鍵，繼續前往下一個中斷點進行偵錯，此時我們可以發現j值出現了變化，如下圖。



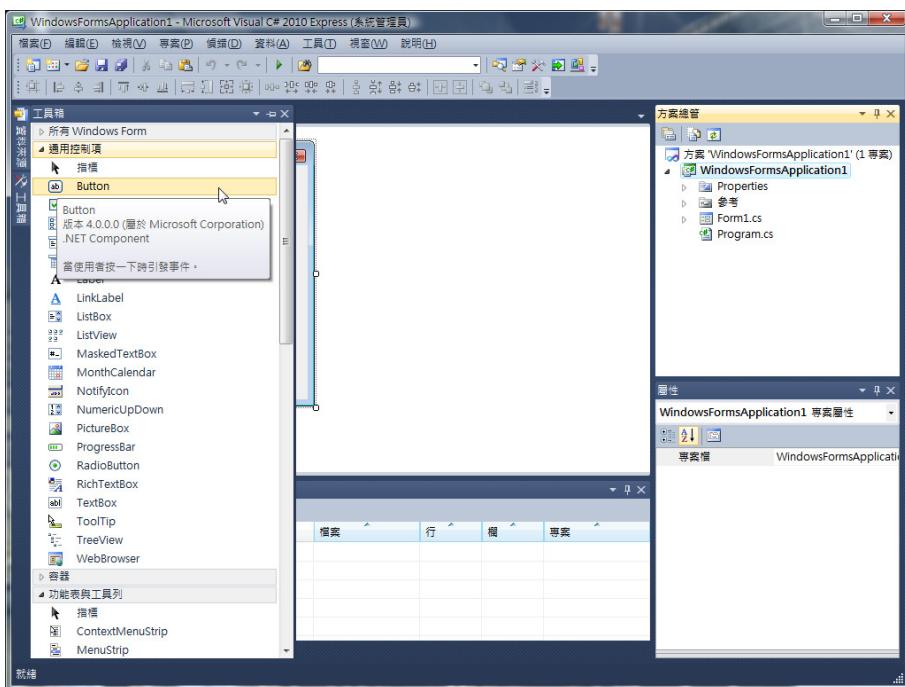
Visual C#提供的這項設定中斷點來偵錯的功能，在我們以後要找程式錯誤時，將會非常好用！

第四堂課：Visual C#視窗應用程式開發

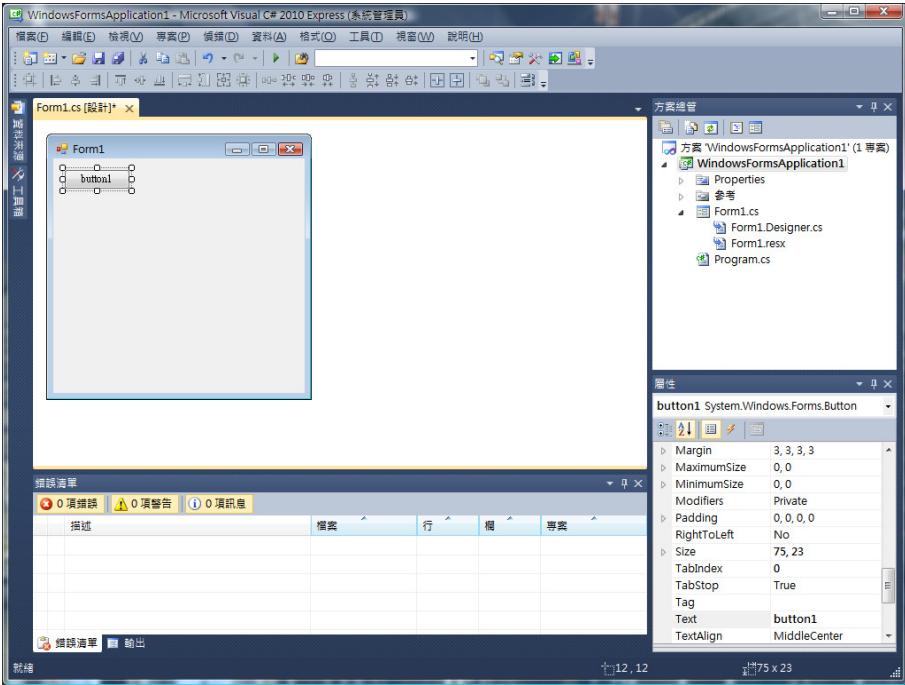
1.認識視窗 程式工作 環境	10 分	<p>開啟新專案的方式與之前一樣，從開始功能表找到 Microsoft Visual C# 2010 Express Edition 項目，選取執行進入 Visual C# 的開發環境，接下來在功能表中選擇「檔案/新增專案」選項即可。</p>  <p>In the screenshot, the 'File' menu is open, and the 'New Project...' option is highlighted with a yellow selection bar. The main window shows a welcome message for Visual C# 2010 Express.</p> <p>在「新增專案」視窗中，選擇「Windows Form 應用程式」，接著命名一個自訂的名稱，最後再按下「確定」鈕即完成視窗應用程式的新增了。</p>  <p>The 'Add New Project' dialog box is shown. The 'Windows Form Application' option under the 'Visual C#' category is selected. The 'Name' field contains 'WindowsFormsApplication1'. The 'OK' button is visible at the bottom right.</p> <p>完成以上操作動作之後，就可以看到視窗應用程式的工作環境了，區塊的配置與主控台應用程式的工作環境大同小異，比較特別的是原本主控台模式下，撰寫程式碼的區域在視窗應用程式模式下變成了「Windows 表單設計區」。</p>
----------------------	------	--

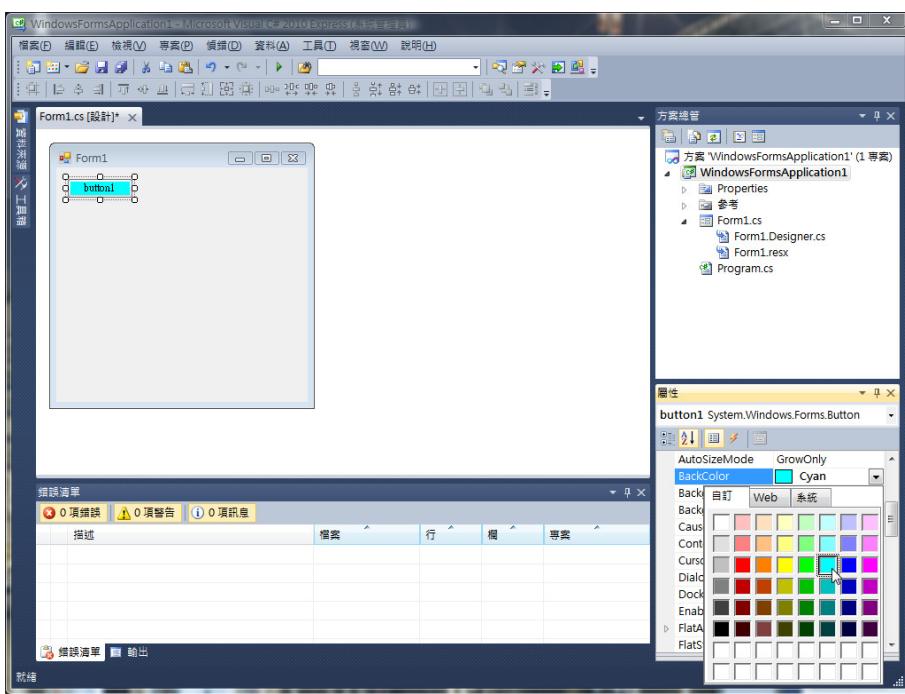


接下來我們可以從左側的「工具箱」中，隨意點選一個工具試試看，筆者以按鈕(Button)元件為例，讀者可以試著用點選或是拖曳的方式，在剛剛的 Windows 表單設計區上面新增一個按鈕。



