第 3 章　小小電子琴

簡介：

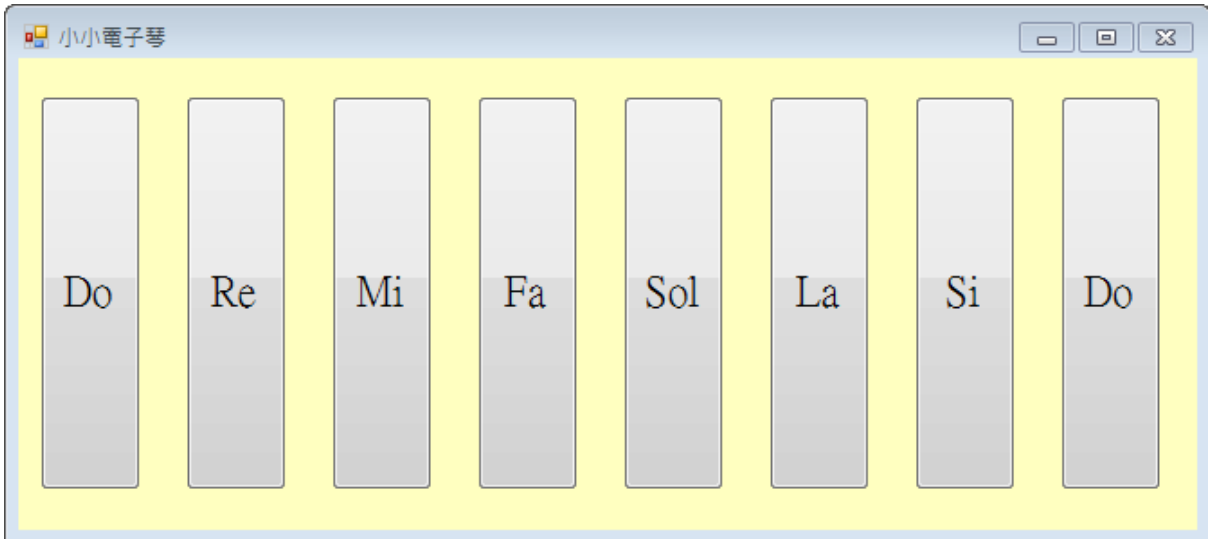
這一章要製作一個可以發出不同音調的小小電子琴程式，主要目的是讓同學學習如何製作不同外觀的按鍵(物件)，並建立按鍵的程式碼，以及製作鍵盤事件的程式。也就是說完成的電子琴程式可以使用滑鼠點擊演奏，也可直接用電腦鍵盤彈奏！有趣的是，即使你的電腦沒有喇叭，甚至沒有音效卡！這個程式還是能有效的彈奏，發出正確的聲音！神奇吧？

事實上這個程式很簡單！簡單到不太足夠當作一次上課的內容，所以本單元還要加碼教大家製作一個小時鐘以及一個會閃爍的節拍器，讓大家學習使用程式自動控制事件的機制，以及基本的 C#時間函數。

3-1 製作鍵盤

[鍵盤介面的設計]

請先開啟一個新專案，在表單上建立如下的八個鍵盤，使用的物件是工具箱的 Button，接著請用滑鼠拖曳或者修改屬性 Size->Wdith(寬度)與 Size->Height(高度)的方式將它變成接近電子琴按鍵的外觀。當然還必須修改 Text 屬性讓它顯示 Do, Re, Mi…等等，也可以用 BackColor 屬性修改它們的顏色。要一一製作八個鍵盤會花掉很多時間，有效率的做法是先細心的作好一個自己滿意的按鍵，再使用複製貼上的方式產生八個按鍵，之後你只要修改個別按鍵上的文字就可以了！



[鍵盤設計的其他可能]

如果你非常有創意與審美觀，想讓鍵盤更美一點，可以試試使用 BackgroundImage 的屬性，用圖片當作鍵盤的背景，就可以做出非常美又有質感的鍵盤。當然用影像使得

表單背景變成鋼琴的樣子也不錯，表單也有 BackgroundImage 屬性的。另一方面，也不一定要使用 Button 物件來製作鍵盤，如果用 Label，Picturebox 等等其他可以設定顏色、影像或文字的物件也都可以，重點是該物件必須可以產生 Click 事件，也就是要對滑鼠事件有反應就行了！

3-2 Beep 函數與蜂鳴器

[呼叫蜂鳴器]

接下來請先雙擊第一個按鍵，寫入以下程式：

```
private void button1_Click(object sender, EventArgs e)
{
    Console.Beep(262, 500);
}
```

