Chapter 6

Introduction to Windows Form Application

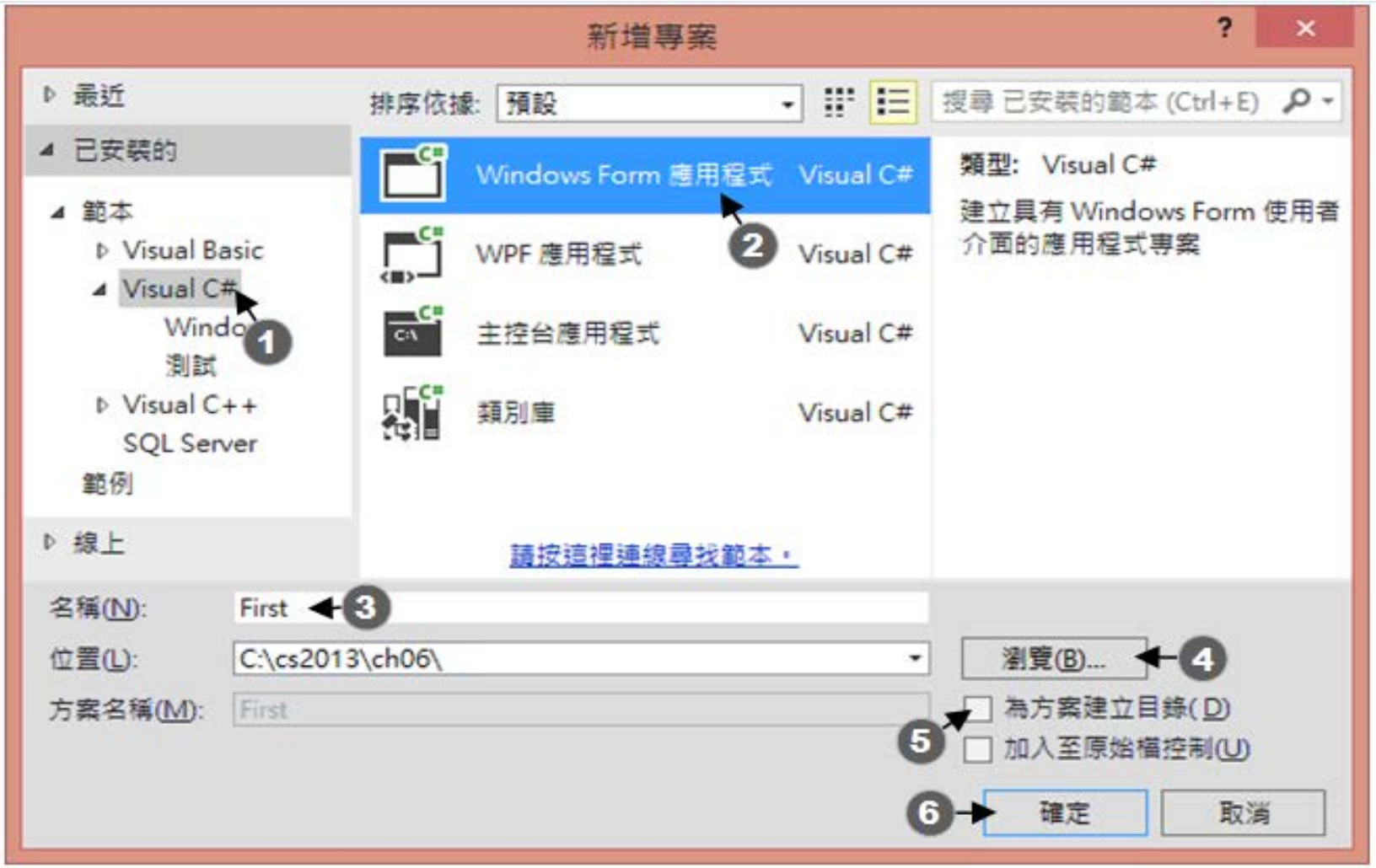
6-1 First Discovered Windows Application

- Former chapters: run under console mode
- Windows application runs under window mode
- Main difference
  ① input and output at I/O interface
  ② WYSIWYG(What You See Is What You Get) user interface maintenance, no need to wait until execution

