
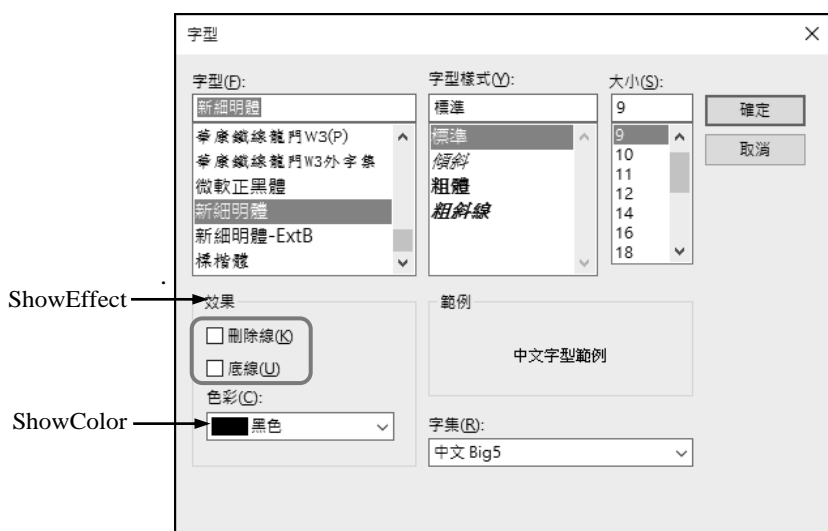


對話方塊與 多表單應用


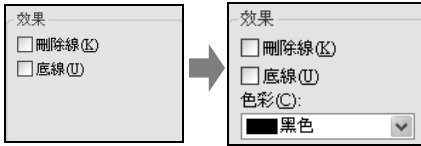
14.1 FontDialog 字型對話方塊控制項

在 Windows 環境下，編輯文字時如下圖提供統一的字型對話方塊，以方便對所選取的文字做各種設定。在 IDE 整合開發環境下是由工具箱提供  FontDialog 字型對話方塊工具來完成。字型對話方塊和 Timer 計時器控制項一樣都是屬於幕後執行的控制項，建立該控制項時位於表單的正下方。程式執行時必須透過 ShowDialog() 方法才能開啟字型對話方塊，接著透過字型對話方塊控制項所提供的相關屬性依需求來做設定，最後將設定的結果傳回給對應的控制項。






下表是 FontDialog 字型對話方塊常用的成員功能說明：

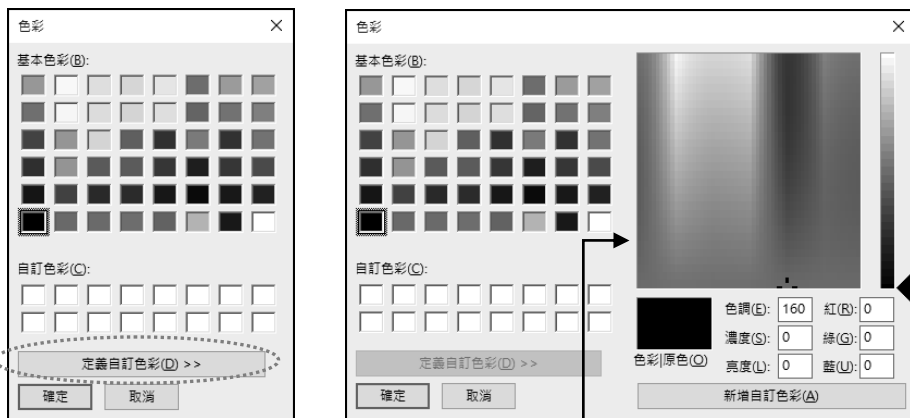
成員名稱	說明
AllowVectorFonts 屬性 (預設值 True)	用來設定是否允許選取向量字型。
AllowVerticalFonts 屬性 (預設值 100)	用來設定是否允許選取垂直字型。
Color 屬性 (預設值 Black)	用來取得在字型對話方塊中所選取的色彩。
FixedPitchOnly 屬性 (預設值 False)	控制是否僅能選取固定字幅的字型。
FontMustExist 屬性 (預設值 False)	如果所選取的字型不存在時，是否回應有錯誤發生。
MaxSize/MinSize 屬性 (預設值 0/0)	用來設定字型對話方塊內『大小(S)』欄內可被選取的最大和最小點數大小(0 表示停用)。
ScriptOnly 屬性 (預設值 False)	設定是否排除 OEM 和符號字元集。
ShowApply 屬性 (預設值 False)	控制是否在字型對話方塊內出現  鈕。
ShowColor 屬性 (預設值 False)	控制是否在字型對話方塊內的「效果」框架內出現字體色彩下拉清單。 
ShowEffects 屬性 (預設值 True)	控制在字型對話方塊內的「效果」框架是否出現加底線、刪除線核取方塊功能。
ShowHelp 屬性 (預設值 False)	控制是否在字型對話方塊內出現  鈕。
Reset 方法	將字型對話方塊控制項內的屬性還原成預設值。



成員名稱	說明
ShowDialog 方法	<p>顯示字型對話方塊。若 fontDialog1 是字型對話方塊控制項的名稱，先將字型對話方塊對話方塊控制項顯示出來，接著再判斷使用者是否按下  鈕？若是，再將字型對話方塊所選取的字型種類指定給 lblShow 標籤控制項的 Font 屬性。其寫法如下：</p> <pre> if (fontDialog1.ShowDialog() == DialogResult.OK) { lblShow.Font = fontDialog1.Font ; } </pre>

14.2 ColorDialog 色彩對話方塊控制項

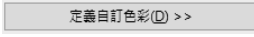
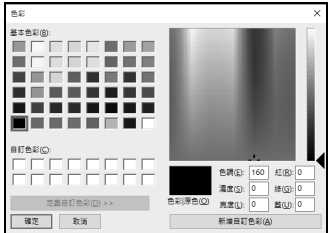
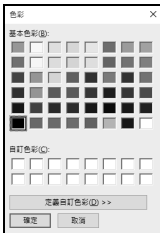


當您希望在表單上出現色彩對話方塊，以方便對所選取的控制項或表單的前景色或背景色做設定，此時可透過  ColorDialog 色彩對話方塊控制項來完成。在設計階段此控制項和字型對話方塊控制項一樣是放在表單的正下方，都是屬於幕後執行的控制項，程式執行時才透過 ShowDialog() 方法開啟如下圖的色彩對話方塊，做完相關屬性設定後，再將設定結果傳回給對應的控制項。