試試看執行程式並點擊此按鍵，會聽到電腦發出一個「嗶」聲。有趣的是它並不是來自喇叭，而是電腦主機板內建的蜂鳴器。相信各位都聽過電腦開機時常常會發出「嗶」的一聲，有故障時還會連續嗶好幾聲(即使你的電腦並沒有喇叭或者音效卡)，那就是它在叫了！蜂鳴器是一個主機板上的古老元件，目的是讓電腦出現異常時可以發出一些聲音來提示工程師可能的狀況。只是很少人注意到它其實可以用程式控制，發出不同音高與持續時間的聲音。

[Console 是甚麼？]

Console 一般翻譯為「主控台」，在此你可以將它視為電腦的核心硬體，Console.Beep 就是請電腦硬體(在此就是蜂鳴器)發出一個 Beep 的聲音。附帶的兩個參數，第一個是控制它的頻率，單位是 Hz，就是每秒震動幾次的意思；第二個參數是音響延續的時間，單位是毫秒，就是千分之一秒。以後你會發現寫程式時多數的時間控制單位都是毫秒。所以上面的程式 Console.Beep(262, 500)就是請蜂鳴器發出頻率 262Hz，時間延續半秒鐘的音響！這是理論上鋼琴上中央 C 的頻率，就是八度音階上的某一個 Do 啦！下表列出其他音高的理論頻率：

Do	Re	Mi	Fa	Sol	La	Si	Do
261.63	293.66	329.63	349.23	392.00	440.00	493.88	523.26

[音高的控制]

不過必須注意到 Beep 函數的參數只接受「整數」，所以要寫為程式碼之前必須先將音高理論值四捨五入一下。那麼如果我想發出更高或更低的聲音呢？唱歌也得有兩三個八度的音域吧？很簡單！只要記得所謂的「高八度」就是將音頻乘以 2，低八度就是音頻的一半了！看看上表的資料，高低音的兩個 Do 是不是剛好為兩倍的關係？其他的音也都照樣處理就可以了！但是蜂鳴器或任何音響器材都有音域範圍，正如逼人唱太高或太低的聲音，不是唱不出來就是聲音非常小。給 Beep 物件太高或太低的音高也會變得

太小聲而聽不到。如果你希望還可以控制「音量」，那就超過一般蜂鳴器的能力了！

3-3 滑鼠按鍵程式

[撰寫鍵盤按鍵程式]

了解蜂鳴器的原理之後要寫出音響程式應該就很容易了！請一一雙擊各個鍵盤，切換到程式碼頁面，可以產生個別 button 物件的 Click 事件程式框架，寫入以下程式：

```
private void button2_Click(object  sender, EventArgs e)
{
    Console.Beep(294, 500);
}

private void button3_Click(object  sender, EventArgs e)
{
    Console.Beep(330, 500);
}

private void button4_Click(object  sender, EventArgs e)
{
    Console.Beep(349, 500);
}

private void button5_Click(object  sender, EventArgs e)
{
    Console.Beep(392, 500);
}

private void button6_Click(object  sender, EventArgs e)
{
    Console.Beep(440, 500);
}

private void button7_Click(object  sender, EventArgs e)
{
    Console.Beep(493, 500);
}

private void button8_Click(object  sender, EventArgs e)
{
    Console.Beep(523, 500);
}
```

試試看！執行程式你已經可以用滑鼠點擊按鍵演奏這個小小電子琴了！

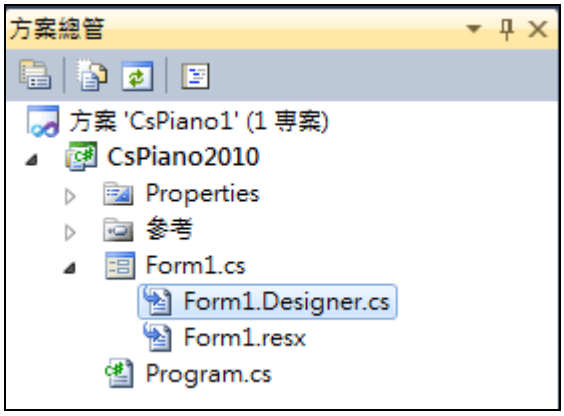
[善用複製才是王道]

相信在寫上面程式時已經有很多聰明的同學開始用複製程式碼的技巧了！譬如將 Console.Beep(262, 500)複製到每個事件框架內，再修改頻率就好了！恭喜你！你做對了！老師不會罵你偷懶的！這是值得鼓勵的正確作法！少打字就少出錯，可以更快且「更正確」的完成工作。事實上我們使用工具箱物件時就是在複製之前的人寫好的程式！如果沒有很多的複製技巧，現代文明可能根本無法出現，或者正常運作的！

