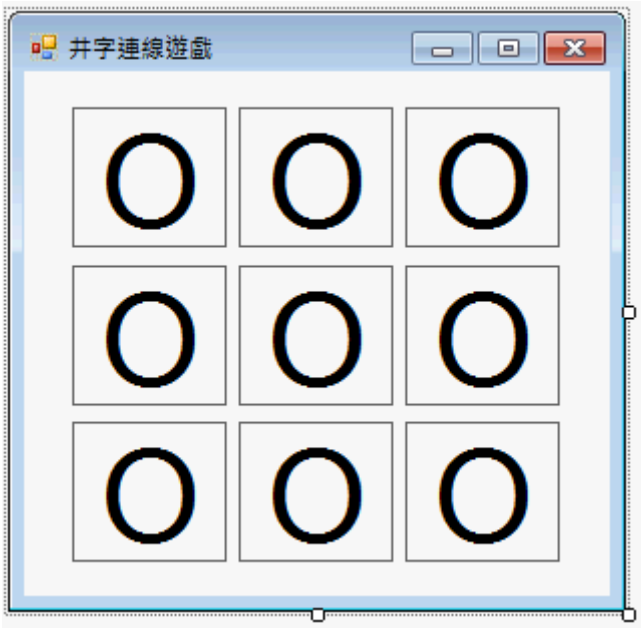
# 第 10 章井字連線遊戲

## 簡介：

遊戲程式事實上會使用到相當複雜的程式技巧，因為互動性高，也可以讓程式設計的學習增加許多趣味，自本章起我們將以遊戲設計為主軸，繼續介紹更多的程式設計技巧與概念。首先是較為靜態的棋盤類遊戲，這類遊戲的介面設計通常不難，較困難的常常是判斷遊戲勝負與限制下棋規則的程式碼。本單元將以最簡單的井字圈叉連線遊戲作為開始。

### 10-1 建立井字連線遊戲畫面

建立專案之後請先加入一個 Label 物件，將字型(Font)放大到 48，AutoSize 屬性設定為 False，BorderStyle 設定為 FixedSingle(單線固定)，文字(Text)改成英文大寫的"O"。緊接著將做好的這個標籤複製為九個，呈九宮格狀，如下圖：




這個過程中有許多製作大量物件時必須知道的小技巧，首先是如果你要設計很多相似外觀的物件，請千萬不要先複製一堆原始樣貌的物件再一一修改，這樣會浪費很多時間。相信很多讀者在之前的電子琴單元已有體會，應該先製作好一個完整的物件再一一複製。「一一複製」也不太對，請記住複製其實可以「大批進行」！以上例來說，可以先複製好三個之後全部選取再一次複製三個，數秒鐘之內就可以完成九個物件的複製了！對齊的部分也很重要，千萬不要一個一個拖曳，既費時也對不整齊，應該選取要對齊的一排或一列，再使用功能表的「格式」→「對齊」的選項。

接下來請將上面九個標籤預設的 Label?的名稱(Name 屬性)一一修改為 C1~C9。記得必須按照「由上而下，由左而右」的次序，左上角是 C1，右下角是 C9。這樣作有兩

個目的，首先是稍後我們會寫程式判斷連線狀態，物件名稱必須有規律，使用預設名稱因為每個人複製的次序可能不同，結果每個人的程式碼就會各自不同了！而且順便將冗長的 Label 簡化為一個字母 C，寫程式時就可以簡短很多，也好閱讀。那麼可不可以直接使用數字 1~9 為名稱呢？不行的！命名規則規定物件名稱不能以阿拉伯數字開始，用中文字，如「一、二、三…」反而是沒問題的！

接下來請再加入一個標籤(名稱預設應該是 Label1)作為勝負的訊息，以及一個重玩的按鍵(Button1)，當然前面標籤中的"O"只是用來檢視文字位置的，你可以在此刪除它們，刪除時記得可以全選九個標籤一起將 Text 屬性刪掉就可以了！當你同時選取多個物件時，屬性視窗顯示的是它們的「共同屬性」，各自不同的屬性(如位置)就不會出現。最終完成的設計畫面如下：