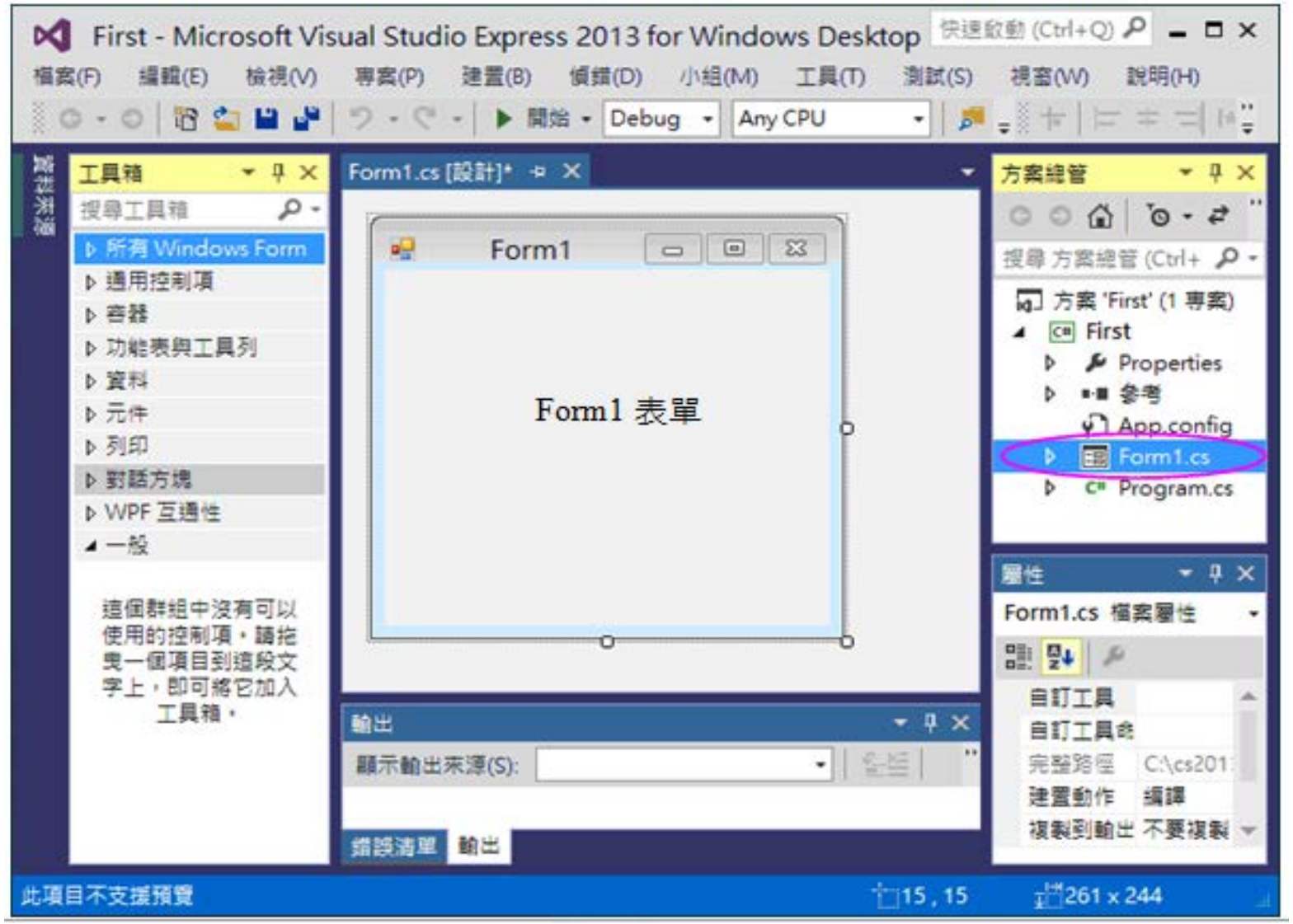
Project

- Use “Form” as a container
- Includes menu, list, text box, command, button, check box, option, etc.
- Use event-driven method to connect source code, “form” and “control items”
- Form is going to be a Windows application as the project is running

1. Open Visual C# IDE



Entering IDE



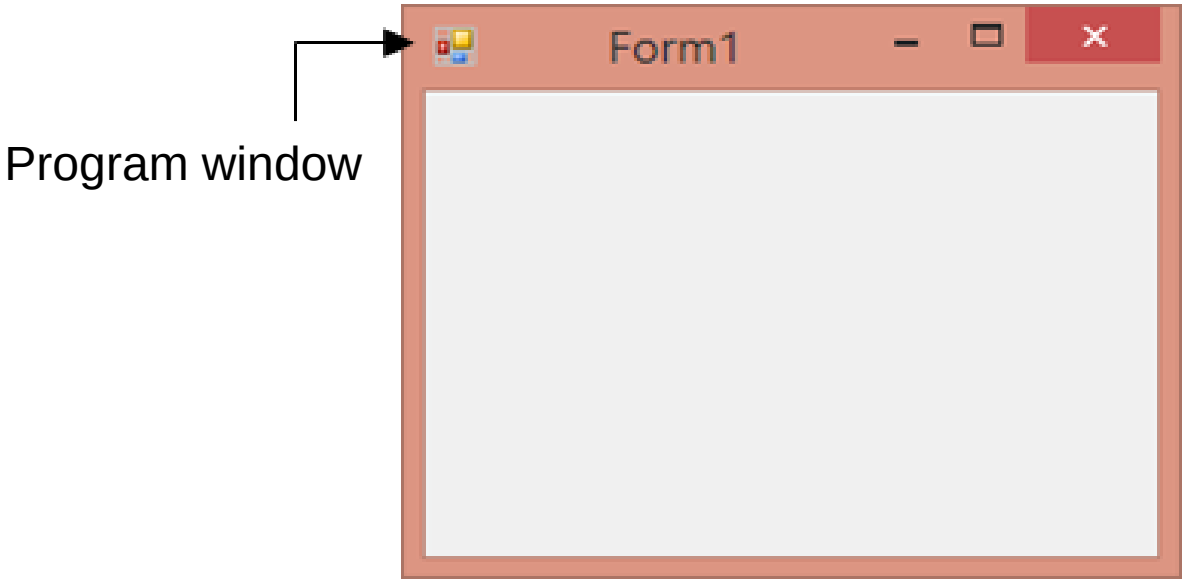
2. Windows Application – Run & Close

1. Run

① menu Debug(D)/Start Debug(S)