在電腦剛剛起步的 1960 年代曾有人憂心的說：「照這樣程式越來越複雜的趨勢，到了 20 世紀末，全世界的人都在寫程式也敢不上實際的軟體需求了！」還稱之為『軟體危機』咧！但是現在都 21 世紀了，並沒發生這種危機，寫程式仍然只是少數人用來賺錢的行業，多數人都不會寫程式，但也都快樂的享受著各種程式的服務。主要原因就是我們學會了「複製」的精隨！不必一直重複前人的工作。在物件導向程式設計理論中還有個美美的名稱叫做「繼承」！

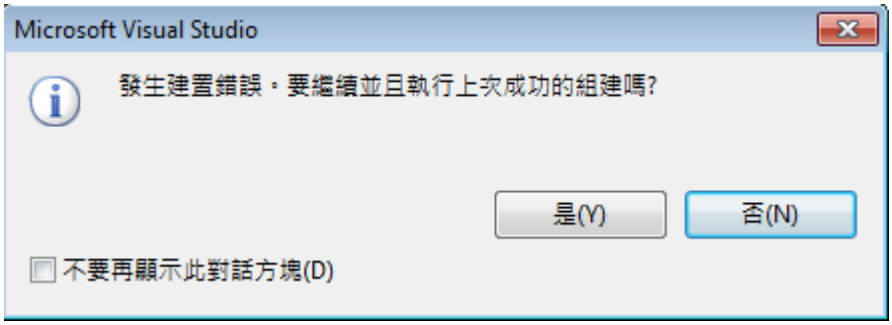
[事件副程式的框架不能複製]

不過也請小心不要複製玩過頭，將事件副程式的**框架**也一起複製，在 C#程式製作中這是一定不行的！原因是專案之內除了你看到的檔案之外還有一些建構檔(Designer)，平常由 Visual Studio 軟體自動維護，你用合法方式(如雙擊物件)產生程式框架時這些資訊會寫入建構檔案，如果你直接複製程式碼就會讓 Designer 不知道已經有了這個副程式，導致執行時不正常。你可以到方案總管的視窗將 Form1 前面的小三角形點一下就可以看到這些檔案了！如下圖反白的就是表單平時隱藏的建構檔，可以打開看看，你在設計頁面作的所有事情，包括產生的事件副程式都可以在此看到紀錄。

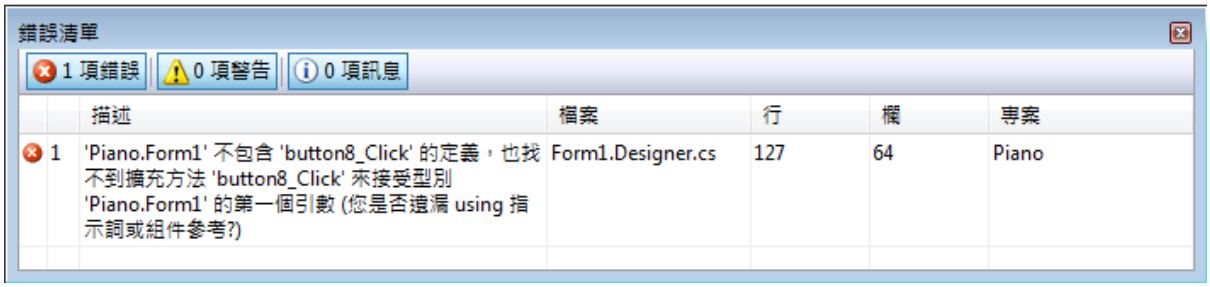


[刪除事件副程式時必須注意的事]

同樣的道理，如果你依循正常的方式產生了程式框架，稍後又覺得不必用到它，而將該段程式碼刪除，譬如你可以試試看將剛剛寫好的 button8_Click 刪除或標示為註解。結果執行程式時會出現錯誤提示視窗如下圖，請記得此時必須選「否」，先讓程式停止！看看錯在哪裡！