**10-2 開始與重玩的程式**

本範例中「開始」與「重玩」的意義其實是相同的，就是將所有的遊戲過程內容清除，在此就是將九宮格裡的圈叉與「訊息」清除。要定義「開始」的狀態應該將程式碼寫在 Form_Load 事件，但是我們也讓 Button1.Click 共用就不必重複寫一樣的程式了。比較簡單的想法是可以一一替每個標籤寫清除文字的程式，如 C1.Text=""，在此介紹一個比較專業的語法，程式碼如下：

```
'開始與重玩時清理棋盤
Private Sub Form1_Load(sender As Object, e As EventArgs) Handles MyBase.Load, Button1.Click
    For Each c In Me.Controls '表單上的每一個控制項
        If TypeOf (c) Is Label Then '如果是標籤物件
            c.text = "" '清除文字內容
        End If
    Next
End Sub
```

For Each 如英文字面意義就是「每一個」的意思，c 是一個任意物件的代名詞，

Me.Controls 表示表單上的所有控制項，我們的目標是將所有「標籤」(Label)的字都清掉，不包括 Button1，所以加入一個條件：If TypeOf(c) Is Label，其中 TypeOf 函數可以提取物件的資料型態名稱，是標籤(Label)的話就清除其中的文字(Text)。

想想看，此例中只有十個標籤文字要清除，一一寫程式也不過十行，但如果是一百個物件呢？那時這個技巧就很重要了！執行程式看看，連「訊息」兩個字也會不見了！

### 10-3 圈叉的輪替

「下棋」在此的意義就是使用者會點選九宮格之一，然後應該有文字"O"或"X"出現在格子內，用到的事件副程序就是標籤的 Click 事件了，請先雙擊 C1 產生一個副程序框架，寫入 C1.Text="X"，是不是一點就有字出現了？但是重點在"O"與"X"必須「交替」出現，所以必須設計一個變數，讓程式記得現在應該出現"O"或"X"，程式碼可修改如下：

```
'下棋動作控制程式
Dim T As Boolean
Private Sub C1_Click(sender As Object, e As EventArgs) Handles C1.Click
    If T = True Then
        C1.Text = "X" '畫叉
    Else
        C1.Text = "O" '畫圈
    End If
    T = Not (T) '切換下棋序
End Sub
```

T 是一個 Boolean(布林變數)，它的值只有 True 與 False 兩種，非「是」即「否」，預設值是 False！當使用者點 C1 時，程式依據 T 的狀態，如果是 True 就寫"X"，否則(就是 False)寫"O"，記得寫完一個字就必須切換 T 的狀態，Not 是專用於布林變數的函數，使 True 或 False 相反的意思，所以玩家再點一次 C1 就會變成另一種記號了！試一下程式，你已經可以連續點選 C1 產生圈叉交替的狀況了。