② shortcut key F5

Open the program window named Form1



2. Windows Application – Run & Close

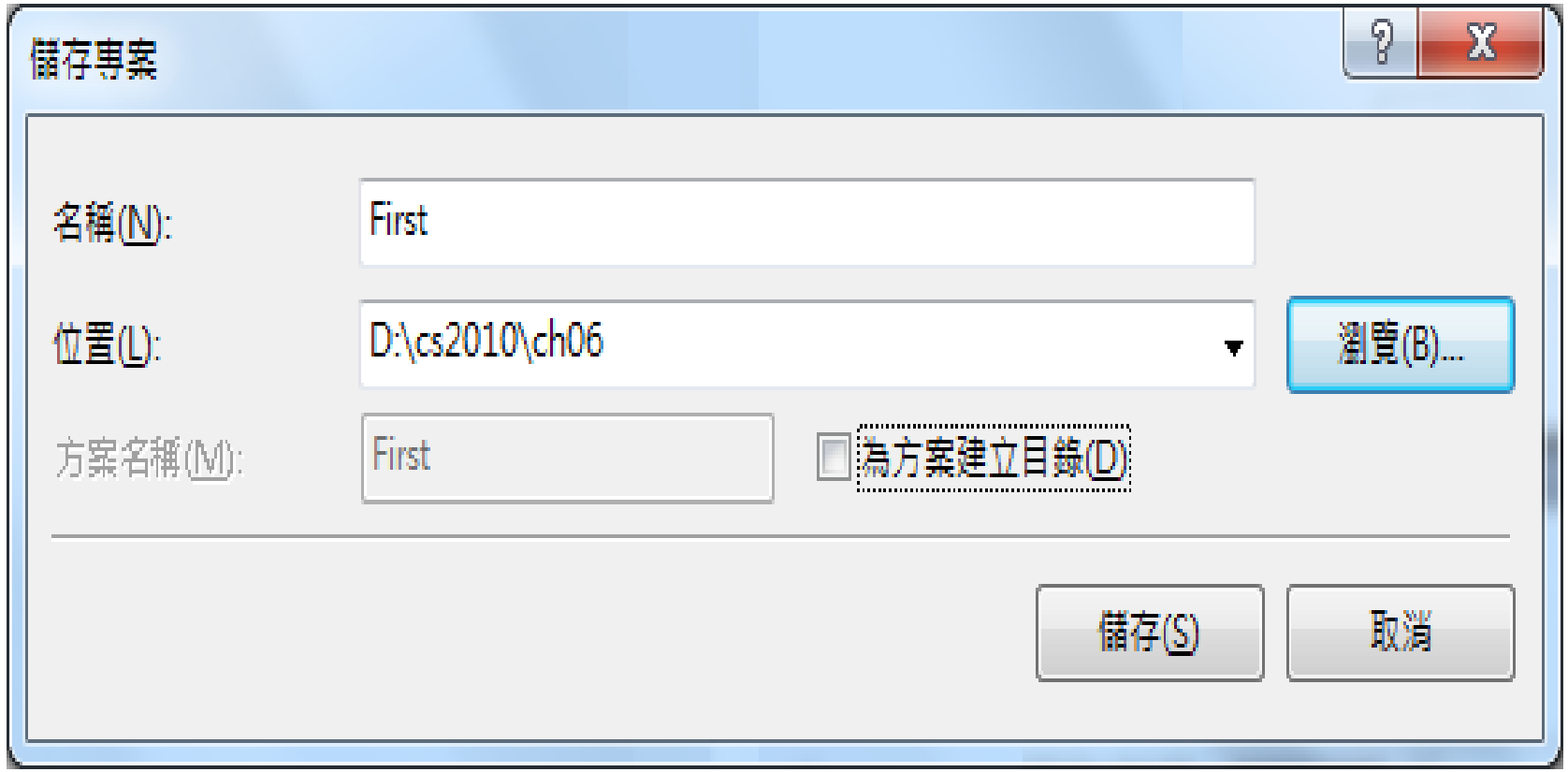
2. Close

① press the close button at the right-top side of window

② menu Debug(D)/Stop debug(E)

terminate running Windows application and back to the IDE

3. Windows Application – Save and Open



3. Windows Application – Save and Open

1. **Save program**

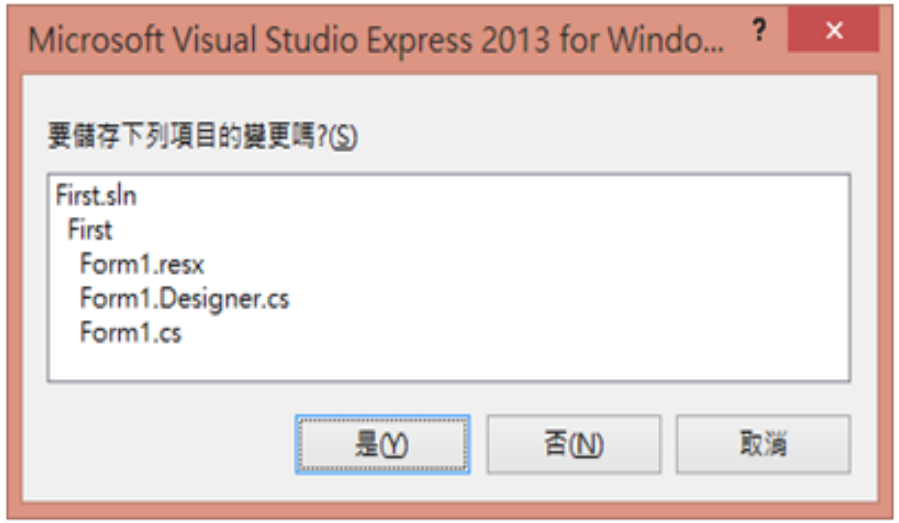
   ① press the “Save all” icon at tool bar 

   ② menu File(F)/Save all(L)

   The related file of First project is stored at assigned directory

2. Exiting IDE

When execute File(F)/Exit(X) to leave IDE, the IDE will question about saving if the project is modified.