選否之後你該去的地方是「錯誤清單」視窗，如果沒看到就到功能表的「檢視」去找出來！如下圖：



上面很奇怪的說找不到 button8_Click 的定義？我不是剛剛把它刪掉了嗎？電腦怎麼還在找它？仔細看看出錯的地方其實不是你目前寫程式的 Form1.cs 檔案，而是在建構檔 Form1.Dsigner.cs 檔，你可以直接雙擊上面的錯誤訊息，畫面就會直接跳到錯誤發生的地方，畫面如下：

```
//
// button8
//
this.button8.Font = new System.Drawing.Font("新細明體", 14.25F, System
this.button8.Location = new System.Drawing.Point(610, 33);
this.button8.Name = "button8";
this.button8.Size = new System.Drawing.Size(50, 190);
this.button8.TabIndex = 7;
this.button8.Text = "Do";
this.button8.UseVisualStyleBackColor = true;
this.button8.Click += new System.EventHandler(this.button8_Click);
```

也就是在你雙擊 button8 產生程式框架時，不僅在 Form1.cs 檔案內產生程式，在 Form1.Dsigner.cs 裡面也多了一行程式，所以要刪除程式時兩個檔案內的程式都必須同時刪掉。這種事情其實常常碰到，對於初學者常常造成困擾，希望下一版的 C#可以自動處理這種事情。

3-4 用鍵盤彈琴的程式

[設定使用鍵盤事件]

視窗程式以「滑鼠」操作為主不讓人意外，但是有點奇怪的是：它預設是「不使用鍵盤」的！可能是這樣比較節省資源吧？所以要寫鍵盤操作的相關程式必須先修改預設值，就是表單(Form1)屬性中的 **KeyPreview** 屬性，預設為 False 必須修改為 True。電腦鍵盤的全名叫 Keyboard，在程式軟體中習慣以 Key 代表鍵盤。

[撰寫鍵盤事件副程式]

接下來請到表單的屬性頁，點選閃電標誌進入事件選單，選擇 KeyDown 事件產生副程式框架，寫入以下程式：

```
private void Form1_KeyDown(object  sender, KeyEventArgs e)
{
    switch (e.KeyCode)
    {
        case Keys.D1:
            Console.Beep(262, 500);
            break;
        case Keys.D2:
            Console.Beep(294, 500);
            break;
        case Keys.D3:
            Console.Beep(330, 500);
            break;
        case Keys.D4:
            Console.Beep(349, 500);
            break;
        case Keys.D5:
            Console.Beep(392, 500);
            break;
        case Keys.D6:
            Console.Beep(440, 500);
            break;
        case Keys.D7:
            Console.Beep(493, 500);
            break;
        case Keys.D8:
            Console.Beep(523, 500);
             break;
    }
}
```

[多選項的 switch 語法]

這裡使用到一個多選項，但是答案為單選的語法結構，以 switch (e.KeyCode)起始，其中 switch 表示切換，括號內是切換時依據的參數，e 在此表示 KeyDown 事件發生時會記錄的參數群，當然使用者按了哪一個鍵盤的資訊是一定在裡面的，就是 e.KeyCode 了！接下來使用一組大括號界定選項範圍，每一個 case 就是一種狀況，不同狀況出現(按不同鍵)時處理方式不同，就據此寫不同的程式了。但是要小心每個案例結束必須寫一個 break！作用是跳離這個選擇結構，因為是單選題嘛！只會有一個狀況是合適的，找到對的那個選項執行程式碼之後就不必再往下執行其他的 case 了！

[參數集合 e 的意義]

說到前面那個 e，不同的事件會夾帶不同的參數群(或稱集合)，所以不同性質的事件副程式中參數集合 e 的內容未必相同，譬如鍵盤按下時會夾帶使用者按的是「哪個按鍵？」的訊息，KeyCode 就是鍵盤的號碼。根據不同的鍵盤號碼(e.KeyCode)我們就可以決定要電腦作不同的事情了！

[物件命名規則：不能用數字作開頭]

至於 Keys.D1 的意義呢？Keys 是所有的鍵盤代碼常數的集合，在它之後打個小數點就會看到所有鍵盤碼的提示選項。那麼 D1 是哪個鍵呢？其實是鍵盤上方橫排的數字

"1"，因為程式設計的命名規則中不允許用數字做為任何物件或變數名稱的第一個字，所以就替所有 0~9 的數字前面加一個 D(Digit 數字)。如果是標準鍵盤右邊的方塊數字鍵盤上的數字就是 NumPad1~ NumPad9 了，這些鍵在多數的筆記型電腦上就沒有了！

上面程式讓 1~8 的數字鍵分別對應到八個音階的發聲，執行程式就可以使用數字鍵彈奏電子琴了！但是此時一定有些人的程式寫得一模一樣卻沒有聲音？這通常是他的電腦處於中文輸入狀態啦！在多數的中文輸入法狀態下鍵盤代碼會變得不一樣的！切換回到英數字模式就可以了。還是不行嗎？我猜一定是你忘了改 **KeyPreview** = True 吧？