讀者可能也注意到了，隨著我們所選物件的不同，從一開始的 Windows 表單到我們剛剛新增的按鈕，畫面右下角的屬性視窗中的內容，也會跟著變動，其實在視窗設計模式下，每個物件都會有他自己的屬性，我們可以在屬性視窗中進行各種設定。

		
2. 視窗程式 的四個構 成	20 分	<p>一、物件(由程式碼構成)</p> <p>以日常生活的概念來說，任何實體的物品皆可視為物件，例如：電腦主機、飛機、汽車...等等。而在視窗的設計模式下也有類似概念，例如：視窗程式執行時，我們所看到的表單 Form、按鈕 Button、文字方塊 TextBox...等等，都可以視為物件。</p> <p>每一個物件將其內部資訊給封裝起來，只提供了「屬性」和「方法」來讓我們操作，因此當我們設計視窗程式時，並不需要知道物件內部是如何運作的，我們只需瞭解物件提供的介面與特性，就可以很方便的存取它的內部功能。我們接下來會在視窗模式下，使用各式各樣的物件，像是先前提到的表單 Form 和按鈕 Button，都是微軟所提供的物件。</p> <p>二、屬性(描述物件的特性)</p> <p>有了物件的概念後，我們就可以來看看如何改變物件的特性了。當我們想改變物件的外觀時，我們可以經由屬性視窗來做設定。以日常生活中的物品為例子來舉例，例如：「一個黑色的盒子」是指「一個盒子的顏色是黑色的」，其中我們可以把盒子(box)當作是物件的意義，而顏色(color)就是盒子的屬性的意義了。</p> <p>Visual C#提供的各種物件，都有許多屬性可以設定，多加摸索各個元件的屬性，可以讓我們開發出來的程式更豐富。舉例來說，我們可以從前面看到的「屬性」區塊中，設定一個物件的某個屬性，例如：筆者在這裡改變了按鈕「button1」的屬性「BackColor」(底色)為「Cyan」。</p>

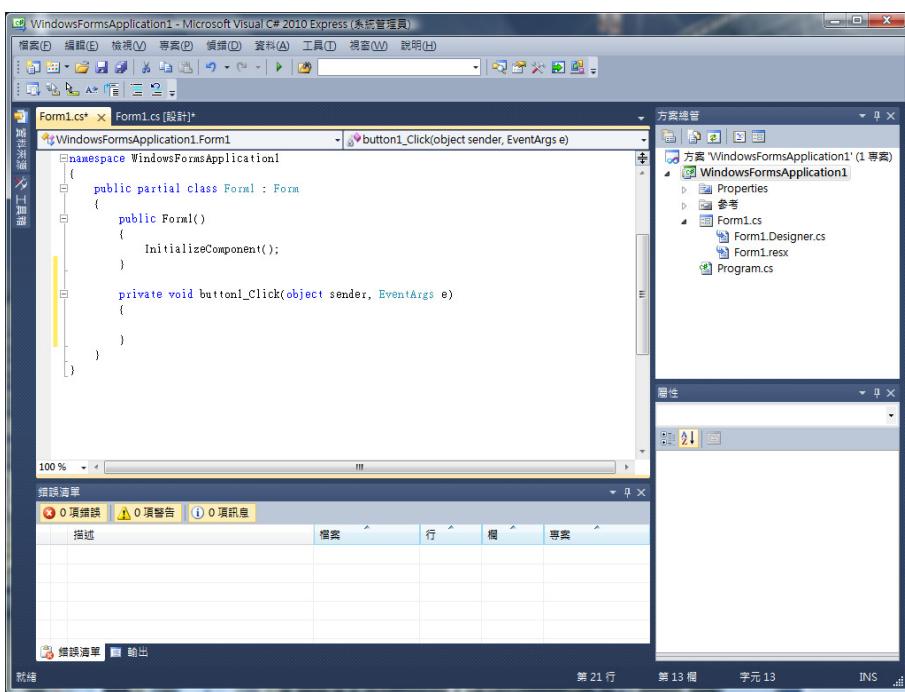


除了在屬性視窗裡面設定以外，我們也可以在程式碼中，動態的改變某個物件的某項屬性，例如：將此例寫成程式碼即為「`button1.BackColor = Color.Cyan;`」。

三、事件(對於物件所做的動作)

人與程式能輕鬆的進行互動是視窗程式的一大賣點，而程式要如何知道使用者的操作動作，就要透過特定事件的觸發。視窗應用程式以事件驅動程式動作，像是按滑鼠左鍵一下、鍵盤輸入、Windows Form 載入...等等，通通都是事件的動作，使用者透過觸發事件就能輕鬆的與程式互動，至於一個事件觸發後要引發什麼動作，就是我們程式碼所要設計的部分了。

在 Visual C# 中，我們可以用兩種方式來開始設定事件的觸發，第一種方法就是用滑鼠左鍵雙擊畫面中的物件，讀者可以試試看用滑鼠左鍵雙擊表單設計畫面中的按鈕，表單設計畫面就會切換到程式碼設計的頁面。



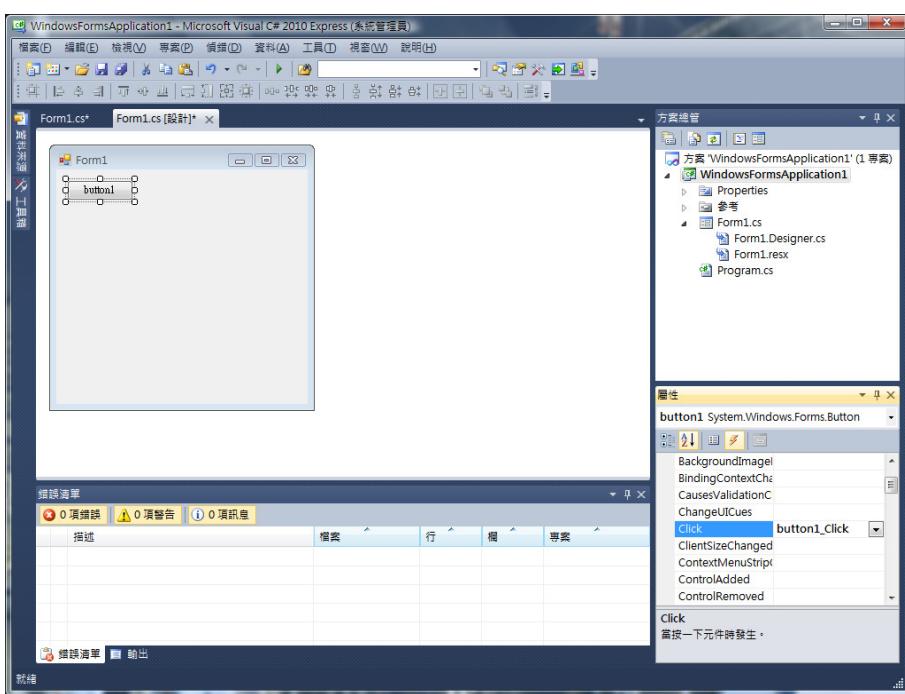
這邊的程式碼是 Visual C# 帶我們處理好的基本架構，而其中跟我們剛剛滑鼠雙擊按鈕有關的程式碼是下面這個部份：

```
private void button1_Click(object sender, EventArgs e)
{
}
```