▲ AllowFullOpen 屬性設為 True



下表是 ColorDialog 色彩對話方塊控制項常用成員的功能說明：

成員名稱	說明
AllowFullOpen 屬性 (預設值 True)	設定 ColorDialog 對話方塊內是否顯示如上圖  功能。
AnyColor 屬性 (預設值 False)	控制是否可以選取色彩。
Color 屬性 (預設值 Black)	用來設定或取得對話方塊中使用者所選取的色彩。
FullOpen 屬性 (預設值 False)	<p>ColorDialog 一開始時是否馬上顯示「自訂色彩(C)」區段。 當 AllowFullOpen 為 False 時此屬性無效。</p> <div></div> <p>▲FullOpen=False ▲FullOpen=True</p>
ShowHelp 屬性 (預設值 False)	設定是否顯示  鈕。
SolidColorOnly 屬性 (預設值 False)	設定是否僅能選取純色。True 表示只能選取純色。Color Dialog 並不允許使用者設定自訂的色彩，但允許顯示整套的基本色彩。當將 SolidColorOnly 屬性設為 False 時，它就可顯示由系統上其他 256 色 (含) 彩度以下的色彩組合。此屬性適用於 256 色或較少色彩的系統。
ShowDialog 方法	<p>將指定的 ColorDialog 控制項打開。譬如：先將 txtShow 控制項的前景色傳給 colorDialog1 色彩對話方塊當預設開啟色彩對話方塊的顏色後，接著再開啟 colorDialog1 色彩對話方塊，最後將色彩對話方塊內所選取的色彩再回傳給 txtShow 控制項，其寫法如下：</p> <pre>colorDialog1.color=textBox1.ForeColor; colorDialog1.ShowDialog(); txtShow.ForeColor = colorDialog1.Color ;</pre> <p>出現色彩對話方塊並判斷使用者是否按  鈕，寫法：</p> <pre>if (colorDialog1.ShowDialog() == DialogResult.OK) { // 將所選取的色彩指定給某個控制項的色彩屬性 }</pre>



Reset 方法

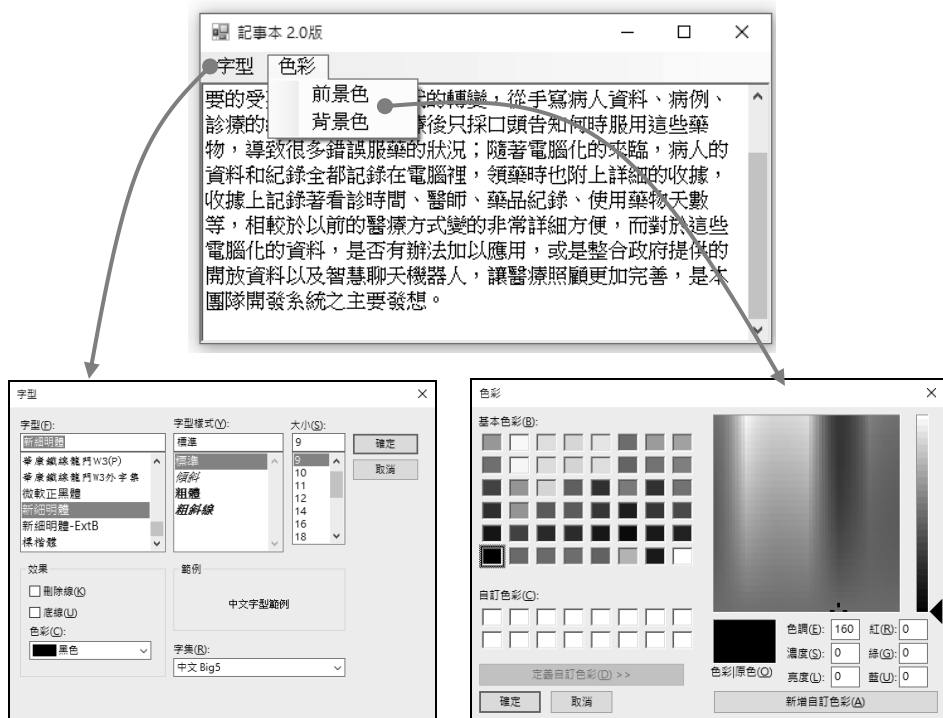
將指定的 colorDialog1 控制項的設定值重新恢復成預設值其寫法如下：
colorDialog1.Reset()

範例：WinFontColorDialog.sln

使用功能表、豐富文字方塊、字型與色彩對話方塊製作一個簡易記事本程式。透過下列三個功能選項做相關的設定。

執行結果

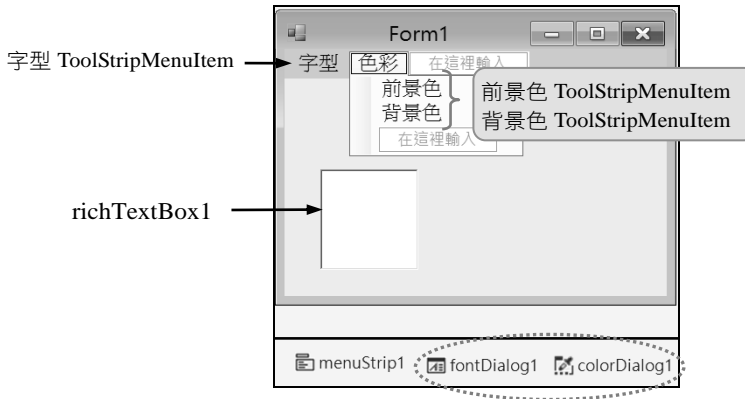
1. 執行功能表的 [字型] 指令出現字型對話方塊，將選取的字型和顏色傳回給豐富文字方塊控制項中選取文字的字型樣式與前景色彩。
2. 執行功能表的 [色彩/前景色] 指令出現色彩對話方塊，將選取的顏色傳回給豐富文字方塊控制項中選取的文字顏色(即前景色)。
3. 執行功能表的 [色彩/背景色] 指令出現色彩對話方塊，將選取的顏色傳回給豐富文字方塊控制項中選取文字的背景色。