[為何不用 button_KeyDown 事件？]

鍵盤事件的程式與滑鼠事件不同的是：滑鼠事件寫在 **button1**.Click 事件中，鍵盤事件卻不是寫在 **button1**.KeyDown 事件之中，為什麼？雖然 button1 等物件也有 KeyDown 事件，但是這些事件必須是程式執行期間「焦點」停在此物件上時才會有反應！譬如你剛剛用滑鼠點了 Do 的按鍵，程式的焦點就會停在 button1 上面，此時你寫在 button2.KeyDown 事件內的程式碼就沒有反應，拼命按數字鍵 2 也不會發出 Re 的聲音。所以我們的 KeyDown 程式必須寫在 Form1 的事件當中，表單是所有物件的母體，它的事件就可以隨時指揮所有的事情，不受物件焦點的影響了！

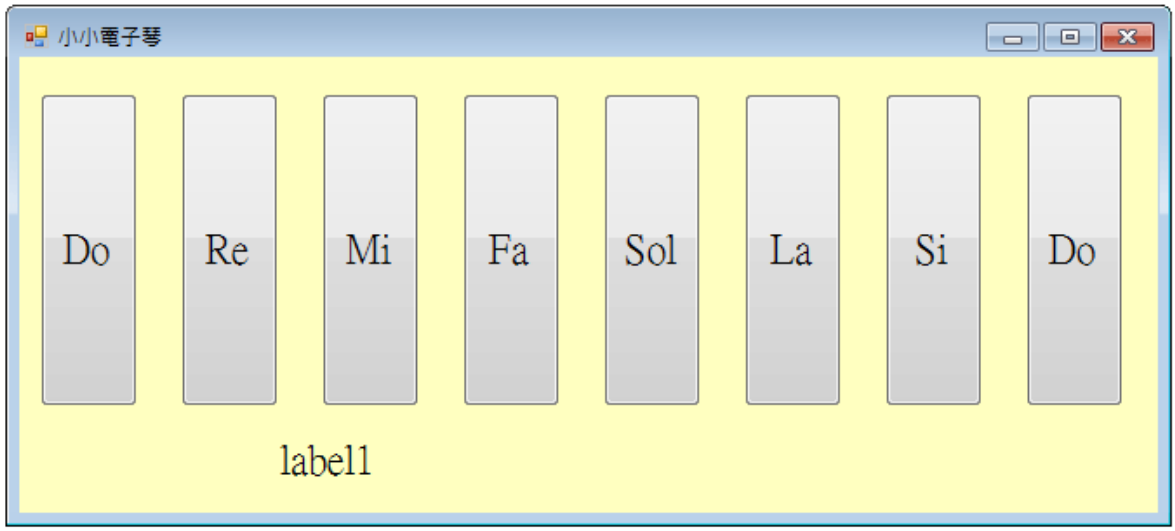
[使用編輯→格式化文件]

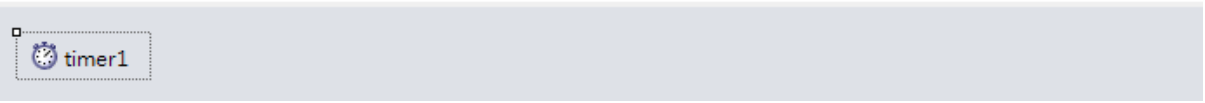
還有一點小技巧必須告訴各位，相信很多人在寫前面的程式碼時都會使用複製技巧，但是奇怪老師的程式碼為何總是很整齊，層次分明？我們直接複製後多半都是格式很亂的，一一調整很費事欸！其實有工具可以用的，到功能表「編輯」→「格式化文件」按一下就自動排整齊了！但是多餘的空行不會自動刪掉。

3-5 小時鐘

[使用計時器物件]

接下來請到物件設計頁面，在適於顯示時間的地方，加入一個標籤物件(label1)，接著再到工具箱的「元件」分類下面找到一個 Timer(計時器)物件加入表單，它是一個程式執行期間不可見的元件，所以會被放到表單下面的一個灰底的區域內，不會落在表單上。如下圖：