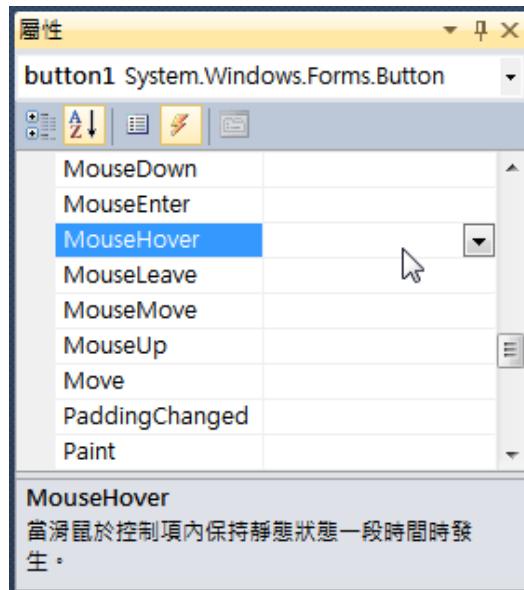
此處我們可以發現其中的關鍵字「button1_Click」，C# 會自動以事件動作取名，方便我們辨識這是一段要處理「點擊按鈕」這個事件的程式。

所以在表單設計畫面用滑鼠左鍵雙擊物件，會讓 Visual C# 以該物件最常被觸發的事件，來幫我們增加一段「方法」的程式碼框架，如在此例中，當我們雙擊按鈕，Visual C# 就會幫我們新增一段與「點擊按鈕」事件相關的程式碼。

接著就來介紹設定其他事件觸發的方式，我們先切換回到表單設計頁面，點選一個要設定事件觸發的物件，在此例中一樣以表單上的按鈕為例，點選之後，讀者可以在**右下角的屬性視窗當中的工具列**，看到一個「閃電」圖案的按鈕，點選該按鈕之後，Visual C# 便會列出該物件所有可觸發事件的列表，讀者可以在圖中看到我們剛剛用雙擊 Button1，所設定的「Click」事件已經在此列表中被關聯到「button1_click」這個「方法」了。



有些事件或許看名稱，不一定清楚該事件的觸發條件為何，這時讀者便可以參考屬性視窗下方說明方塊中的說明。



事實上讀者可以發現，在 Visual C# 中光是按鈕這個元件所能相關的事件觸發動作，就已經是五花八門一大堆了，例如讀者將事件列表往下拉，可以看到一個名為「MouseHover」的事件，也就是我們僅僅需要將滑鼠游標移到按鈕上方，就可以觸發該事件了。讀者選定觸發事件後，只要在屬性視窗的事件右側空白地方雙擊滑鼠左鍵，Visual C# 便會幫我們增加一段相應的程式碼框架到程式碼中了。

```

using System.Windows.Forms;
namespace WindowsFormsApplication1
{
    public partial class Form1 : Form
    {
        public Form1()
        {
            InitializeComponent();
        }

        private void button1_Click(object sender, EventArgs e)
        {

        }

        private void button1_MouseHover(object sender, EventArgs e)
        {
        }
    }
}

```

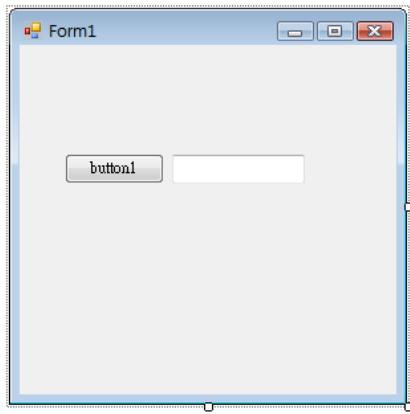
四、方法(物件本身具備的功能)

除了自己定義的「方法」以外，基本上在 Visual C# 中，每個物件本身也都會有許多 Visual C# 先幫你定義好的方法，方便我們去執行與該物件相關的動作。

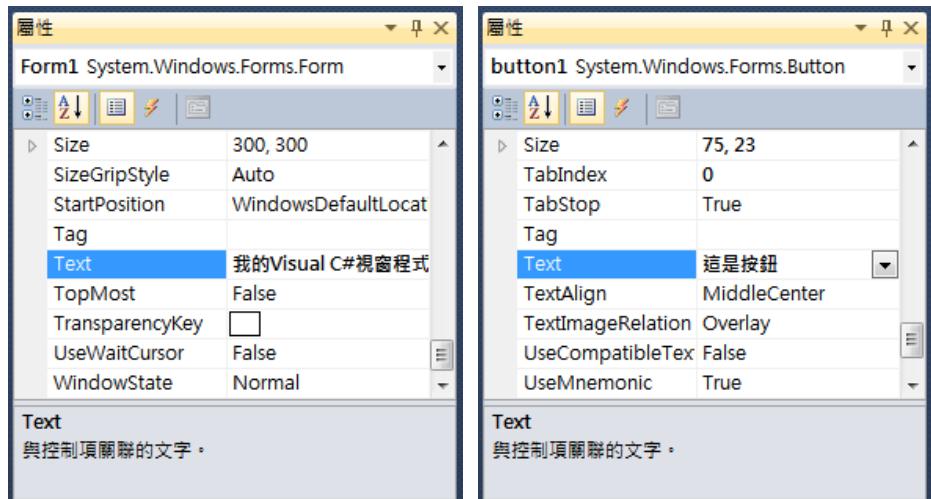
以日常生活中的例子來說，例如：「把電風扇打開」，在 Visual C# 程式裡可以用類似這樣子的概念寫法：「Fan.Turnon()」。Turnon 這個動作是 Fan 所具有的功能，因此用「物件.方法」的格式，來讓 Fan 去執行 Turnon 這個動作。