操作步驟

Step 01 設計表單的輸出入介面：



Step 02 撰寫程式碼

程式碼 FileName:Form1.cs

```

01 namespace WinFontColorDialog
02 {
03     public partial class Form1 : Form
04     {
05         public Form1()
06         {
07             InitializeComponent();
08         }
09
10         // === 表單載入時執行
11         private void Form1_Load(object sender, EventArgs e)
12         {
13             this.Text = "記事本 2.0 版";
14             // 使 richTextBox1 填滿整個表單
15             richTextBox1.Dock = DockStyle.Fill;
16             // 使 fontDialog1 預設出現色彩下拉式清單
17             fontDialog1.ShowColor = true;
18             // 使 colorDialog1 預設出現自訂色彩區段
19             colorDialog1.FullOpen = true;
20         }
21

```










```



22      // === 執行功能表的 [字型] 指令時執行此事件處理函式
23      private void 字型 ToolStripMenuItem_Click(object sender, EventArgs e)
24      {
25          // === 判斷開啟字型對話方塊時是否按 <確定> 鈕
26          if (fontDialog1.ShowDialog() == DialogResult.OK)
27          {
28              // 將字型對話方塊選取的字型樣式指定給 richTextBox1 中選取文字
29              richTextBox1.SelectionFont = fontDialog1.Font;
30              // === 將選取色彩指定給 richTextBox1 中選取的文字色彩 (即前景色)
31              richTextBox1.SelectionColor = fontDialog1.Color;
32          }
33      }
34
35      // === 執行功能表的 [色彩/前景色] 指令時執行此事件處理函式
36      private void 前景色 ToolStripMenuItem_Click
37          (object sender, EventArgs e)
38      {
39          // 判斷開啟色彩對話方塊時是否按 <確定> 鈕
40          if (colorDialog1.ShowDialog() == DialogResult.OK)
41          {
42              // 將色彩對話方塊選取的色彩指定給 richTextBox1 中選取文字的前景色
43              richTextBox1.SelectionColor = colorDialog1.Color;
44          }
45
46          // === 執行功能表的 [色彩/背景色] 指令時執行此事件處理函式
47          private void 背景色 ToolStripMenuItem_Click
48              (object sender, EventArgs e)
49          {
50              // 判斷開啟色彩對話方塊時是否沒有按 <取消> 鈕
51              if (colorDialog1.ShowDialog() != DialogResult.Cancel )
52              {
53                  // 將色彩對話方塊選取的色彩指定給 richTextBox1 中選取文字的背景色
54                  richTextBox1.SelectionBackColor = colorDialog1.Color;
55              }
56          }
57      }

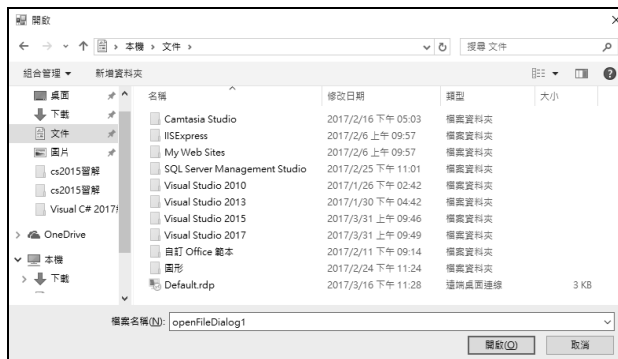
```

**說明**

1. 第 23-33 行：當執行功能表的 [字型] 指令時執行此事件處理函式，先使用 `ShowDialog()` 方法打開字型對話方塊，並判斷使用者是否按下  鈕？若是，則將字型對話方塊所選取的字型與顏色指定給 `richTextBox1` 豐富文字方塊控制項所選取的文字樣式與前景色；若按  鈕則保留原狀。
2. 第 36-44 行：當執行功能表的 [色彩/前景色] 指令時執行此事件處理函式，先使用 `ShowDialog()` 方法打開色彩對話方塊，並判斷使用者是否按下  鈕，若是則將色彩對話方塊所選取的顏色指定給 `richTextBox1` 豐富文字方塊控制項選取文字的前景色；若按  鈕則保留原狀。
3. 第 47-55 行：當執行功能表的 [色彩/背景色] 指令時執行此事件處理函式，先使用 `ShowDialog()` 方法打開色彩對話方塊，並判斷使用者是否未按下  鈕，若未按下  鈕則將色彩對話方塊所選取的顏色指定給 `richTextBox1` 豐富文字方塊控制項所選取文字的背景色；若按  鈕則保留原狀。

14.3 檔案對話方塊

檔案對話方塊(`FileDialog`)是 Windows Form 應用程式設計下所提供許多通用對話方塊之一。它是屬於抽象類別，而且無法直接被執行實體化。`FileDialog` 類別可細分成 `OpenFileDialog` 和 `SaveFileDialog` 兩種類別，由於兩者都是繼承 `FileDialog` 而來，因此兩者的屬性和方法相近。在表單上欲建立開啟或儲存檔案的對話方塊就必須使用  `OpenFileDialog` 開啟檔案對話方塊工具或是  `SaveFileDialog` 儲存檔案對話方塊工具來完成。





至於 `OpenFileDialog` 控制項允許由對話方塊中選取要開啟的檔案、一次選取多個檔案、篩選檔案類別、設定是否唯讀等功能。相反地，當您希望將資料寫入到指定的磁碟機和指定的資料夾和檔案時，可以透過 `SaveFileDialog` 控制項，藉由開啟的存檔對話方塊中選取指定磁碟機、資料夾以及設定欲儲存的檔案名稱。至於 `OpenFileDialog` 和 `SaveFileDialog` 控制項常用的屬性、方法和事件相似，各成員的說明如下表：

成員名稱	說明
AddExtension 屬性	設定在檔案名稱後面是否自動加上副檔名。預設值為 <code>True</code> 表示自動加上。程式中寫法如下： <code>openFileDialog1.AddExtension = true;</code>
CheckFileExists 屬性	在對話方塊將選取檔案傳回前，先檢查檔案是否存在。預設為 <code>True</code> ，表示先檢查。
CheckPathExists 屬性	在對話方塊將選取檔案傳回前，先檢查檔案路徑是否存在。預設值為 <code>True</code> ，表示先檢查。
DefaultExt 屬性	預設的副檔名。當輸入檔名時未加上副檔名，則將此副檔名加在檔名的後面。預設值為空字串。
FileName 屬性	第一次顯示在開啟檔案對話方塊中檔案名稱輸入框內的檔案或是最後一次所選取的檔名。預設值為空字串。程式中寫法如下： <code>openFileDialog1.FileName = "test.rtf";</code>
Filter 屬性	對話方塊中所顯示檔案的篩選條件。預設值為空字串。
FilterIndex 屬性	對話方塊中檔案篩選條件所選取的索引，預設值為 1。
InitialDirectory 屬性	用來設定對話方塊內起始資料夾位置，預設值為空白。 <code>openFileDialog1.InitialDirectory = "D:\\myDir";</code>
MultiSelect 屬性	允許是否可同時選取多個檔案，預設值為 <code>False</code> 。 <code>SaveFileDialog</code> 控制項無此屬性。
ReadOnlyChecked 屬性	若設為 <code>True</code> ，表示檔案以唯讀模式開啟(<code>SaveFileDialog</code> 控制項無此屬性)。預設值為 <code>False</code> 。
ShowReadOnly 屬性	在對話方塊中是否出現唯讀核取方塊(<code>SaveFileDialog</code> 控制項無此屬性)。預設值為 <code>False</code> 。
RestoreDirectory 屬性	設定離開對話方塊時，是否需要還原目前的目錄。