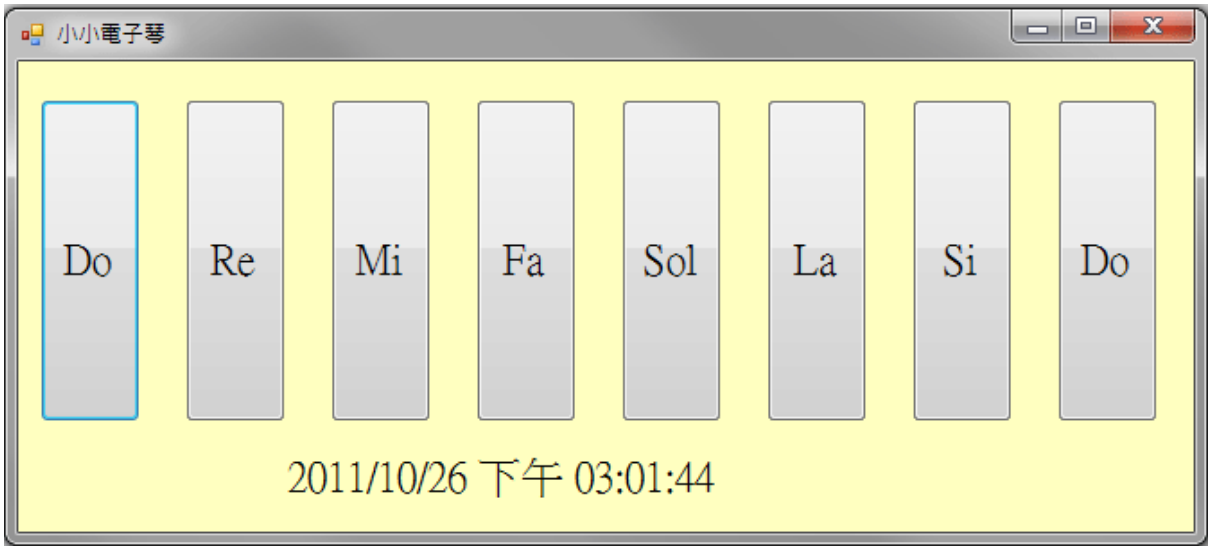




點選一下計時器(timer1)將屬性 Interval 改成 1000，它的單位也是毫秒，1000 就是一秒鐘執行一次的意思，我們要用它來控制小時鐘更新時間。接著請雙擊 timer1 物件進入它的事件副程式(timer1_Tick)，寫程式如下：

```
private void timer1_Tick(object sender, EventArgs e)
{
    label1.Text = DateTime.Now.ToString();
}
```

上述程式碼的意義是將現在 (Now)的時間顯示於 label1 的 Text(文字)之中，DateTime 表示此資料的型態是「日期時間」。但是這時候執行程式卻不會有任何動靜，原因是 Timer 物件是有「**開關**」的！如果直接將它的屬性 Enabled 改為 True 再執行程式，時間就會出現而且每秒鐘都會更新了！執行畫面大致如下：



[在 Form_Load 設定初始狀況]

不過還是有點不理想，程式一開始時不會立即顯示時間，而是顯示一個預設的 label1 文字。簡單的處理是將 label1 的預設文字拿掉，但是太賴皮了吧？應該一開始就有時間啊？如何指揮程式一開啟表單就立即作事情呢？應該是將程式寫在 Form1 事件的 Load(載入)事件之內。我們可以用產生 KeyDown 事件相同的方式產生 Load 程式框架，但是因為通常有太多事情都需要在表單起始時先作，所以它是「預設」的表單事件，你只要在物件設計的頁面如同點 button1 產生 button1_Click 一樣，雙擊表單空白處就會產生 Form1_Load 框架了！寫入程式如下：

```
private void Form1_Load(object sender, EventArgs e)
{
    label1.Text = DateTime.Now.ToString();
    timer1.Start();
}
```

第一行與 timer1 裡面的程式一樣，第二行是啟動計時器的另一種方式(Start 方法)，其實與設定 timer1 的 Enabled=True 效果一樣！當然如果你這樣寫了，設計階段就不必改 timer1 的 Enabled 屬性為 True 了！試試看，應該一開始就有時間顯示了！

[用程式讓物件置中]

還有個小問題，時鐘的文字顯示好像位置有點偏，如果我在設計時將標籤置中好像沒用，因為我不知道程式插入時間之後文字會有多長，煩惱嗎？你可以用程式動態的將標籤置中的。請加入一行程式如下：