在 Turnon 後面的小括弧「()」，則可以放入一些參數，讓物件可以依據參數的內容，來決定該如何去執行要去做動作。例如：「電風扇的轉速要多快？」。為了做這類更細節的要求，我們可以傳入相關參數，像是我們想讓電風扇轉快一點，程式的敘述就可以寫成「Fan.Turnon(fast)」，讓電風扇知道它應該運作得多快。這些參數的傳入格式，包括：參數的個數、類型... 等等，都可以由程式撰寫者來做定義，我們將會在後面專門介紹「方法」的章節中再做進一步說明。在此舉一個在 Visual C# 中，把按鈕隱藏起來的方法，其程式碼為「Button1.Hide();」，直接呼叫物件本身具備的功能，相當簡單。

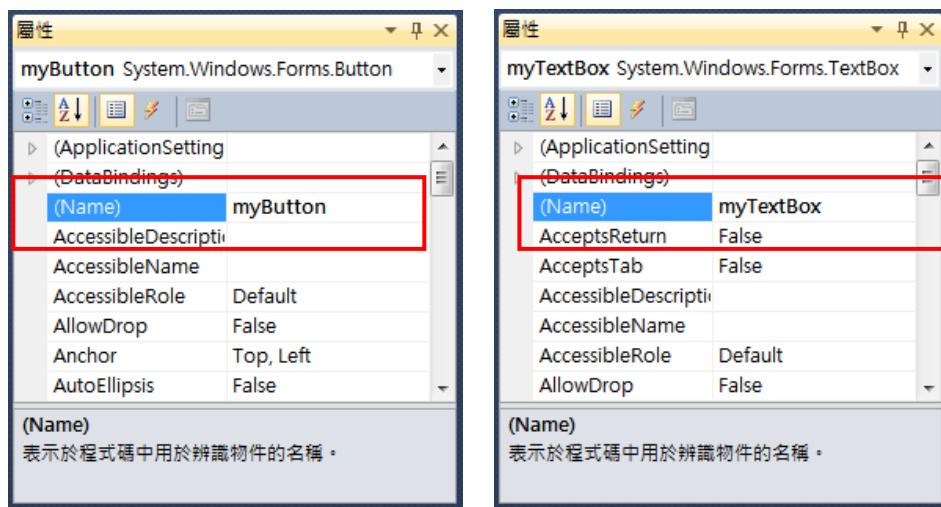
3.建立第一個視窗程式	20 分	以剛剛所學到的概念出發，我們可以開始建立第一個 Visual C# 視窗程式了！請各位讀者先用工具箱中的 Button(按鈕)和 TextBox(文字方塊)，在表單上面佈置一個按鈕和文字方塊，讀者可以自行配置按鈕與文字方塊的大小和位置。
--------------------	------	--



接下來我們要利用屬性視窗修改這個表單的標題以及按鈕和文字方塊的名稱，讀者只要用滑鼠左鍵點選表單上空白的地方，就可以選取到表單，接著在屬性視窗當中找到 Text 這項屬性並將其改成「我的 Visual C# 視窗程式」，接著點選按鈕(button1)，如法炮製將屬性視窗中的 Text 屬性改成「這是按鈕」。



接下來筆者希望讀者用一樣的方法，更改按鈕和文字方塊的 Name 屬性。為什麼要改變(Name)(名稱)這個屬性呢？因為 Visual C# 取名的方式太簡單，例如按鈕它就取 button1, button2, button3...，依序往下取。此種取名方式在小程式當中還可以接受，但隨著程式的規模越來越大，這樣的取名方式就很容易讓我們弄不清楚到底哪個按鈕是什麼名稱，故建議各位讀者為每個物件依該物件特性，取一個能夠讓自己很清楚記得的名稱，例如：在此例中筆者就分別取名按鈕和文字方塊為「myButton」和「myTextBox」。



好！到目前為止，我們的第一個程式現在應該是變成了這個樣子吧，請注意標題列和按鈕文字的改變。



接著要開始為這個程式寫一些簡單的程式碼了，在寫程式碼之前，我們要設定要被觸發的事件，讀者先依照前一節設定觸發事件的第一種方式，雙擊 myButton 這個按鈕後，Visual C# 會幫我們建立好「點擊 myButton」這個事件觸發的方法基本框架，接著在該方法中填入程式碼，如下圖：

```

namespace WindowsFormsApplication1
{
    public partial class Form1 : Form
    {
        public Form1()
        {
            InitializeComponent();
        }

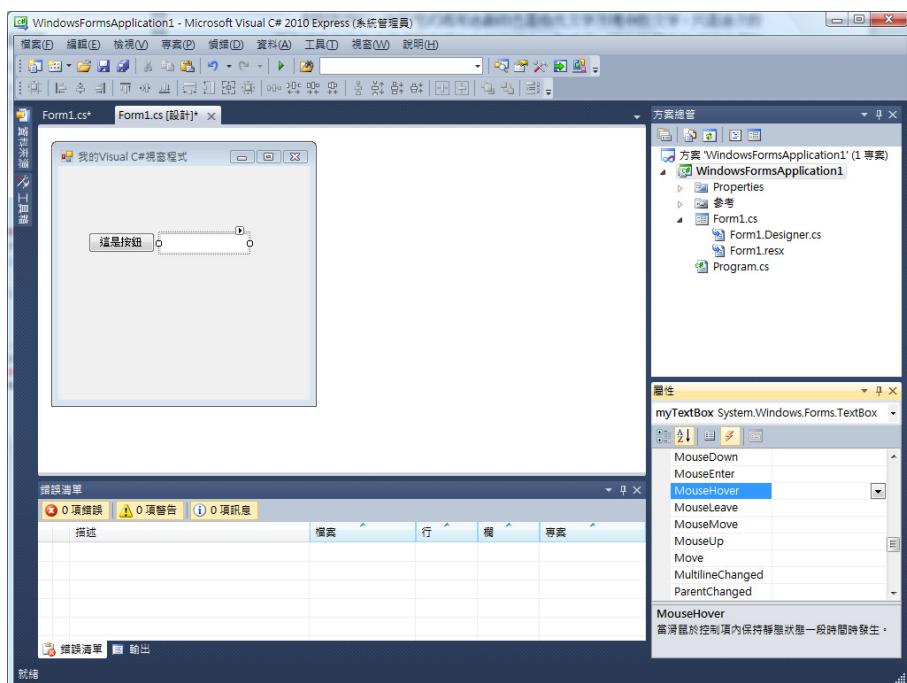
        private void myButton_Click(object sender, EventArgs e)
        {
            myTextBox.Text = "您點擊了「這個按鈕」";
        }
    }
}

```

程式碼內容如下：

```
private void myButton_Click(object sender, EventArgs e)
{
    myTextBox.Text = "您點擊了「這個按鈕」";
}
```

這段程式碼的作用就是讓「myButton_Click」這個事件被觸發後，文字方塊中的文字會被改成「您點擊了「這個按鈕」」，接著我們選取 myTextBox 文字方塊，並在右下角的屬性視窗中列出該文字方塊的「事件列表」，找到 MouseHover 事件並用滑鼠左鍵雙擊旁邊的空白，Visual C#便會幫我們建好該事件觸發的方法基本框架。