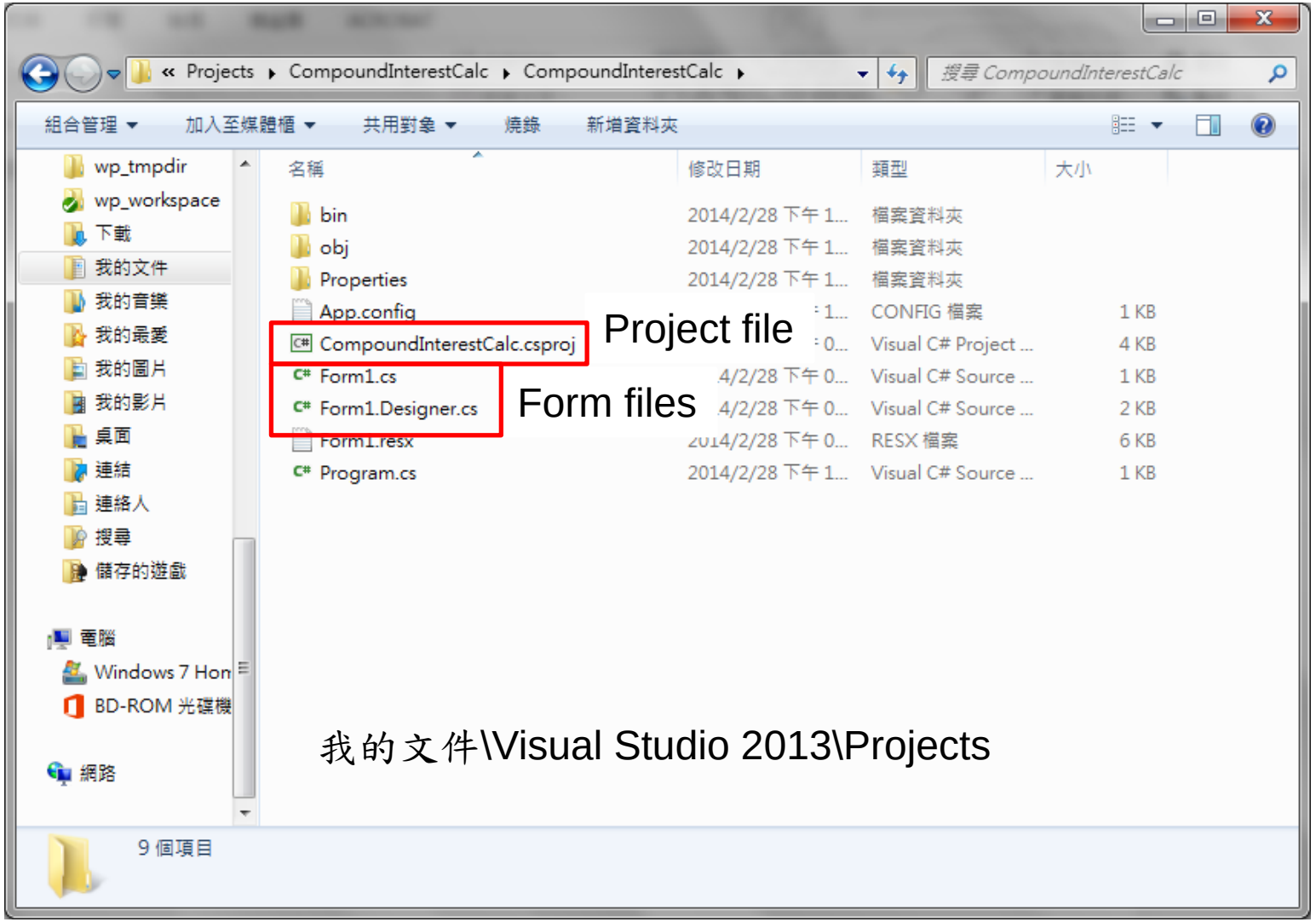


① 是(Y) Replace the original content

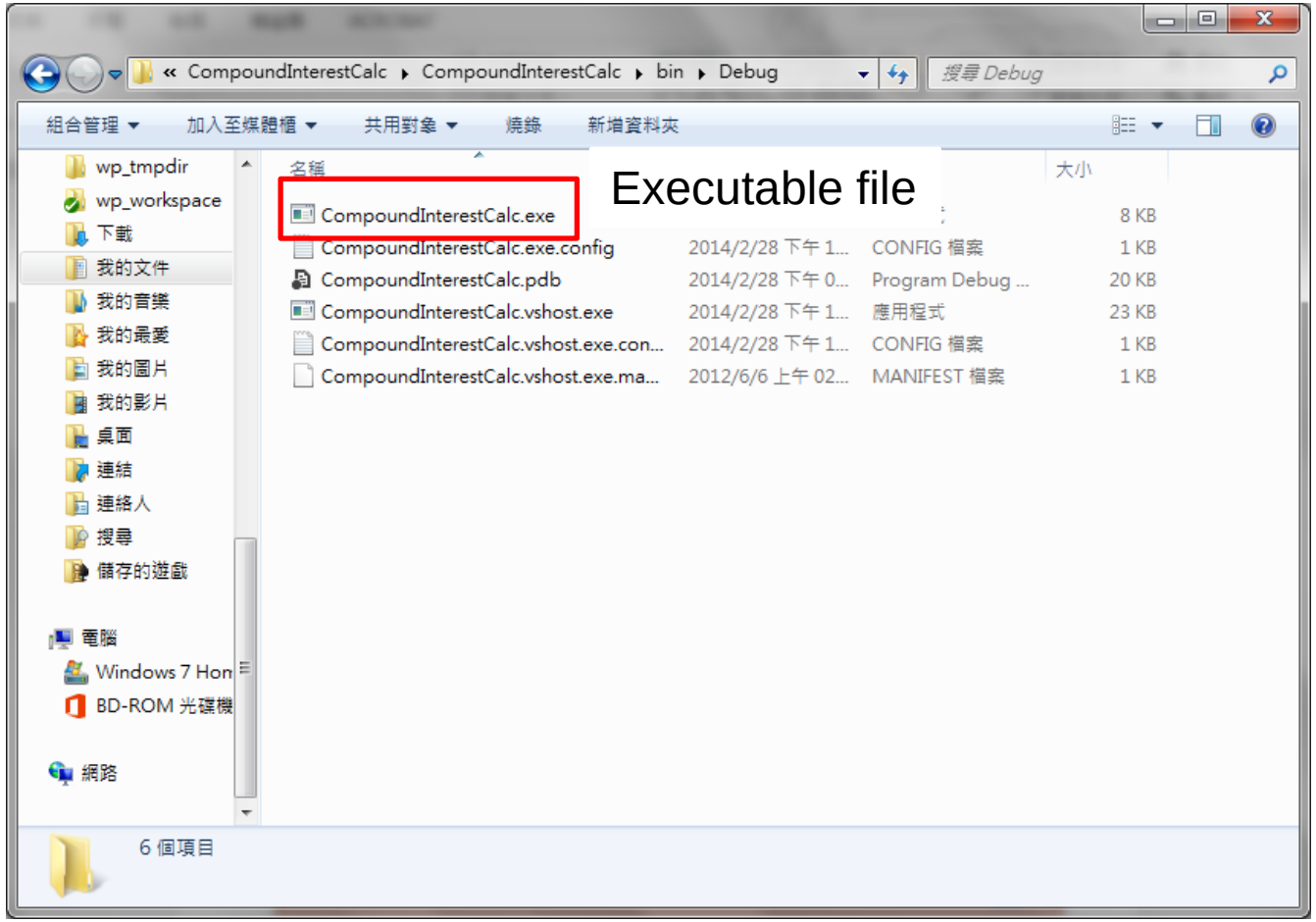
② 否(N) Preserve the original content

③ 取消 Return to IDE and continue editing