成員名稱	說明
Title 屬性 (預設值為空字串)	用來設定顯示在對話方塊標題列上面的名稱，寫法： <code>openFileDialog1.Title = "選取檔案";</code>
ValidateNames 屬性 (預設值為 True)	是否要確認檔名未包含無效的字元或逸出字元。預設值為 True，表示要確認。
OpenFile 方法	以唯讀方式開啟所選取的檔案，並將檔名以 Stream 物件方式傳回。
Reset 方法	將所有屬性重新設成預設狀態。
ShowDialog 方法	顯示對話方塊。程式寫法如下： <code>openFileDialog1.ShowDialog();</code>
FileOK 事件	在對話方塊中按  、  鈕時觸發的事件。

**範例**：WinOpenSaveDialog.sln

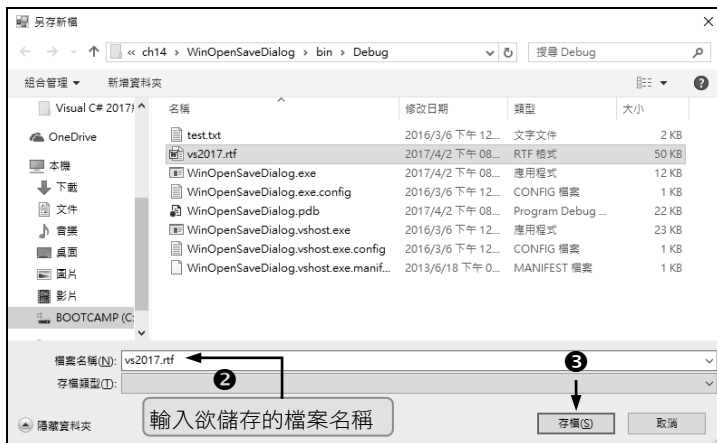
延續 WinFontColorDialog.sln 範例，並在該例新增 OpenFileDialog 及 SaveFileDialog 控制項，使本例擁有可開啟舊檔、儲存檔案之功能。

執行結果

1. 執行功能表的【檔案/開啟舊檔】指令出現開啟舊檔對話方塊，選取欲開啟的檔案之後會將檔案的內容放入豐富文字方塊內。
2. 執行功能表的【檔案/儲存檔案】指令出現另存新檔對話方塊，可將豐富文字方塊的內容寫入到指定路徑下。
3. 執行功能表的【檔案/清除】指令可將豐富文字方塊的所有內容清除。
4. 執行功能表的【檔案/結束】指令可結束程式。

執行【檔案/儲存檔案】
開啟另存新檔對話方塊進行存檔

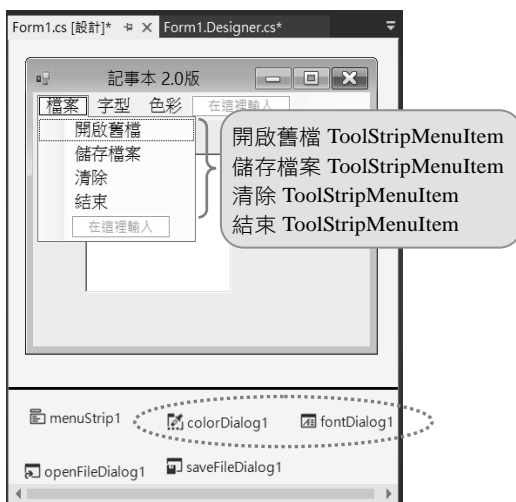




將修改過的檔案存入預設的 C:\cs2017\ch14\WinOpenSaveDialog\bin\Debug 資料夾下，檔案名稱設為「vs2017.rtf」。

操作步驟

- Step 01** 開啟上例 ch14/WinFontColorDialog 資料夾下的「WinFontColorDialog.sln」方案檔。
- Step 02** 先在表單內建立 openFileDialog1、saveFileDialog1 控制項，並在主功能表的「字型」功能表前面新增「檔案」功能表，並在「檔案」功能表下新增「開啟舊檔」「儲存檔案」「清除」「結束」四個子功能表選項，表單的輸出入介面如下圖所示：□



**Step 03** 撰寫程式碼：

延續上例，撰寫如下 WinFontColorDialog.sln 開啟舊檔、儲存檔案、清除、結束等功能選項的 Click 事件處理函式。即在 WinFontColorDialog.sln 插入下列敘述。(完整程式請直接開啟 WinOpenSaveDialog.sln)

程式碼 FileName:Form1.cs

```

    ...
    此部份是 WinFontColorDialog.sln 專案檔程式碼
    ...

01  // === 執行功能表的 [檔案/開啟舊檔] 指令時執行此事件處理函式
02  private void 開啟舊檔 ToolStripMenuItem_Click(object sender, EventArgs e)
03  {
04      // 打開開啟舊檔對話方塊並判斷是否按下 [確定] 鈕
05      if (openFileDialog1.ShowDialog() == DialogResult.OK)
06      {
07          // 使用 richTextBox1 的 LoadFile 方法載入開啟舊檔對話方塊指定的檔案
08          richTextBox1.LoadFile(openFileDialog1.FileName,
                                RichTextBoxStreamType.RichText);
09      }
10  }
11
12  // === 執行功能表的 [檔案/儲存檔案] 指令時執行此事件處理函式
13  private void 儲存檔案 ToolStripMenuItem_Click(object sender, EventArgs e)
14  {
15      // 打開另存新檔對話方塊並判斷是否按 <確定> 鈕
16      if (saveFileDialog1.ShowDialog() == DialogResult.OK)
17      {
18          // 使用 richTextBox1 的 SaveFile 方法將 richTextBox1 內的資料
19          // 存入另存新檔對話方塊指定的檔案內
20          richTextBox1.SaveFile(saveFileDialog1.FileName,
                                RichTextBoxStreamType.RichText);
21      }
22  }
23
24  // === 執行功能表的 [檔案/清除] 指令時執行此事件處理函式
25  private void 清除 ToolStripMenuItem_Click(object sender, EventArgs e)
26  {