接著我們在程式碼被新增的方法中填入程式碼如下圖：

```
namespace WindowsFormsApplication1
{
    public partial class Form1 : Form
    {
        public Form1()
        {
            InitializeComponent();
        }

        private void myButton_Click(object sender, EventArgs e)
        {
            myTextBox.Text = "您點擊了「這個按鈕」";
        }

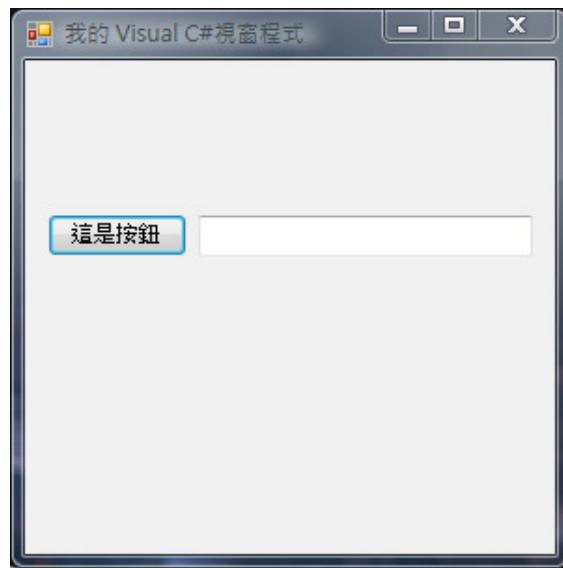
        private void myTextBox_MouseHover(object sender, EventArgs e)
        {
            myTextBox.Text = "您的滑鼠游標現在在文字方塊上面";
        }
    }
}
```

程式碼內容如下：

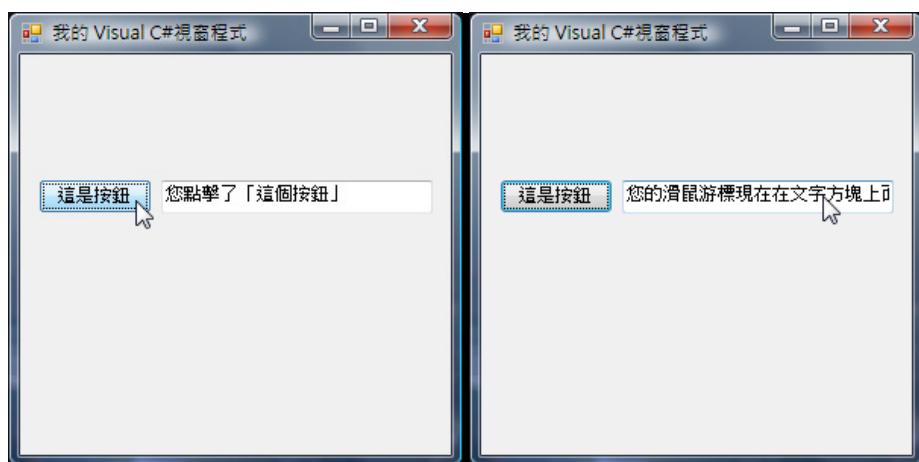
```
private void myTextBox_MouseHover(object sender, EventArgs e)
{
    myTextBox.Text = "您的滑鼠游標現在在文字方塊上面";
```

}

與剛剛類似，這段程式碼的最終用途也是修改文字方塊中的文字，只是這次的事件觸發條件為「當滑鼠游標在文字方塊上」的時候，至此我們就完成了我們的第一個 Visual C# 視窗程式了！



與主控台應用程式一樣，完成了程式的撰寫之後我們便可以開始編譯並執行這個視窗程式了！只要按下工具列上三角形類似播放的符號或是按鍵盤的「F5」鍵，Visual C#便會開始編譯並執行程式！我們的第一個 Visual C# 視窗程式的執行結果如下。點擊按鈕後，文字方塊便會顯示「您點擊了「這個按鈕」」，如果將滑鼠游標移到文字方塊上方，文字方塊則會顯示「您的滑鼠游標現在在文字方塊上面」。

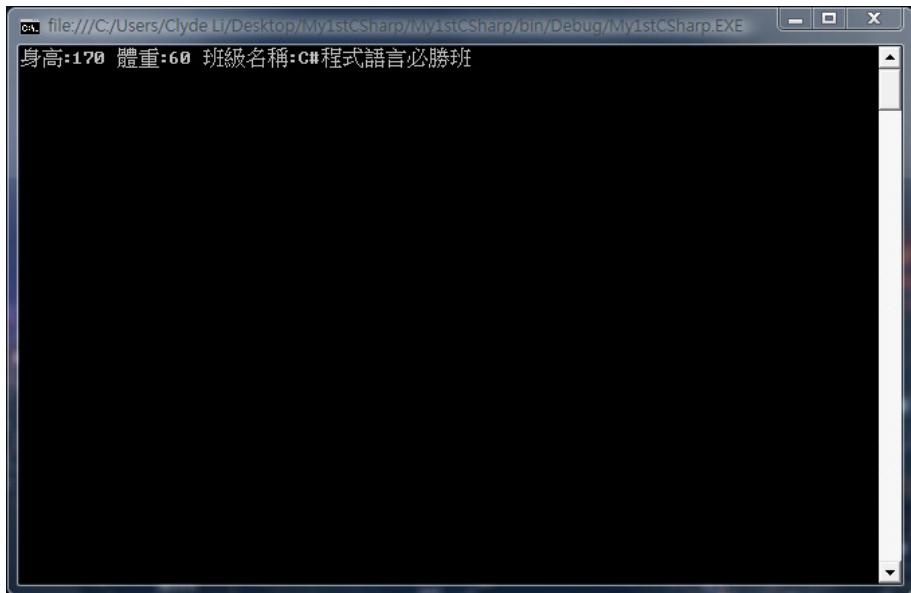


第五堂課：變數與資料型態

1. 變數的宣告與使用	15 分	<p>變數（Variable）是程式語言最基本的元素之一，它可以存放各種不同型態的資料，我們可以依據程式設計的需求，宣告各種類型的變數，以利程式的開發。變數相當於數學中的未知數（x,y,z...），如數學方程式 $2x+3y=4$，可以將 x 和 y 視為數字變數。在程式語言中的變數，除了表示數字外，還有字串、日期與布林值（True/False）…等各種資料型態。變數在電腦領域是一個佔有記憶體空間的資料存放區。</p> <p>程式的運作，基本上是由一連串的「取出資料」、「運算」、「儲存資料」…等動作所組成。當程式執行需要在記憶體中取得資料時，程式需先知道資料所在的記憶體位址，才能正確抓取到所要的資料；程式語言為了讓使用者容易撰寫程式，特別將資料的記憶體位址以變數的觀念取代，程式中若要存取該資料時，則以變數名稱來指定即可存取。因此我們可以將變數看作是記憶體中存放未知數值的空間。</p> <p>而在程式的設計過程中，變數的宣告是相當重要的一環。所謂變數宣告，就是將程式所使用的變數之相關資訊告知電腦，之後我們便可以透過變數名稱來存取變數所記錄的值。在 C# 語言中，變數的宣告格式如下：</p> <p style="background-color: #e0e0e0; padding: 5px;">type var-name;</p> <p>其中 type 為變數的資料型態，如：<code>int</code>、<code>float</code>、<code>char</code>…等等，我們在下一節當中會詳細介紹，而 var-name 則是指變數的名稱，除了 C# 語言中有一些保留字不可作為變數的名稱之外，變數的名稱基本上是可以讓使用者隨意設定的，不過筆者建議讀者在宣告變數名稱時，要盡量選用簡潔好記且不會混淆的名稱。另外要注意宣告敘述結束後，有一個分號（；）是一定要加上的，這個分號是用來告知編譯器此行程式到此結束，很多 C# 程式語言的初學者，都會忘記在程式敘述的結尾加上分號，各位讀者可別忘記了喔。</p> <p>我們以主控台模式實作一個宣告變數的範例如下：</p> <pre>using System; class Program { public static void Main() { int height, weight; //宣告整數變數 height 和 weight string class_name; //宣告字串變數 class_name height = 170; weight = 60; class_name = "C#程式語言必勝班"; Console.WriteLine("身高：" + height + " 體重：" + weight + " 班級名稱" + class_name); } }</pre>
-------------	------	---

```
        Console.Read();
    }
}
```