3. Open Project Directory

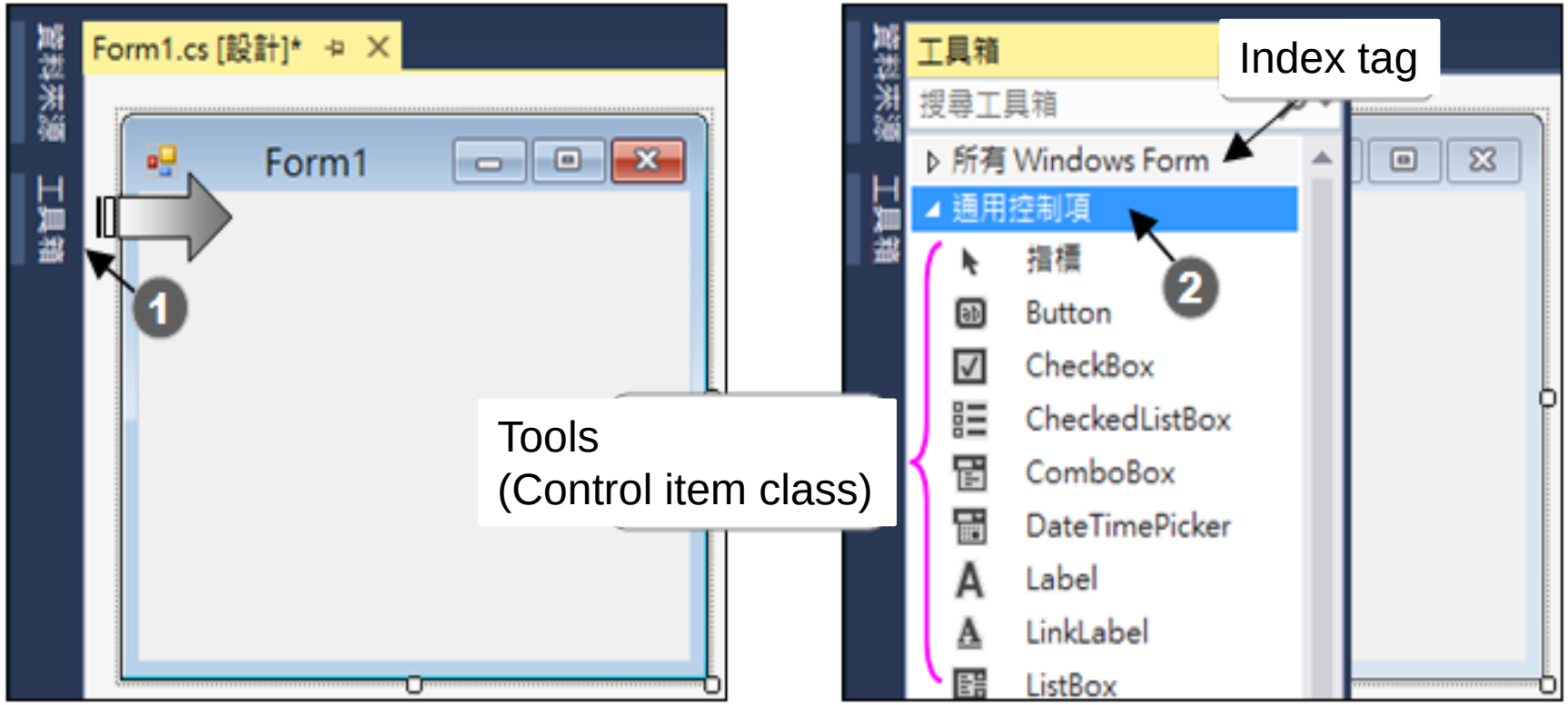


- **In bin\Debug directory:**
  the project is compiled and placed at the executable Windows application