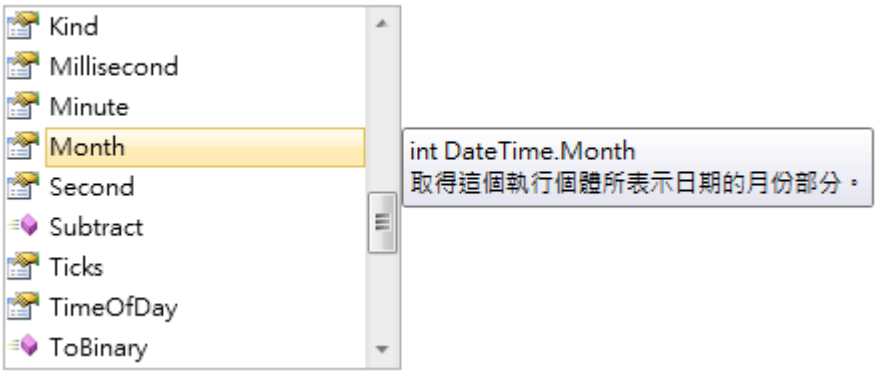
```
private void Form1_Load(object sender, EventArgs e)
{
    label1.Text = DateTime.Now.ToString();
    timer1.Start();
    label1.Left = (this.ClientSize.Width - label1.Width) / 2;
}
```

如前一單元教過的 this.ClientSize.Width 是視窗可編輯區域的寬度，減去標籤加入時間訊息之後的寬度(label1.Width)，再除以 2 就是標籤置中時的左邊(Left)應該有的座標。這是常常需要使用到的物件動態置中的程式技巧。改變 Left 屬性就是使物件左右移動的意思！

[時間函數的其他選項]

C#的時間函數當然不只是一個 Now 而已，你可以試試看寫程式時在 Now 的後面打個小數點會看到很多的時間相關選項，舉凡年月日時分秒甚至星期幾或格林威治時間都有！如下圖：

Now.



3-6 節拍器

[製作閃爍的物件]

編輯此書時書局的主編小姐說此單元加入小時鐘有點不搭，電子琴嘛！應該弄個節拍器才對呀！好主意，我們就在此使用計時器物件作個會閃爍的物件當節拍器好了！請先在表單右下角加入一個 label2 物件，並將它的 BackColor 變成黑色，看起來它就是一個黑色方塊了，如果你還想調整它的形狀，可以將 AutoSize 屬性改成 False(預設為 True)，就可以任意調整長寬了。接著再叫用一個 timer2 設定 Interval=500(半秒鐘執行程式一次)，設定 Enabled=True 讓此計時器一開始就執行，程式碼如下：

```
private void timer2_Tick(object sender, EventArgs e)
{
    label2.Visible = !(label2.Visible);
}
```

此處用到一個有趣的『函數』就是一個驚嘆號("！")，在 C 語言中這是相反的意思，Visible 英文原意是『可見的』，如果 Visible=False 就是隱藏了！讓 Visible 一再的變成相反就是 True/False 交替，使物件時有時無，試試看程式會有一個像節拍器的東西一直在閃爍了！不會閃嗎？你應該是忘了啟動計時器(timer2)吧？

3.7、進階挑戰

一、如何建立多個八度的電子琴程式？
　　提示：音頻乘以二為高八度音。

二、如何讓時間只顯示時間不要日期？
　　提示：請尋找 Now 的屬性值選項。

課後閱讀

本單元除了介紹有趣的蜂鳴器之外，最重要的新技術是教大家如何寫鍵盤事件程式以及計時器物件的使用，我們在此稍微深入談談這兩項技術。

有關鍵盤事件

雖然滑鼠是視窗程式的主要操作介面，但是鍵盤控制在很多時候還是無可取代，它有操作快速以及明確的優勢。寫鍵盤事件程式首先必須注意到的兩個機關是：第一、視窗程式預設是「不」使用鍵盤事件的！所以如果你忘了將表單屬性的 KeyPreview 屬性改設為 True，寫的程式就會沒有作用。第二、雖然多數的物件都有鍵盤相關的事件，如 KeyDown，KeyUp 等等，但是都必須先取得物件的焦點(Focus)，否則寫的程式還是沒反應。譬如本單元的鍵盤彈奏程式如果寫在個別的 button 事件之中，而眾多的 button 物件又總是只有一個會取得焦點，彈奏時就只有一個音符會響了！

[鍵盤碼真的是數字]

因此不可避免的，你要替多個鍵盤按鍵寫不同的回應程式時就必須通通擠在同一個事件副程式就是 Form1_KeyDown 之中！這時首先必須讓電腦知道你按的是哪一個按鍵？此時鍵盤事件的參數 e.KeyCode(鍵盤碼)就是主角了！事實上這些「鍵盤碼」是一些整數的數字，通常對應於 ASCII 編碼。但考慮到應該沒有人會去背誦空白鍵的代碼是 32 號，逗點是 44 號等等。所以 C#建立了文字化的常數群，以 Keys 開頭，譬如 Keys.Enter 你一定知道是表示 Enter 鍵！當然如果你也知道 Enter 鍵的鍵盤碼是 13，直接寫 13 取代 Keys.Enter，電腦也看得懂的！