這段程式碼中我們用到了兩種資料型態，整數 int 以及字串 string，並用這兩種型態宣告了三個變數，分別是整數型態的 height 和 weight 以及字串型態的 class_name，最後在完成宣告之後，分別給予他們整數數值 170 和 60 以及字串「C# 程式語言必勝班」。程式執行的結果如下：



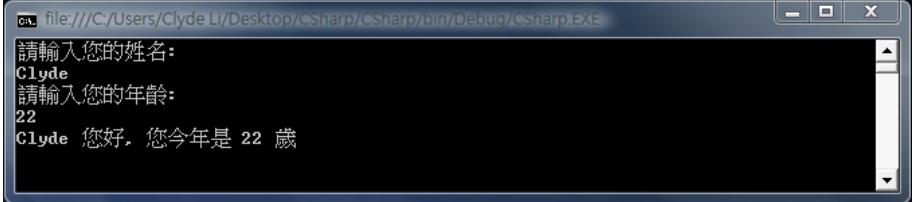
我們也可以在宣告變數的同時就給予變數起始值，以上面的程式為例，我們可以更改程式碼如下：

```
using System;

class Program
{
    public static void Main()
    {
        //宣告整數變數 height 和 weight 並給予起始值
        int height = 170, weight = 60;
        //宣告字串變數 class_name 並給予起始值
        string class_name = "C#程式語言必勝班";
        Console.Write("身高:" + height + " 體重:" + weight + "班級名稱:" +
                     class_name);
        Console.Read();
    }
}
```

這段程式碼的編譯執行結果會與前面的範例相同，唯一的不同就是我們在宣

		告變數的同時，就設定了起始值！
2.程式練習：讀取使用者輸入的姓名年齡程式	20分	<p>一、程式設計目標</p> <p>撰寫一個簡單的程式，程式當中會宣告兩個字串型態的變數，分別是 age 以及 name，運用 WriteLine 方法提示使用者輸入姓名與年齡，再利用 ReadLine 方法將使用者輸入的值，讀入這兩個變數當中，最後將獲取的資訊用 Write 或 WriteLine 方法印在螢幕上。</p> <p>二、程式碼撰寫</p> <pre> 01 using System; 02 03 class Program 04 { 05 public static void Main() 06 { 07 string age, name; 08 09 Console.WriteLine("請輸入您的姓名:"); 10 name = Console.ReadLine(); 11 Console.WriteLine("請輸入您的年齡:"); 12 age = Console.ReadLine(); 13 14 Console.Write(name + " 您好，您今年是 " + age + " 歲"); 15 Console.Read(); 16 } 17 }</pre> <p>三、程式碼解說</p> <p>程式一開始就是要先宣告變數囉，依照我們的程式目標，我們使用的資料型態是字串 string，兩個變數名稱則分別是 age 和 name，宣告 string 變數的程式碼如下。</p> <div style="background-color: #e0e0e0; padding: 5px;"> <pre>string age, name;</pre> </div> <p>接著我們用 WriteLine 方法印出提示，請使用者輸入姓名，這邊的重點是 ReadLine 的用法，Console.ReadLine() 會讓程式停下來等待使用者輸入字串，而它本身也就代表這個字串(精確一點來說，也就是 ReadLine 這個方法的回傳值就是使用者輸入的字串)，所以我們在前面加上 「name = 」 就可以把讀到的字串存入變數 name 當中，以同樣的方式，我們接著把使用者的年齡存到變數 age 當中。</p> <div style="background-color: #e0e0e0; padding: 5px;"> <pre>Console.WriteLine("請輸入您的姓名:"); name = Console.ReadLine();</pre> </div>

		<pre>Console.WriteLine("請輸入您的年齡:"); age = Console.ReadLine();</pre> <p>最後用 Write 將讀到變數裡的結果，整理出來印在螢幕上即可。</p> <pre>Console.Write(name + " 您好，您今年是 " + age + " 歲");</pre>																											
3. 介紹資料型態	15 分	<h4>四、執行結果</h4>  <p>一、整數 – sbyte, int, short, long</p> <p>C#語言提供了數個用來儲存整數的型態，使用的方法與儲存方式其實是一樣的，只是能儲存的整數範圍不同。範圍最大的 long 型態，使用的記憶體空間最多，需要 8 個 bytes；範圍最小的 sbyte 型態，只需要 1 個 byte 儲存；常用的 int 型態，則是用 4 個 bytes 儲存。</p> <p>sbyte、int、short、long 型態都代表了從 0 開始，往數線正負兩端延伸的一段範圍，不過相對於這些能儲存正負整數的資料型態，C#語言還提供了 byte, uint, ushort, ulong 等型態，這些對應的資料型態只用來儲存正整數，也因此能儲存的範圍相較於 sbyte, int, short, long 型態而言會延伸兩倍。</p> <table border="1" data-bbox="377 1237 1478 2061"> <thead> <tr> <th>資料型態</th> <th>所需記憶體與數值範圍</th> <th>範例</th> </tr> </thead> <tbody> <tr> <td>sbyte 有號單位元組整數</td> <td>1 byte -128 ~ 127</td> <td>sbyte num = -22;</td> </tr> <tr> <td>int 整數</td> <td>4 bytes -2147483648 ~ 2147483647</td> <td>int num = 65000;</td> </tr> <tr> <td>short 短整數</td> <td>2 bytes -32768 ~ 32767</td> <td>short num = -22222;</td> </tr> <tr> <td>long 長整數</td> <td>8 bytes -922337203685475808 ~ 9223372036854775807</td> <td>long num = 12345678901;</td> </tr> <tr> <td>byte 單位元組整數</td> <td>1 byte 0 ~ 255</td> <td>byte num = 222;</td> </tr> <tr> <td>uint 無號整數</td> <td>4 bytes 0 ~ 4294967295</td> <td>uint num = 32132123210;</td> </tr> <tr> <td>ushort 無號短整數</td> <td>2 bytes 0 ~ 65535</td> <td>ushort num = 65000;</td> </tr> <tr> <td>ulong</td> <td>8 bytes</td> <td>ulong num = 98765432109;</td> </tr> </tbody> </table>	資料型態	所需記憶體與數值範圍	範例	sbyte 有號單位元組整數	1 byte -128 ~ 127	sbyte num = -22;	int 整數	4 bytes -2147483648 ~ 2147483647	int num = 65000;	short 短整數	2 bytes -32768 ~ 32767	short num = -22222;	long 長整數	8 bytes -922337203685475808 ~ 9223372036854775807	long num = 12345678901;	byte 單位元組整數	1 byte 0 ~ 255	byte num = 222;	uint 無號整數	4 bytes 0 ~ 4294967295	uint num = 32132123210;	ushort 無號短整數	2 bytes 0 ~ 65535	ushort num = 65000;	ulong	8 bytes	ulong num = 98765432109;
資料型態	所需記憶體與數值範圍	範例																											
sbyte 有號單位元組整數	1 byte -128 ~ 127	sbyte num = -22;																											
int 整數	4 bytes -2147483648 ~ 2147483647	int num = 65000;																											
short 短整數	2 bytes -32768 ~ 32767	short num = -22222;																											
long 長整數	8 bytes -922337203685475808 ~ 9223372036854775807	long num = 12345678901;																											
byte 單位元組整數	1 byte 0 ~ 255	byte num = 222;																											
uint 無號整數	4 bytes 0 ~ 4294967295	uint num = 32132123210;																											
ushort 無號短整數	2 bytes 0 ~ 65535	ushort num = 65000;																											
ulong	8 bytes	ulong num = 98765432109;																											