### 10-4 共用事件副程序

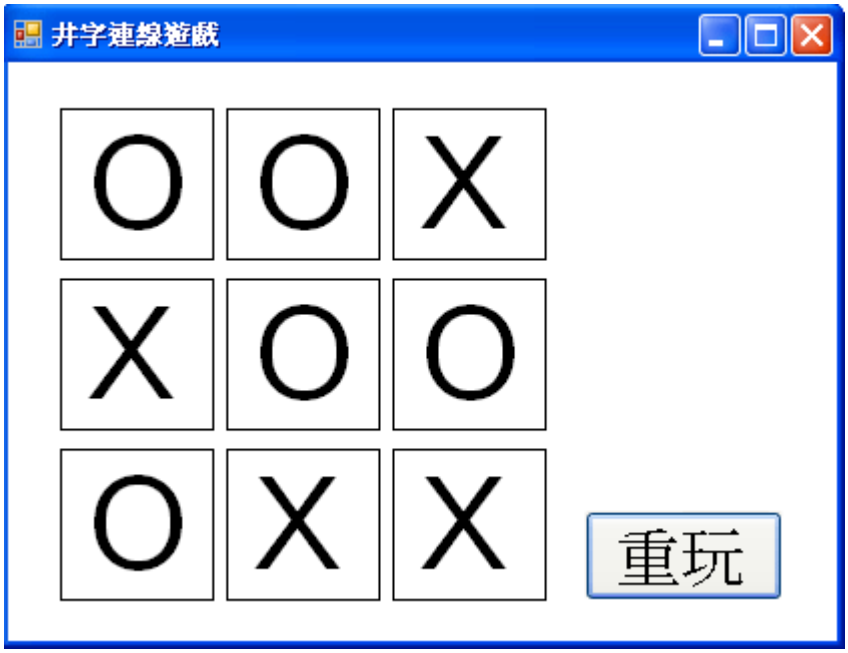
實際下棋時應該不可以在已經下棋的位置重複下棋，而且九個空格應該都要有一樣的功能。要一一製作 C1 到 C9 的九個 Click 事件副程序不是不行，但是我們可以採用事件副程序的「共用」機制，讓 C1~C9 分享同樣的事件副程序，就可以大幅減少程式碼，也減少出錯的機會，修改後的 C1_Click 事件副程序如下：

```
Private Sub C1_Click(sender As Object, e As EventArgs) Handles C1.Click, C2.Click, _
    C3.Click, C4.Click, C5.Click, C6.Click, C7.Click, C8.Click, C9.Click
        If sender.Text = "" Then '棋格為空的
            If T = True Then
                sender.Text = "X" '畫叉
            Else
                sender.Text = "O" '畫圈
            End If
            T = Not (T) '切換下棋序
        End If
End Sub
```

首先我們在事件副程序標題的 Handles 之後加入了 C2.Click~C9.Click 等事件，就是讓 C1~C9 的 Click 事件都共用這段程式碼，程式碼當然變得很長，我們可以使用「接續前行」的語法切斷它以方便檢視，你也可以不必切開，就讓它很長沒關係的。

至於 sender 這個參數的用法在我的小算盤單元中介紹過，碰到上面的狀況 C1~C9 共用一個副程序時，sender 就變成一個代名詞了！C1~C9 之中誰被點到誰就是 sender，程式就去處理誰的狀況，沒有這個機制，就真的要寫九個幾乎完全一樣的副程序了！

回到程式碼的說明，首先要檢查被點的格子是否有文字？沒有文字(sender.Text="")才能繼續下棋的動作。同時也必須用 sender 這個代名詞取代原來的 C1，這樣程式就可以應用到 C1~C9 任何一個事件觸發者了！試試看！現在可以交替在九個格子裡填"O"與"X"了！而且絕對遵守交通規則，填過的就一定改不了，交替次序也不會錯亂了！




## 10-5 連線檢查

接著要在每次新下一步棋時檢查三橫三豎加上兩對角線，共八種可能方向上有沒有出現連線完成的狀況？為了程式結構簡潔起見，在此將檢查連線的程式獨立為一個稱為 chkWin 的 Function 副程序，程式碼如下：

```
'檢查連線的自訂副程序
Function chkWin() As String
    If C1.Text + C2.Text + C3.Text = "OOO" Then Return "O" '圈圈連線(橫一)
    If C4.Text + C5.Text + C6.Text = "OOO" Then Return "O" '圈圈連線(橫二)
    If C7.Text + C8.Text + C9.Text = "OOO" Then Return "O" '圈圈連線(橫三)
    If C1.Text + C4.Text + C7.Text = "OOO" Then Return "O" '圈圈連線(縱一)
    If C2.Text + C5.Text + C8.Text = "OOO" Then Return "O" '圈圈連線(縱二)
    If C3.Text + C6.Text + C9.Text = "OOO" Then Return "O" '圈圈連線(縱三)
    If C1.Text + C5.Text + C9.Text = "OOO" Then Return "O" '圈圈連線(斜一)
    If C3.Text + C5.Text + C7.Text = "OOO" Then Return "O" '圈圈連線(斜二)
    If C1.Text + C2.Text + C3.Text = "XXX" Then Return "X" '叉叉連線(橫一)
```

```
    If C4.Text + C5.Text + C6.Text = "XXX" Then Return "X" '叉叉連線(橫二)
    If C7.Text + C8.Text + C9.Text = "XXX" Then Return "X" '叉叉連線(橫三)
    If C1.Text + C4.Text + C7.Text = "XXX" Then Return "X" '叉叉連線(縱一)
    If C2.Text + C5.Text + C8.Text = "XXX" Then Return "X" '叉叉連線(縱二)
    If C3.Text + C6.Text + C9.Text = "XXX" Then Return "X" '叉叉連線(縱三)
    If C1.Text + C5.Text + C9.Text = "XXX" Then Return "X" '叉叉連線(斜一)
    If C3.Text + C5.Text + C7.Text = "XXX" Then Return "X" '叉叉連線(斜二)
    Return "" '圈叉都未連線
End Function
```