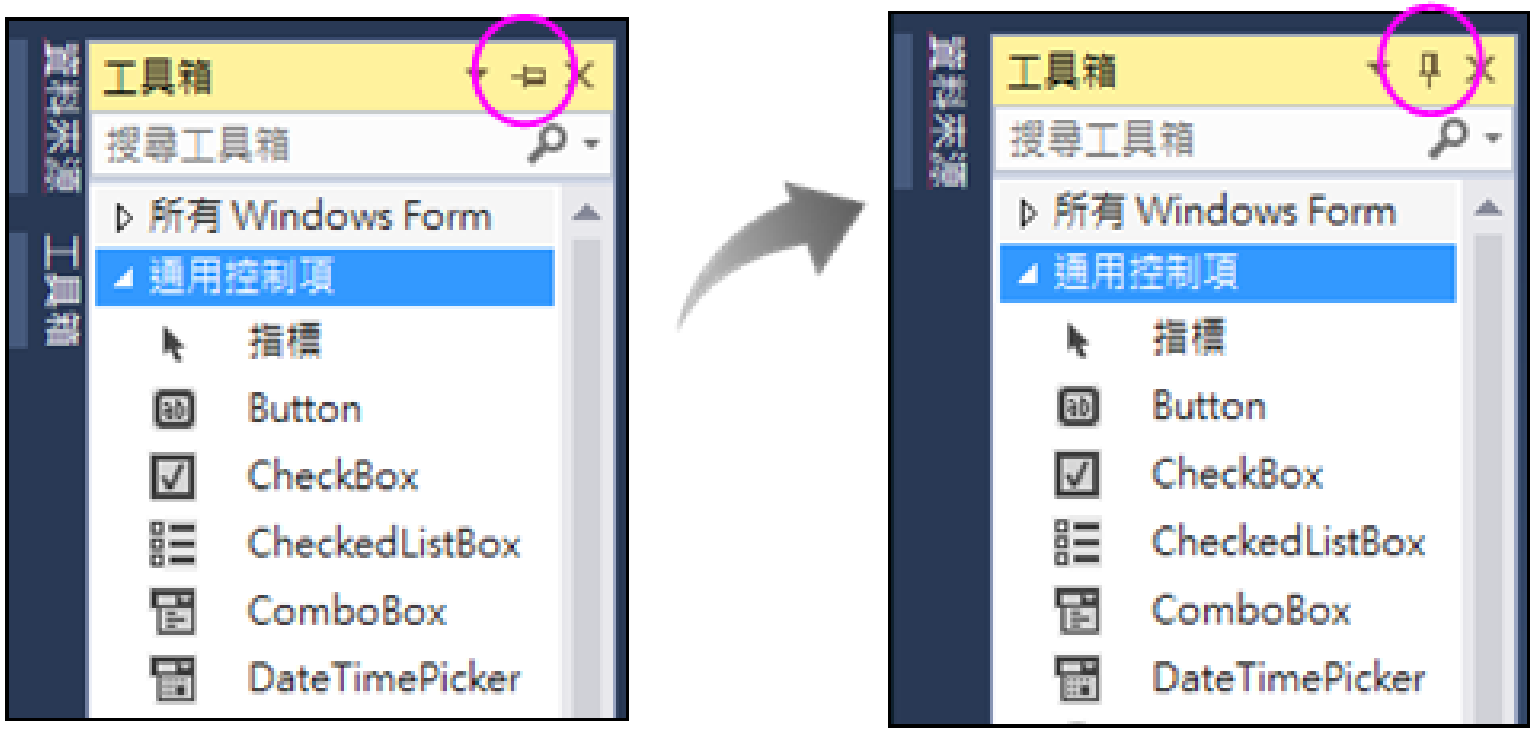


6-2 Development Environment of Windows Application