```



```

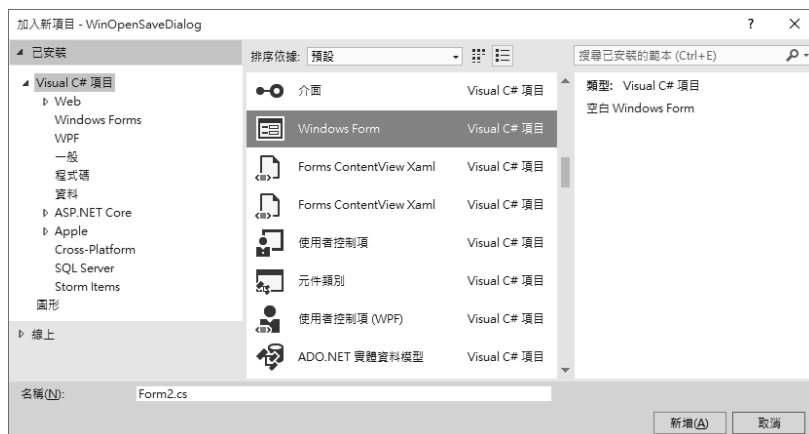
27     richTextBox1.Text = "";
28 }
29
30 // === 執行功能表的 [檔案/結束] 指令時執行此事件處理函式
31 private void 結束ToolStripMenuItem_Click(object sender, EventArgs e)
32 {
33     Application.Exit();
34 }
35
36
37
38
39
40
41
42
43
44
45
46
47
48
49
50
51
52
53
54
55
56
57
58
59
60
61
62
63
64
65
66
67
68
69
70
71
72
73
74
75
76
77
78
79
80
81
82
83
84
85
86
87
88
89
90
91
92
93
94
95
96
97
98
99

```

14.4 多表單開發

前面章節所討論的都是有一個表單的架構下設計出來的程式，一個具有規模的應用程式，其專案所使用表單往往不會只有一個，且多個表單可能會共用一些變數、結構、列舉、方法或是呼叫自定的類別，上述共用的變數、結構、方法及類別…等可定義在獨立的類別檔內，以提供給多個表單呼叫使用。如果要在應用程式內加入類別檔、表單檔(Windows Form)或其他的檔案資源，可以執行功能表的【專案(P)/加入新項目(W)…】開啟下圖「加入新項目」視窗來選擇所要加入的檔案資源。

若專案內有多個表單，常需要在程式執行階段開啟其他的表單。若要開啟其他表單首先必須先建立該表單的物件實體，接著再透過表單類別所提供的方法來對表單進行一些操作，如表單隱藏或顯示…等等。





下面簡例示範如何建立表單物件

```
Form1 f1 = new Form1();           // 建立 f1 表單物件為 Form1 類別  
frmMain f = new frmMain();        // 建立 f 表單物件為 frmMain 類別
```

透過下面表單所提供的方法，可以顯示或隱藏表單

```
f.Show();                          // 顯示 f 表單物件  
f.ShowDialog();                    // 以對話方塊強制回應形式顯示 f 表單物件  
f.Hide();                          // 隱藏 f 表單物件
```

如果欲存取其他表單的成員，如控制項的屬性或類別方法，其做法就是將該成員宣告成 **public** 成員，然後就可以直接呼叫該表單內所有 **public** 成員。接著我們透過下面範例來體驗如何製作擁有兩個表單檔(即*.cs 類別檔)的應用程式。



範例：WinMultiForm.sln

製作擁有多表單的專案。程式執行時開啟 **frmMain** 表單要求使用者輸入本金、利率、幾年後領回的資訊，接著按下 **開啟試算** 鈕即會開啟 **frmCal** 表單讓使用者透過選項按鈕選擇利息的計算方式是採「每年計息」或「每月計息」，計算完成後，會將結果傳回 **frmMain** 表單的 **lblShow** 標籤並顯示出來。使用者也可以按下 **frmMain** 表單的 **使用小算盤** 鈕，開啟 Windows 提供的小算盤應用程式來做試算。

- ① 每年計息公式：本金 $\times(1+\text{年利率})^{\text{年數}}$
- ② 每月計息公式：本金 $\times(1+\text{年利率}/12)^{\text{年數}\times 12}$

執行結果

1. 程式執行時出現左下圖視窗讓使用者輸入本金、年利率、幾年後領回的資料，接著按下 **開啟試算** 鈕由出現的「選擇計息」視窗選擇利息的計算方式是採「每年計息」或「每月計息」，最後按下「選擇計息」視窗的 **×** 鈕之後，如右下圖即返回原視窗並顯示利息計算結果。



表單名稱(Name 屬性)為 frmMain

表單名稱(Name 屬性)為 frmCal

表單名稱(Name 屬性)為 frmMain

2. 若本金、年利率、幾年後領回的資料非數值資料，進行試算時會出現右圖對話方塊，並顯示"請輸入正確的數值資料"。

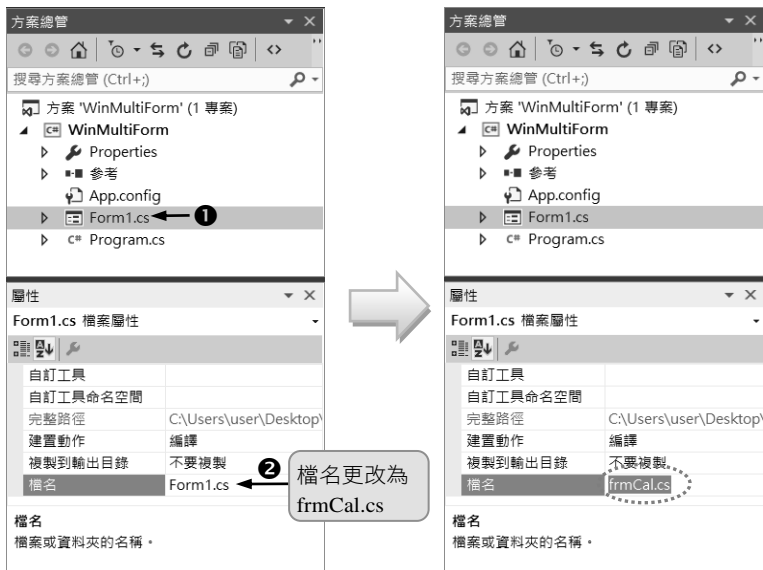
3. 按 **使用小算盤** 鈕即可開啟右圖 Windows 所提供小算盤程式。