這個 Function 是我們自己手動建立的自訂副程序，還記得我們在秀圖軟體的單元中也寫過自訂副程序嗎？那時我們是使用「Sub」為名而不是「Function」，差別是當我們呼叫兩者時，Sub 是沒有回傳「答案」的！動作做完就結束；相對的，Function 副程序必須回傳(Return)一個「答案」！譬如上面的 chkWin 副程序我們希望它計算的結果是「誰」(X 或 O)連線成功？沒人連線成功就回傳空字串，這也算是個答案，我們的程式就據此來決定後續的動作！

chkWin 裡面共有 16 組的 If 判斷式，分別代表檢查三橫三豎及兩個對角斜線，共八個可能連成一線的狀況，加上圈與叉兩種圖案的分別就是 16 個式子了。每個判斷式將三個格子的 Text 相加之後檢查是不是三個相同的字，發現連線立即回報(Return)答案，"O"連線就回答"O"；"X"連線就回答"X"。

有任何連線出現時表示勝負已定，後續的程式碼就不必繼續執行了！如果 16 種連線狀況都沒發生，最後就回傳一個空字串(Return "")。所以 Return 指令就是 Function 與 Sub 副程序最重要的差別了！每個 Function 副程序都必須結束於 Return 指令，但是可以有很多個結束(Retuen)的出口。

寫到這裡其實還沒叫用 chkWin，本節的努力還沒看到效果。我們必須在每次下棋動作完成時都呼叫 chkWin，如果有人連線成功就在 Label1 中顯示誰贏了！如果回傳值是空字串，就繼續等待後續下棋的動作。下棋事件副程序修改如下：

```
Private Sub C1_Click(sender As Object, e As EventArgs) Handles C1.Click, C2.Click, _
    C3.Click, C4.Click, C5.Click, C6.Click, C7.Click, C8.Click, C9.Click
        If sender.Text = "" Then '棋格為空的
            If T = True Then
                sender.Text = "X" '畫叉
            Else
                sender.Text = "O" '畫圈
            End If
            T = Not (T) '切換下棋序
            If chkWin() <> "" Then '檢視連線勝負
                Label1.Text = chkWin() + " 贏了！" '顯示勝負訊息
            End If
        End If
End Sub
```

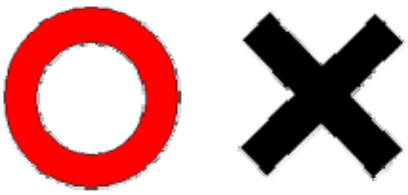
如上程式碼，chkWin 本身有回傳值(字串)，所以可以當一個有內容的字串來使用，如果不是空字串就是有人連線成功了，將這個圈或叉的回傳值(chkWin)加上"贏了！"就是誰獲勝的訊息了！可能的勝負畫面如下：