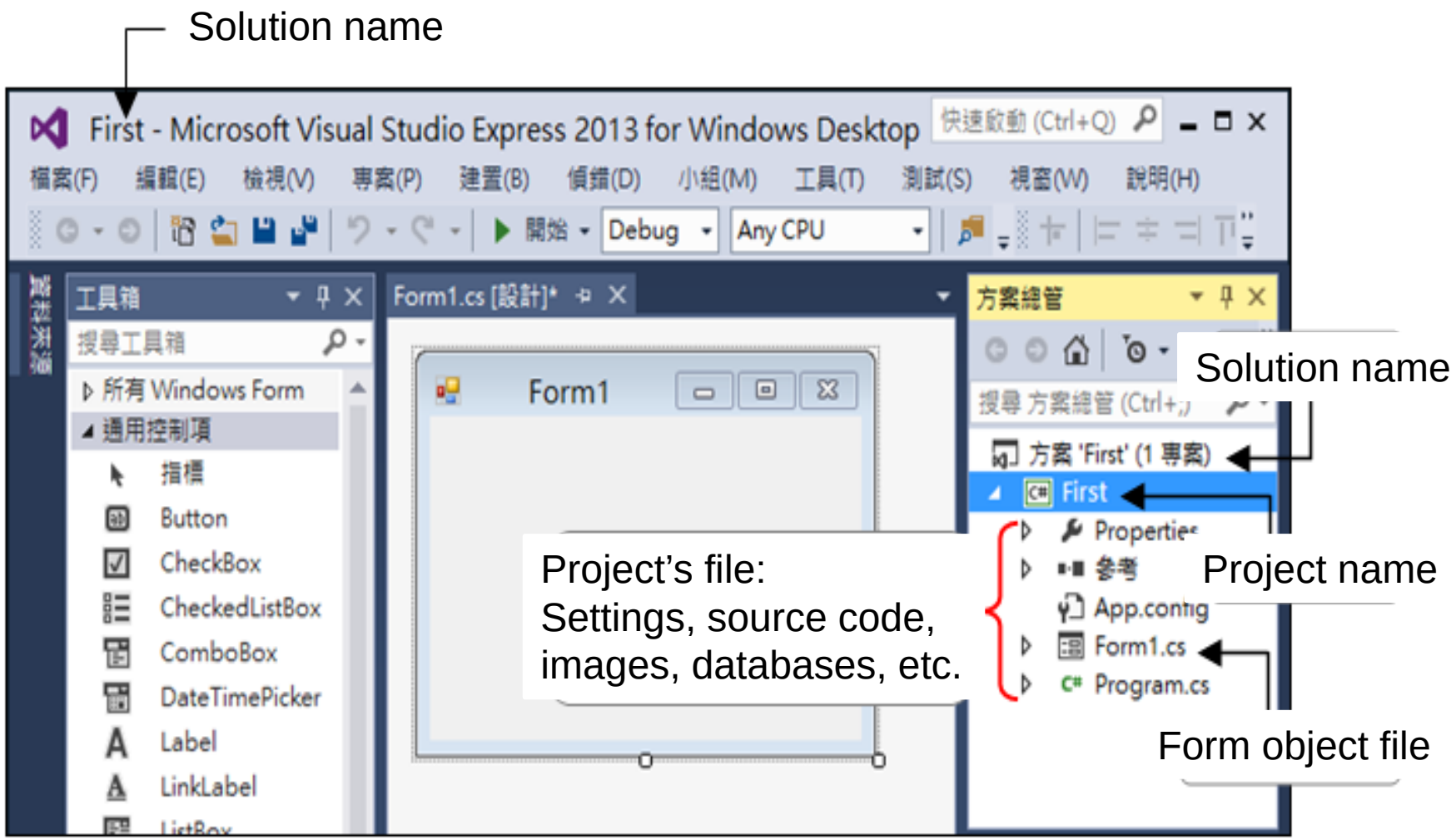
1. pop-up toolbox



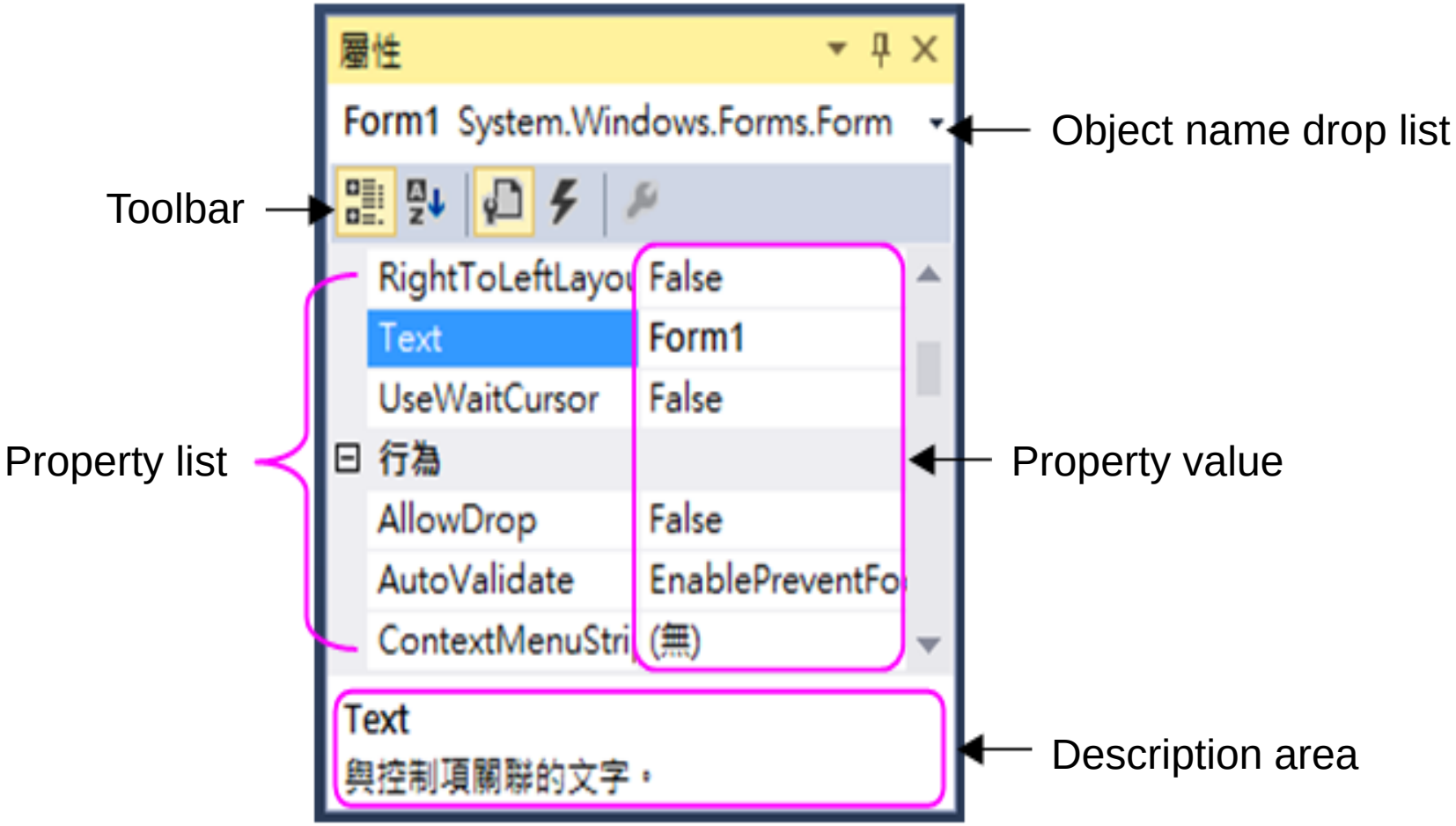