	無號長整數	0 ~ 18446744073709551615	
--	-------	--------------------------	--

二、浮點數 – float, double, decimal

浮點數其實就是小數，基本上儲存浮點數所需的記憶體空間比整數大一些。float 型態需要 4 個 bytes 儲存，double 型態範圍較大，需要 8 個 bytes 儲存，decimal 型態更是用到了 16 個 bytes，如下表所示。

資料型態	所需記憶體與數值範圍	範例
float 浮點數	4 bytes $1.5 \times 10^{-45} \sim 3.4 \times 10^{37}$ 7 個數字精確度	float num = 12.345;
double 雙倍精確浮點數	8 bytes $5.0 \times 10^{-324} \sim 1.7 \times 10^{308}$ 15-16 個數字精確度	double num = 12.345;
decimal 十進位貨幣數	16 bytes $1.0 \times 10^{-28} \sim 7.9 \times 10^{28}$ 28 個數字精確度	decimal num = 1024.768;

讀者看到這邊可能就有懷疑了：**為何整數型態 int 和浮點數 float 都佔用 4 個位元組的記憶體，但 float 型態不但可以記錄小數而且數值範圍還比 int 型態大很多？**其實這是因為 float 所記錄的資料，並沒有包含所有的位數，它用類似於將 123456789 以 1.23e8 的方式儲存，如同我們在科學計算時，常捨去影響不大的位數一樣，即使有點誤差也無妨。另外，浮點數的存放格式是使用二進位，因此有時在十進位狀態可以用有限小數表達的數值，用二進位表示卻會是循環小數，這也是一個誤差來源。

不過為了符合某些運用，如：財務的計算(不容許一毛錢的誤差)，Visual C# 中提供了可以儲存小數且又不失真的資料格式，稱為 decimal 型態。除了不捨去任何位數外，在儲存上也用十進位來表示，以避免誤差。

也許您對於上面的內容還不是很了解，基本上初學者只需要知道哪些型態可以儲存小數以及哪些可以「符合」您所需要的數值範圍。若要使用得更恰當，未來您必須要多了解關於這些數值的儲存原理，並學著選擇使用「適合」的資料型態了。

三、字元與字串 – char, string

在 C# 當中，我們使用 char 字元型態來儲存單一個字元，其中儲存的是一個 Unicode 字元，包含了 26 個英文大小寫字母、阿拉伯數字，或是一些標點符號，甚至是控制字元，char 字元型態需要 2 個 bytes 的記憶體空間，宣告的方式如下：

```
char ch = 'A';
```

string 字串型態則可以看做是一連串的字元，所以他算是字元型態的一種延伸，畢竟在大部分的應用中，我們常常需要的是儲存「一句話」而非只儲存「一個字」，宣告的方式如下：

```
string str = "I love C# language";
```

四、布林值 – bool

布林值是一種邏輯概念的資料型態，只儲存兩種值，分別為 True(真)或是 False(假)，其所佔用的記憶體空間則依系統而異，宣告的方式如下：

```
bool val = true;
```

五、物件 – object

物件是一個比較特殊的類型，它可以存放任何型別的資料，如前面提到的：整數、字串或布林值...等等，指派給 object 型態的變數，預設值為 Nothing，也就是未指定任何物件實體，宣告的方式如下：

```
object myObject;
```

然後我們可以指派任意的型態值給物件，例如：`myObject = “Test”;`

第六堂課：程式練習

1.主控台
程式練
習：圓周計
算程式

20
分

一、程式設計目標

撰寫一個簡單的程式，程式目標是讀取使用者輸入的圓形半徑，然後計算出該圓形圓周的長度，請使用宣告常數的方式來設定圓周率 3.14159 為一浮點數常數，最後將圓周(半徑 x 2 x 圓周率)印在螢幕上，讓使用者得知結果即可。

二、程式碼撰寫

```
01 using System;  
02  
03 class Program  
04 {  
05     public static void Main()  
06     {  
07         int radius;  
08         const float pi = 3.14159F;  
09  
10         Console.WriteLine("請輸入半徑");  
11         radius = int.Parse(Console.ReadLine());  
12  
13         Console.WriteLine("所求圓周為" + radius*2*pi);  
14         Console.Read();  
15  
16     }  
17 }
```

三、程式碼解說

這個程式相當的簡短，一開始只要依照題目的要求，使用 int 宣告整數變數 radius 為半徑、以及使用 const float 宣告浮點數「常數」pi，也就是圓周率 3.14159，**不過這裡特別注意後面多加了一個「F」**，因為一般含小數點的數字都會被 C#以雙倍精度浮點數 double 型態來處理，在後面加上 F，可以提示 C#這是一個一般浮點數 float，如此一來才能順利將值存入 float 常數 pi 當中。

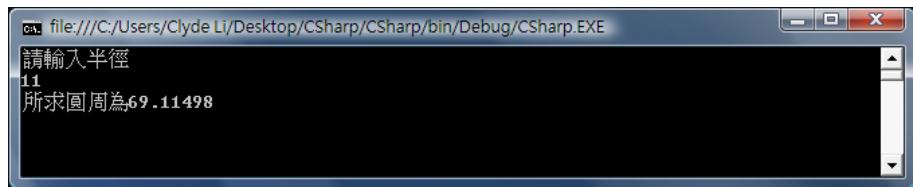
```
int radius;  
const float pi = 3.14159F;
```