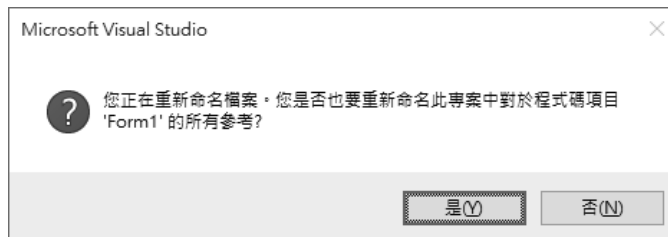
操作步驟

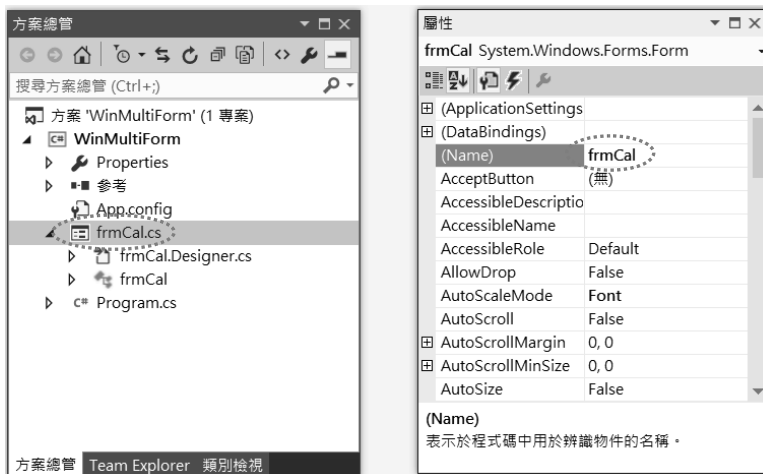
Step 01 建立 frmCal.cs 選擇利息計算方式表單

- ① 本例用到兩個表單，建議將每個表單檔檔名(.cs)與表單物件名稱 (Name 屬性)另取有意義的名稱以茲識別。在下圖先選取「方案總管」視窗的 Form1.cs，接著再透過「屬性」視窗將表單檔名更改為「frmCal.cs」，此時表單物件名稱同時會更名為「frmCal」。

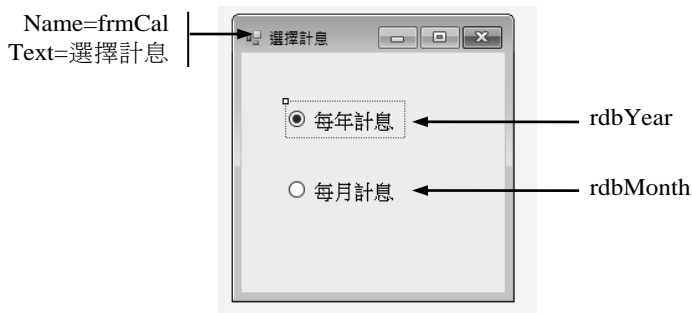


當更名完畢時，出現下圖詢問畫面，請按 **是(Y)** 鈕，將檔名和物件名稱 (Name 屬性)一起更名。





② 請在下圖 frmCal 表單放置兩個 RadioButton 選項按鈕控制項：



③ 請在 frmCal 表單撰寫 public 的 Cal 方法，此方法會依據使用者所選擇的每年計息(radYear)或每月計息(radMonth)之選項按鈕來計算配息方式，並傳回配息後的本利和。程式碼如下：

程式碼 FileName: frmCal.cs

```

01 namespace WinMultiForm
02 {
03     public partial class frmCal : Form
04     {
05         public frmCal()
06         {
07             InitializeComponent();
08         }

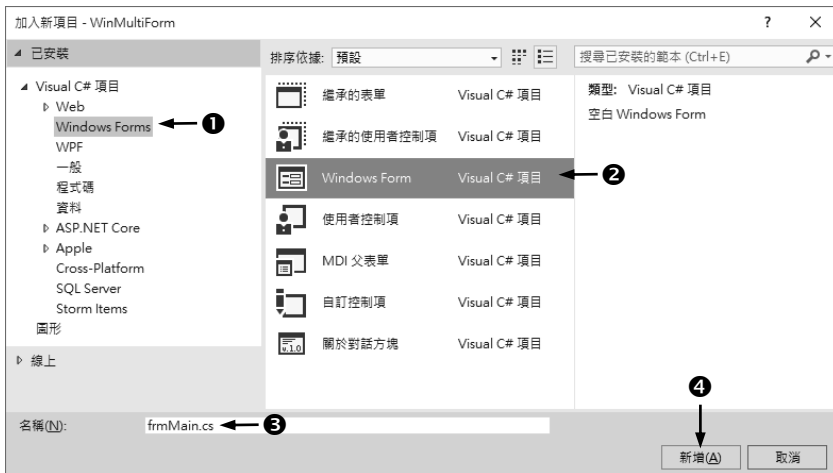
```



```
09
10      //Cal 方法可計算配息方式
11      public int Cal(int vMoney, int vYear, double vRate)
12      {
13          if (rdbYear.Checked)
14          {
15              // 每年計息一次
16              return (int)(vMoney * Math.Pow(1 + vRate, vYear));
17          }
18          else
19          {
20              //每月計息一次
21              return (int)(vMoney * Math.Pow(1 + (vRate) / 12, vYear * 12));
22          }
23      }
24  }
25 }
```

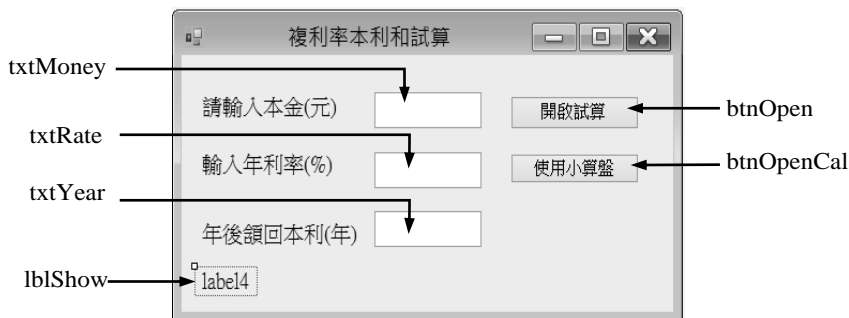
Step 02 建立 frmMain.cs 程式主表單

1. 執行功能表的【專案(P)/加入新項目(W)…】開啟下圖「加入新項目」視窗，然後依下圖操作新增一個 Windows Form 項目(表單檔)，其檔名為「frmMain.cs」。