接下來程式必須依據不同的按鍵作出不同反應，此時 switch 的語法就很重要了。它與一般單選的選擇題意義相似，所謂的 case 是「案例」，就是宣告我們要依據哪一個參數決定到哪一個軌道去處理此案例？在此的參數當然就是 KeyCode 了！再回頭看一下下面的程式碼，是不是更有感覺了呢？

```
switch (e.KeyCode)
{
    case Keys.D1:
        Console.Beep(262, 500); break;
    case Keys.D2:
        Console.Beep(294, 500); break;
    case Keys.D3:
        Console.Beep(330, 500); break;
    case Keys.D4:
        Console.Beep(349, 500); break;
    case Keys.D5:
        Console.Beep(392, 500); break;
    case Keys.D6:
        Console.Beep(440, 500); break;
    case Keys.D7:
        Console.Beep(493, 500); break;
    case Keys.D8:
        Console.Beep(523, 500); break;
    default:
```

```
MessageBox.Show("你按錯了！"); break;                }
```

仔細看看和前面的程式寫法好像稍有不同？在此順便介紹一下";"(分號)在 C#中的的意義。分號是多數 C 語言指令結束的標記，其實多數的 C 語言編輯器根本不理會分行字元！所以如果你的程式硬是不分行，將所有指令串成一串，只以分號區隔，也是可以執行的！只是這樣程式會很難閱讀而已。還有最後一個與 case 平行的選項是 default，意義是預設值，如同選擇題常見的「以上皆非」選項！如果按錯鍵時要警告使用者，這個語法就很好用了！

計時器很重要

[程式自動化的主角]

程式很多時候需要產生由電腦主動的動作，譬如本單元的小時鐘，總不能讓使用者點一下某個物件才更新時間吧？又譬如很多遊戲是由電腦持續產生動作讓使用者反應接招的，既然我們的程式必須由「事件驅動」，那麼甚麼事件可以持續提醒電腦作動作呢？答案就是計時器(Timer)物件了！

計時器在程式執行過程中不會被看見，它的屬性也非常簡單，必須操作的通常只有 Interval(時間間隔)以及 Enabled(開啟與否)。它的事件通常也只有 Tick 會被用到，就是指每一次指定的間隔時間到的時候要作的動作。這個看起來簡單的物件其實是很多程式內部運作的靈魂，所有必需由程式主導的動作都必須由它掌控，我們之後的很多單元都會用到它！

[合理的設定計時間隔]

使用 Timer 很容易，但是必須注意到它無形中耗損的電腦資源，譬如預設的 Interval 值是 100，相當 0.1 一秒，就是一秒鐘會執行 Timer 事件副程式 10 次！以小時鐘來說，系統預設的時間顯示最小單位是一秒鐘，不改 Interval 預設值(0.1 秒)表示電腦會連續顯示 10 次一樣的時間，即使真的可以顯示 0.1 秒的差異，我們的眼睛是不是來得及閱讀也要考慮。無謂的電腦資源浪費會讓我們的程式變得比較沒效率，也拖慢整台電腦的運作，因此合理的 Interval 設定很重要。

[動畫的計時間隔]

程式設計中 Timer 也常用於製造動畫的效果，要讓動畫看起來連續平順當然 Interval 越小越好，但是實際上人的眼睛有視覺暫留的現象，每秒鐘只要超過 20 格的動畫，應該所有人都無法感覺有何差異！因此將動畫間隔調到 Interval=50，相當一秒 20 格的影像，動畫效果就很好了！甚至很多時候你會發現這樣的間隔下，電腦來不及完成動作，還是會停格的！那你就必須調得更慢。以 Flash 動畫來說，預設就只有一秒 12 格，相當 Interval=83。總之不要以為電腦耐操就隨便將 Interval 調得太小，這是無謂的浪費！

[神經反射的時間]

另一方面，有個數字也可以參考，就是人類的反射動作最快約 0.1 幾秒，如果你要作一個考驗人反應速度的遊戲程式，用 Timer 控制電腦畫面的變換速度，設定 Interval=100，效果大概就是所有人都看得到卻夠不著！150 呢？也許就剛剛好很好玩，因為有時來得及有時來不及，趣味性就出來了！當然如老師這種 LKK 可能要調得更慢一點才行了！

所以給大家使用 Timer 的 Interval 時參考的關鍵數字是：50 剛好來得及「看到」，150 剛好來得及反應(可能是按鍵盤或點滑鼠的動作)！