接下來的部分，就是要讀取使用者輸入的半徑，使用 WriteLine 及 ReadLine 方法即可輕鬆完成這個部分，忘記型態轉換用法的讀者也可以趕快趁機複習一下喔，最後再使用一次 WriteLine 方法，直接將「radius*2*pi」結果印出即可。

```
Console.WriteLine("請輸入半徑");  
radius = int.Parse(Console.ReadLine());
```

```
Console.WriteLine("所求圓周為" + radius*2*pi);
```

四、執行結果



2. 視窗程式練習：簡易計算機程式

30 分

一、程式設計目標

設計一個簡易計算機程式，可以輸入兩個數字，並用按鈕進行兩數的「加」、「減」、「乘」、「除」四則運算以及比較兩數是否「等於」、「不等於」、「大於」、「小於」等功能。

二、表單配置

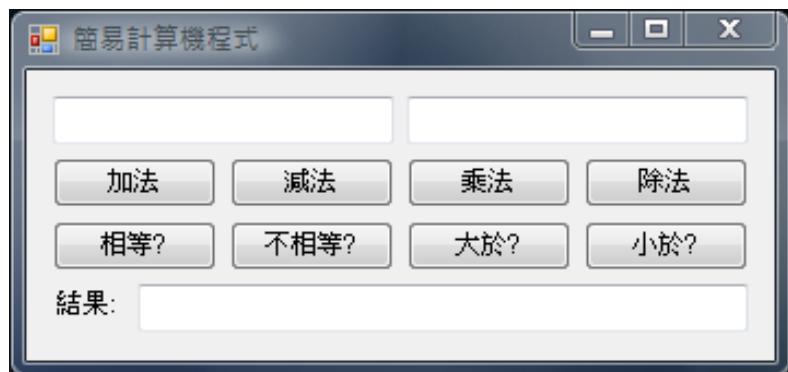
新增 3 個 TextBox、8 個 Button 與 1 個 Label，大約配置如下圖。



然後依照以下表格更改各物件屬性值：

物件	Name 屬性	Text 屬性
textBox1	Val1	*不用變更
textBox2	Val2	*不用變更
textBox3	resultVal	*不用變更
button1	addButton	加法
button2	minButton	減法
button3	mulButton	乘法
button4	divButton	除法
button5	eqlButton	相等？
button6	noteqButton	不相等？
button7	biggerButton	大於？
button8	smallerButton	小於？
label1	resultLabel	結果：

更改後的表單配置大約如下：



三、程式碼撰寫

```
01 private void addButton_Click(object sender, EventArgs e)
02 {
03     resultVal.Text = (int.Parse(Val1.Text) + int.Parse(Val2.Text)).ToString();
04 }
05
06 private void minButton_Click(object sender, EventArgs e)
07 {
08     resultVal.Text = (int.Parse(Val1.Text) - int.Parse(Val2.Text)).ToString();
09 }
10
11 private void mulButton_Click(object sender, EventArgs e)
12 {
13     resultVal.Text = (int.Parse(Val1.Text) * int.Parse(Val2.Text)).ToString();
14 }
15
16 private void divButton_Click(object sender, EventArgs e)
17 {
18     resultVal.Text = (int.Parse(Val1.Text) / int.Parse(Val2.Text)).ToString();
19 }
20
21 private void eqlButton_Click(object sender, EventArgs e)
22 {
23     resultVal.Text = int.Parse(Val1.Text) == int.Parse(Val2.Text) ? "是" : "否";
24 }
25
26 private void noteqlButton_Click(object sender, EventArgs e)
27 {
```

```

28     resultVal.Text = int.Parse(Val1.Text) != int.Parse(Val2.Text) ? "是" : "否";
29 }
30
31 private void biggerButton_Click(object sender, EventArgs e)
32 {
33     resultVal.Text = int.Parse(Val1.Text) > int.Parse(Val2.Text) ? "是" : "否";
34 }
35
36 private void smallerButton_Click(object sender, EventArgs e)
37 {
38     resultVal.Text = int.Parse(Val1.Text) < int.Parse(Val2.Text) ? "是" : "否";
39 }

```

四、程式碼解說

依照前一章設定事件的方法，我們用最簡單的方式，用滑鼠左鍵分別雙擊每個按鈕後，得到基本程式碼架構，再分別為每個事件輸入程式碼，我們先以雙擊「加法」按鈕的程式碼作為範例：

```

private void addButton_Click(object sender, EventArgs e)
{
    int result = int.Parse(Val1.Text) + int.Parse(Val2.Text);
    resultVal.Text = result.ToString();
}

```

這邊要特別注意的是 `int.Parse(var)` 以及 `ToString()` 方法的使用，由於文字方塊的 `Text` 屬性為字串(`String`)，故我們必須利用 `int.Parse(var)` 先把 `Val1` 和 `Val2` 兩個文字方塊的內容，先轉換成整數才能正常的進行計算，最後要把結果指派到 `resultVal` 文字方塊的 `Text` 屬性時，也需要利用 `ToString()` 方法將整數型態還原回字串型態。接著我們可以更簡化這段程式碼為：

```

private void addButton_Click(object sender, EventArgs e)
{
    resultVal.Text = (int.Parse(Val1.Text) + int.Parse(Val2.Text)).ToString();
}

```

依此類推我們可以完成「減」、「乘」、「除」的程式碼如下：

```

private void minButton_Click(object sender, EventArgs e)
{
    resultVal.Text = (int.Parse(Val1.Text) - int.Parse(Val2.Text)).ToString();
}

```

```

private void mulButton_Click(object sender, EventArgs e)
{
    resultVal.Text = (int.Parse(Val1.Text) * int.Parse(Val2.Text)).ToString();
}

private void divButton_Click(object sender, EventArgs e)
{
    resultVal.Text = (int.Parse(Val1.Text) / int.Parse(Val2.Text)).ToString();
}

```

接著是「等於」、「不等於」、「大於」、「小於」這四個按鈕的部份，這邊我們用到了三元運算子來幫助我們撰寫程式碼，以「等於」按鈕的程式碼為例：

```

private void eqlButton_Click(object sender, EventArgs e)
{
    resultVal.Text = int.Parse(Val1.Text) == int.Parse(Val2.Text) ? "是" : "否";
}

```

一樣的，我們可以如法炮製「不等於」、「大於」、「小於」這三個按鈕的程式碼：

```

private void noteqlButton_Click(object sender, EventArgs e)
{
    resultVal.Text = int.Parse(Val1.Text) != int.Parse(Val2.Text) ? "是" : "否";
}

private void biggerButton_Click(object sender, EventArgs e)
{
    resultVal.Text = int.Parse(Val1.Text) > int.Parse(Val2.Text) ? "是" : "否";
}

private void smallerButton_Click(object sender, EventArgs e)
{
    resultVal.Text = int.Parse(Val1.Text) < int.Parse(Val2.Text) ? "是" : "否";
}

```

如此一來我們就完成了所有程式碼的撰寫了，最終執行的畫面如下：