2. 開啟 frmMain 表單，接著在表單建立下圖指定的控制項。



3. 在 frmMain 表單撰寫程式：

- ① 按 **開啟試算** **btnOpen** 鈕時執行 **btnOpen_Click** 事件處理函式，在此事件處理函式先判斷本金、利率、幾年後領回的資料是否符合數值，接著建立 **f** 物件為 **frmCal** 表單類別並開啟 **f** 表單物件，最後呼叫 **f.Cal()** 方法計算配息的結果並顯示在 **frmMain** 表單的 **lblShow** 標籤上。
- ② 按 **使用小算盤** **btnOpenCal** 鈕時執行 **btnOpenCal_Click** 事件處理函式，在此事件處理函式內撰寫開啟小算盤應用程式的敘述。

frmMain.cs 表單完整程式碼：

程式碼 FileName: frmMain.cs

```

01 namespace WinMultiForm
02 {
03     public partial class frmMain : Form
04     {
05         public frmMain()
06         {
07             InitializeComponent();
08         }
09         // === 表單載入時執行
10         private void frmMain_Load(object sender, EventArgs e)
11         {
12             lblShow.Text = "";
13         }
14         // === 按 <開啟試算> 鈕執行

```



```
15     private void btnOpen_Click(object sender, EventArgs e)
16     {
17         int myMoney = 0, myYear = 0;
18         double myRate = 0;
19         try
20         {
21             myMoney = int.Parse(txtMoney.Text);
22             myYear = int.Parse(txtYear.Text);
23             myRate = double.Parse(txtRate.Text) / 100;
24         }
25         catch (Exception ex)
26         {
27             MessageBox.Show("請輸入正確的數值資料");
28             return;
29         }
30         frmCal f = new frmCal(); //宣告並建立 frmCal 表單類別的 f 物件
31         //使用 ShowDialog() 方法使 f 以強制回應形式顯示表單
32         f.ShowDialog();
33         //呼叫 frmCal 的 Cal 方法以計算配息方式
34         lblShow.Text = myYear.ToString() + " 年後領回本利和：" +
35             f.Cal(myMoney, myYear, myRate).ToString();
36         // === 按 <使用小算盤> 鈕執行
37     private void btnOpenCal_Click(object sender, EventArgs e)
38     {
39         // 開啟小算盤應用程式
40         System.Diagnostics.Process.Start
41             ("C:\\WINDOWS\\system32\\calc.exe");
42     }
43 }
```

說明

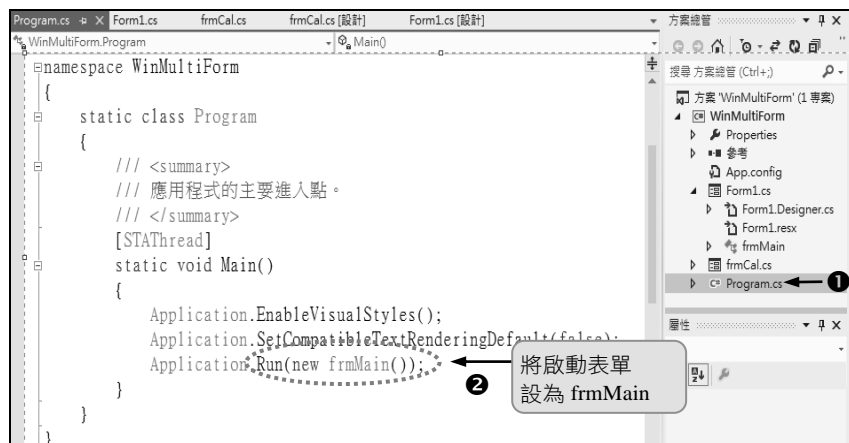
1. 第 19-29 行：使用 try{...}catch{...} 敘述來監控 frmMain 表單的文字方塊輸入的值是否為數值，若不是數值會發生例外，發生例外時會執行第 27-28 行，由出現的對話方塊顯示 "請輸入正確的數值資料"。
2. 第 21 行：使用 int.Parse 方法將 txtMoney 文字方塊的值轉成整數，然後再指定給 myMoney 變數。
3. 第 22,23 行：執行方式同第 21 行。



4. 第 30,32 行：建立 f 為 frmCal 表單類別，接著使用 ShowDialog()方法開啟 f 表單物件。
5. 第 34 行：呼叫 f 表單的 Cal 方法，以便將計算配息的結果顯示在 lblShow 標籤上。

Step 03 設定啟動表單為 frmMain 主表單

當視窗應用程式內有兩個以上的表單，C# 預設會以第一個建立的表單當作程式的啟動表單(本例預設啟動表單是 frmCal.cs)，若要將 frmMain.cs 第二個建立的表單設定為本程式的啟動表單，可以如下圖開啟 Program.cs 檔並修改 Application.Run()方法內欲指定的啟動表單物件即可。在「方案總管」視窗的 Program.cs 檔案上快按滑鼠兩下，開啟 Program.cs 檔，接著將 Application.Run()方法中的參數改為「new frmMain()」就可以了。

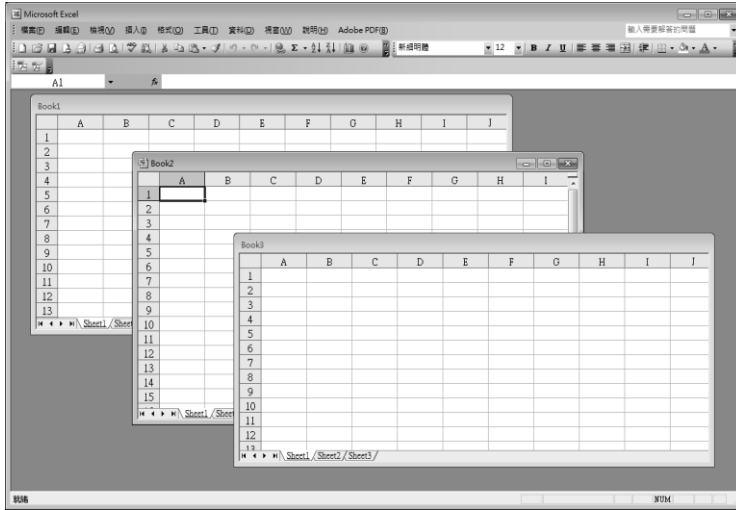


14.5 MDI 多表單開發

上一節學習開發多表單的應用程式，由於一次只能開啟一份表單(即一份文件介面)，此種程式稱為單一文件介面(SDI-Single Document Interface)，SDI 所開啟的表單(或稱文件)可放置在任何位置。在 Windows Form 應用程式中允許您開發多重文件介面 (MDI-Multiple Document Interface) 應用程式，讓您能夠在一個父表單容器內同時開啟多個子表單，子表單皆置於父表單的工作區域內，父表單通常會擁有功能表選單來管理子表單的開啟、關閉、存檔...等動作。例如：Microsoft Excel 就是一個