程式發展至此還有一個小漏洞，就是勝負已定之後空白處仍然可以繼續選圈叉，這會讓旁觀者搞糊塗不知道誰贏了？當然必須禁止！方法很簡單，我們的勝負訊息 Label1 可以做個指標，如果上面有字時就是勝負已定，是不准任何人下棋的！除非按下重玩鍵清除此訊息才可以繼續玩。問題解決了，在 C1_Click 副程序的程式碼外圈加一個條件即可：

```
If Label1.Text = "" Then '勝負未定
   If sender.Text = "" Then '棋格為空的
         …
         …
   End If
End If
```

## 10-6 使用圖案美化介面

前面的程式介紹了遊戲控制的程式，但是使用文字模式的 OX 代表圈叉總是不夠美麗，遊戲嘛！總是應該美美的。我們可以使用真的圈叉圖案來取代製作介面，首先可以自書附光碟找到或自己製作兩個長寬接為 100 點的圖案如下：



將它們加入為資源檔案，程序請參考螢幕保護程式單元，結果如下：


接著將表單介面上的棋格調整為可以容納這兩個圖案的大小(100x100 點)。

## 10-7 使用 Tag 取代 Text

程式碼部分需要的變動主要是原先我們以棋格內的 Text 屬性來判斷連線與否，如果稍後「下棋」時不是改變 Text 而是改變圖案，那麼整個邏輯都不對了！如果堅持繼續寫 Text="X"或"O"，預設狀況下是會顯示在圖案上方擾亂視線的。此時可以使用一個多數物件都有的 Tag 屬性來記錄狀態，也就是之前寫在 Text 屬性的"X"或"O"可以寫在 Tag 屬性中。這個 Tag 屬性可以記錄一些物件的資訊，但是不會顯示於外觀，很方便作為物件資訊的「註記」。

因此我們先將 chkWin 副程序內的 Text 都改成 Tag：

```
'檢查連線的自訂副程序
Function chkWin() As String
    If C1.Tag + C2.Tag + C3.Tag = "OOO" Then Return "O" '圈圈連線(橫一)
    If C4.Tag + C5.Tag + C6.Tag = "OOO" Then Return "O" '圈圈連線(橫二)
    If C7.Tag + C8.Tag + C9.Tag = "OOO" Then Return "O" '圈圈連線(橫三)
    If C1.Tag + C4.Tag + C7.Tag = "OOO" Then Return "O" '圈圈連線(縱一)
    If C2.Tag + C5.Tag + C8.Tag = "OOO" Then Return "O" '圈圈連線(縱二)
    If C3.Tag + C6.Tag + C9.Tag = "OOO" Then Return "O" '圈圈連線(縱三)
    If C1.Tag + C5.Tag + C9.Tag = "OOO" Then Return "O" '圈圈連線(斜一)
    If C3.Tag + C5.Tag + C7.Tag = "OOO" Then Return "O" '圈圈連線(斜二)
    If C1.Tag + C2.Tag + C3.Tag = "XXX" Then Return "X" '叉叉連線(橫一)
    If C4.Tag + C5.Tag + C6.Tag = "XXX" Then Return "X" '叉叉連線(橫二)
    If C7.Tag + C8.Tag + C9.Tag = "XXX" Then Return "X" '叉叉連線(橫三)
    If C1.Tag + C4.Tag + C7.Tag = "XXX" Then Return "X" '叉叉連線(縱一)
    If C2.Tag + C5.Tag + C8.Tag = "XXX" Then Return "X" '叉叉連線(縱二)
    If C3.Tag + C6.Tag + C9.Tag = "XXX" Then Return "X" '叉叉連線(縱三)
    If C1.Tag + C5.Tag + C9.Tag = "XXX" Then Return "X" '叉叉連線(斜一)
    If C3.Tag + C5.Tag + C7.Tag = "XXX" Then Return "X" '叉叉連線(斜二)
    Return "" '圈叉都未連線
End Function
```