MDI 多重文件應用程式的例子。在下圖 Microsoft Excel 內同時開啟了 Book1.xls、Book2.xls 及 Book3.xls 三個活頁簿。



若欲建立 MDI 多重文件應用程式，步驟如下：

Step 01 設定可容納子表單的父表單容器

欲將 `frmMain` 表單設為可容納子表單的父表單容器，就必須將 `frmMain` 表單的 `IsMdiContainer` 屬性設為 `true` (`IsMdiContainer` 屬性預設為 `false`)，使該表單成為 MDI 的容器。

Step 02 透過程式設定 `frmBar` 表單的父表單為 `frmMain`

假設目前操作表單為 `frmMain` (目前表單可用 `this` 表示)，先建立 `frmBar` 表單物件 `f`，接著透過 `MdiParent` 屬性指定表單物件 `f` 的父表單為 `this` (即 `frmMain` 表單)，最後使用 `Show` 方法將表單物件 `f` 顯示於 `frmMain` 父表單內。其寫法如下：

```
frmBar f = new frmBar(); // 建立 f 表單物件為 frmBar 表單類別
f.MdiParent = this;      // 設定 f 表單的父表單為 frmMain
f.Show();                // 開啟 f 表單，此時 f 表單會置於 frmMain 內
```



範例：MDIMultiFormEx.sln

製作擁有 MDI 多表單的應用程式。程式執行時開啟 frmMain 表單，執行該表單功能表的 [遊戲種類/拉霸遊戲] 指令即可開啟 frmBar 拉霸遊戲表單；若執行 [遊戲種類/記憶大考驗] 指令即可開啟 frmMemory 記憶大考驗表單；執行 [結束] 則關閉應用程式。

執行結果



操作步驟

Step 01 開啟 MDIMultiFormEx.sln 方案檔練習

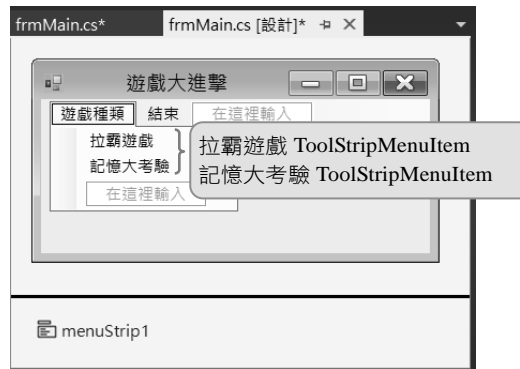
本例專案已建立好 frmBar 拉霸遊戲表單及 frmMemory 記憶大考驗遊戲表單，關於拉霸與記憶體大考驗遊戲的實作方式，可參閱 Visual C# 2017 基礎必修課一書，在範例內也有對兩個遊戲的程式加上註解以方便讀者閱讀。(本例完成品請參閱 MDIMultiFormDemo.sln)

Step 02 建立 frmMain.cs 父表單

1. 執行功能表的【專案(P)/加入新項目(W)...】開啟下圖「加入新項目」視窗，然後依下圖操作新增一個 Windows Form 項目(表單檔)，其檔名為「frmMain.cs」。



2. 開啟 frmMain 表單，接著在表單上建立下圖功能表控制項。



3. 在 frmMain 主表單撰寫程式：

- ① 表單載入時請將目前表單 this 的 IsMdiContainer 屬性設為 true，使目前的表單成為 MDI 的容器。
- ② 在按 【遊戲種類/拉霸遊戲】項目的事件處理函式內建立 f 表單物件屬於 frmBar 拉霸遊戲表單，接著設定 f 表單的父表單為目前表單 this，最後再透過 Show 方法開啟 f 表單。
- ③ 在按 【遊戲種類/記憶大考驗】項目的事件處理函式內建立 f 表單物件屬於 frmMemory 記憶大考驗遊戲表單，接著設定 f 表單的父表單為目前表單 this，最後再透過 Show 方法開啟 f 表單。



frmMain.cs 表單完整程式碼：

程式碼 FileName:frmMain.cs

```

01 namespace MDIMutilFormDemo
02 {
03     public partial class frmMain : Form
04     {
05         public frmMain()
06         {
07             InitializeComponent();
08         }
09
10         // 表單載入時執行此事件處理函式
11         private void frmMain_Load(object sender, EventArgs e)
12         {
13             // 指定目前表單為 MDI 表單的容器
14             this.IsMdiContainer = true;
15         }
16         // 執行功能表的 [遊戲種類/拉霸遊戲] 選項會執行此事件處理函式
17         private void 拉霸遊戲ToolStripMenuItem_Click
18             (object sender, EventArgs e)
19         {
20             frmBar f = new frmBar();
21             f.MdiParent = this; // f 表單的父表單為目前的 frmMain 表單
22             f.Show();
23         }
24         // 執行功能表的 [遊戲種類/記憶大考驗] 選項會執行此事件處理函式
25         private void 記憶大考驗ToolStripMenuItem_Click
26             (object sender, EventArgs e)
27         {
28             frmMemory f = new frmMemory();
29             f.MdiParent = this; // f 表單的父表單為目前的 frmMain 表單
30             f.Show();
31         }
32         // 執行功能表的 [結束] 選項會執行此事件處理函式
33         private void 結束ToolStripMenuItem_Click
34             (object sender, EventArgs e)
35         {
36             Application.Exit();
37         }
38     }
39 }

```



```
35     }
```

```
36 }
```

Step 03 設定啟動表單為 frmMain 父表單

如下圖，本例設定「frmMain」為啟動表單，請先透過「方案總管」視窗開啟 Program.cs 檔，接著將 Application.Run()方法中的參數改為「new frmMain()」。