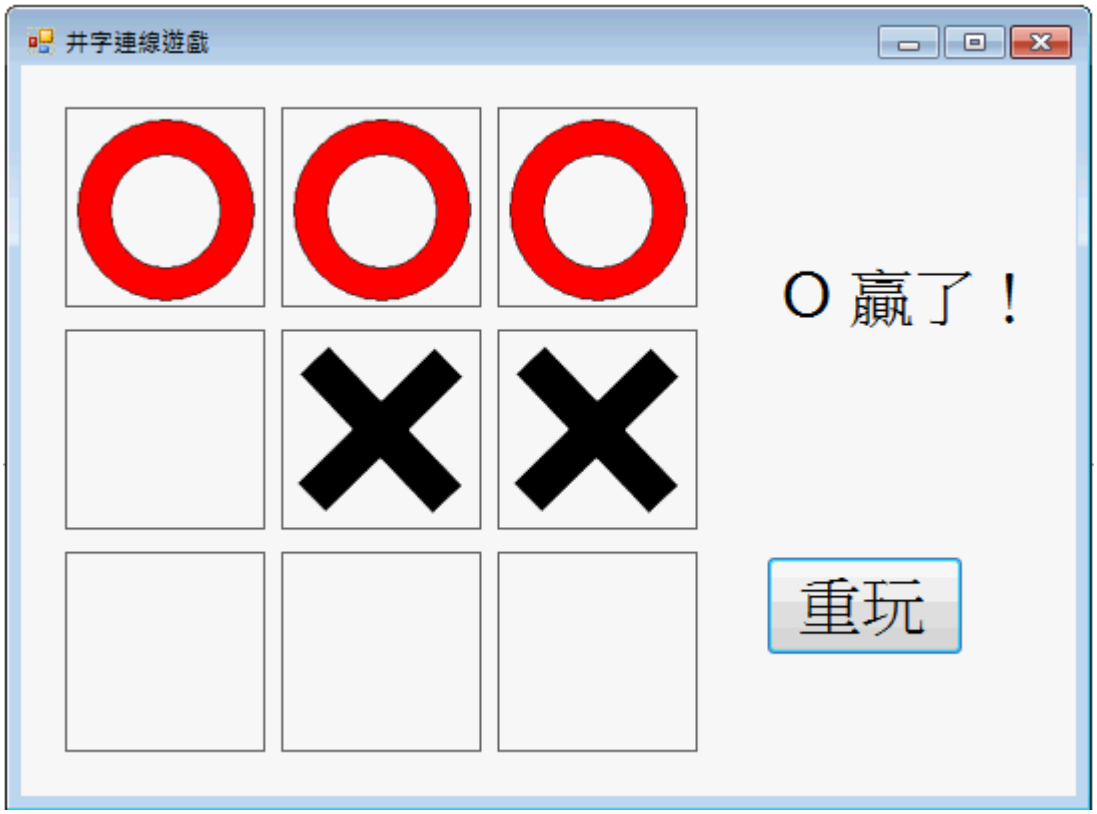
接著將開始與重玩按鍵的副程序修改如下：

```
'開始與重玩時清理棋盤
Private Sub Form1_Load(sender As Object, e As EventArgs) Handles MyBase.Load, Button1.Click
    For Each c In Me.Controls '表單上的每一個控制項
        If TypeOf (c) Is Label Then '如果是標籤物件
            c.text = "" '清除文字內容
            c.Tag = "" '清除Tag
            c.Image = Nothing '清除影像
        End If
    Next
End Sub
```

就是將 Tag 以及可能已經貼上的圈或叉影像(Image)也都清除掉的意思。最後是將下棋的程式修改如下：

```
Private Sub C1_Click(sender As Object, e As EventArgs) Handles C1.Click, C2.Click, _
    C3.Click, C4.Click, C5.Click, C6.Click, C7.Click, C8.Click, C9.Click
    If Label1.Text = "" Then '勝負未定
        If sender.Text = "" Then '棋格為空的
            If T = True Then
                sender.Tag = "X" '註記畫叉
                sender.Image = My.Resources.X '貼上叉叉影像
            Else
                sender.Tag = "O" '註記畫圈
                sender.Image = My.Resources.O '貼上圈圈影像
            End If
            T = Not (T) '切換下棋序
            If chkWin() <> "" Then '檢視連線勝負
                Label1.Text = chkWin() + " 贏了！" '顯示勝負訊息
            End If
        End If
    End If
End Sub
```

執行畫面概略如下：




**10-8 進階挑戰**

一、如何在 OX 連線出現勝負時顯示一個慶祝的小動畫？
提示：先用 PictureBox 載入動畫並隱藏，有人獲勝時讓它出現(Visible=True)。

二、如何顯示目前輪到誰下？
提示：利用 Label1 顯示，T=True 時顯示輪到"X"…

三、如何凸顯出某個三連線已經產生的影像變化？
提示：可以讓連線的棋格邊框變粗、變顏色或使用不同的圖案。

**專案設計頁面與程式碼**

**專案一 Form1：**


```
Public Class Form1
    '下棋動作控制程式
    Dim T As Boolean '下棋次序切換變數
    Private Sub C1_Click(sender As Object, e As EventArgs) Handles C1.Click, C2.Click, _
        C3.Click, C4.Click, C5.Click, C6.Click, C7.Click, C8.Click, C9.Click
        If Label1.Text = "" Then '勝負未定
            If sender.Tag = "" Then '棋格為空的
                If T = True Then
                    sender.Tag = "X" '註記畫叉
                    sender.Image = My.Resources.X '貼上叉叉影像
                Else
                    sender.Tag = "O" '註記畫圈
                    sender.Image = My.Resources.O '貼上圈圈影像
                End If
                T = Not (T) '切換下棋序
                If chkWin() <> "" Then '檢視連線勝負
                    Label1.Text = chkWin() + " 贏了！" '顯示勝負訊息
                End If
            End If
        End If
    End Sub
    '開始與重玩時清理棋盤
    Private Sub Form1_Load(sender As Object, e As EventArgs) Handles MyBase.Load, Button1.Click
        For Each c In Me.Controls '表單上的每一個控制項
            If TypeOf (c) Is Label Then '如果是標籤物件
                c.text = "" '清除文字內容
                c.Tag = "" '清除Tag
                c.Image = Nothing '清除影像
            End If
```

```
        Next
        Dim bmp As New Bitmap(My.Resources.X)
        Label1.Tag = bmp
        Label1.Image = Label1.Tag
    End Sub
    '檢查連線的自訂副程序
    Function chkWin() As String
        If C1.Tag + C2.Tag + C3.Tag = "OOO" Then Return "O" '圈圈連線(橫一)
        If C4.Tag + C5.Tag + C6.Tag = "OOO" Then Return "O" '圈圈連線(橫二)
        If C7.Tag + C8.Tag + C9.Tag = "OOO" Then Return "O" '圈圈連線(橫三)
        If C1.Tag + C4.Tag + C7.Tag = "OOO" Then Return "O" '圈圈連線(縱一)
        If C2.Tag + C5.Tag + C8.Tag = "OOO" Then Return "O" '圈圈連線(縱二)
        If C3.Tag + C6.Tag + C9.Tag = "OOO" Then Return "O" '圈圈連線(縱三)
        If C1.Tag + C5.Tag + C9.Tag = "OOO" Then Return "O" '圈圈連線(斜一)
        If C3.Tag + C5.Tag + C7.Tag = "OOO" Then Return "O" '圈圈連線(斜二)
        If C1.Tag + C2.Tag + C3.Tag = "XXX" Then Return "X" '叉叉連線(橫一)
        If C4.Tag + C5.Tag + C6.Tag = "XXX" Then Return "X" '叉叉連線(橫二)
        If C7.Tag + C8.Tag + C9.Tag = "XXX" Then Return "X" '叉叉連線(橫三)
        If C1.Tag + C4.Tag + C7.Tag = "XXX" Then Return "X" '叉叉連線(縱一)
        If C2.Tag + C5.Tag + C8.Tag = "XXX" Then Return "X" '叉叉連線(縱二)
        If C3.Tag + C6.Tag + C9.Tag = "XXX" Then Return "X" '叉叉連線(縱三)
        If C1.Tag + C5.Tag + C9.Tag = "XXX" Then Return "X" '叉叉連線(斜一)
        If C3.Tag + C5.Tag + C7.Tag = "XXX" Then Return "X" '叉叉連線(斜二)
        Return "" '圈叉都未連線
    End Function
End Class
```

## 課後閱讀

### 一、談談 Function 與 Sub

學到這裡讀者應該已經很習慣 Sub 副程序的使用，最常見的是事件副程序，就是某件事情發生時將要做的事情以程式碼寫進去即可。簡單說 Sub 就是一群指定要做的指令，在秀圖軟體單元中我們也製作過與事件無關的 Sub，就是調整影像大小與位置的 SizeImage 與 CenterImage。我們可以自訂一個 Sub 副程序將我們可能需要重複使用的程式碼集中成一個 Sub，需要時呼叫它即可。

Function 與 Sub 一樣是一個可以隨時呼叫的副程序，差別是他執行完之後必須回傳一個資料，可以是文字、數值或一個任何型態的物件(譬如 Label)。因為必須回傳資料，所以它的標題列後方必須加上一個回傳資料的型態定義，像是本單元的 chkWin 標題是：

```
Function chkWin() As String
```

這表示副程序執行完會回傳一個字串。同時因為回傳值是必要的，寫程式過程中如果軟體發現你沒有完整交代回傳值，就是有些狀況無法執行到 Return 的敘述，就會一直出現波浪底線的警告。請記得 Function 副程序中一旦碰到 Return 敘述就會結束，並返回主程式，也就是後續程式碼是不會繼續執行的！我們可以讓 Function 有很多個出口，但是到 End Function 之前一定要執行到某一個 Return 敘述，如果沒有！就是交代不清，可能有某些狀況會沒有回傳值，這是不允許的！

在 VB 中 Sub 出現的機率遠遠高於 Function，有趣的是：以 C 語言為基礎的多數其他程式語言反而都是只有 Function 副程序這個概念，沒有明確的 Sub 這種程式單元！在 C 語言中如果真的只須作事情而不必回傳值時，就是將回傳值定義為「空」的(在 C# 是 void)！所以 Sub 可說是 VB 特有的一種副程序型態，Function 才是所有程式語言通用的副程序架構。

### 二、談談 Tag 屬性

本範例中用到一個一般書籍中很少介紹的 Tag 屬性，其實讀者可以到屬性視窗看看，多數的物件都有這個屬性。照字面上的意義它就是「標籤」，類似我們在一些東西上面貼的小黃貼紙說明這個東西的身分或用途。如果我們想記錄這個物件的某些資訊，又不需要(或不想要)它外顯出來時，就可以將它當個隱藏的小記事本來用。

它的內容與 Text 一樣可以隨時用程式更改，但實際上它不是只能存文字而已！它的預設資料型態是 Object，就是廣義的物件，如果你要在某物件的 Tag 裡面藏一個數字、Label、PicturBox、資料表或一張影像也都是可以的！譬如下面的程式片段就是將一張影像("X")放到 Label1 的 Tag，以及如何將它取出變成 Label1 物件的影像的程式：

```
Dim bmp As New Bitmap(My.Resources.X)
```

```
Label1.Tag = bmp

Label1.Image = Label1.Tag
```

這個屬性之後在本書的遊戲單元中會繼續用到，通常是以一個數字記錄物件的狀態或分數。但是它可以使用的空間充滿想像，譬如某個物件如果可以變裝，顯示為多種外觀，那麼它的「服裝」影像就可以存在 Tag 裡面。善用這個屬性可以讓你不必定義很多額外的變數，如果某些「東西」是與某物件相關的，就可以直接存在該物件的 Tag 裡面，程式會變得比較容易閱讀，也可以變得較簡短。