2. Fixed toolbox



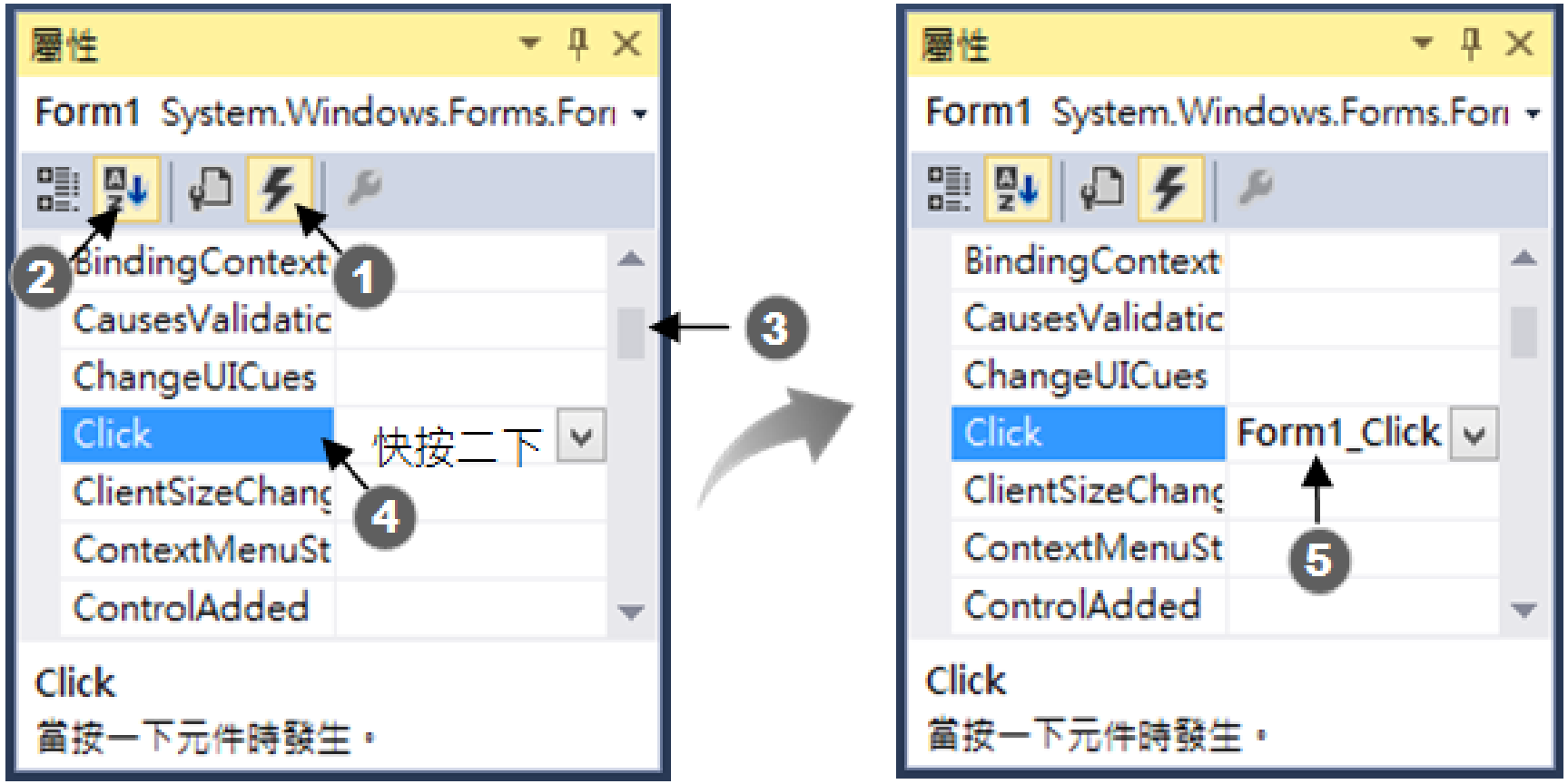
2. Project Manager



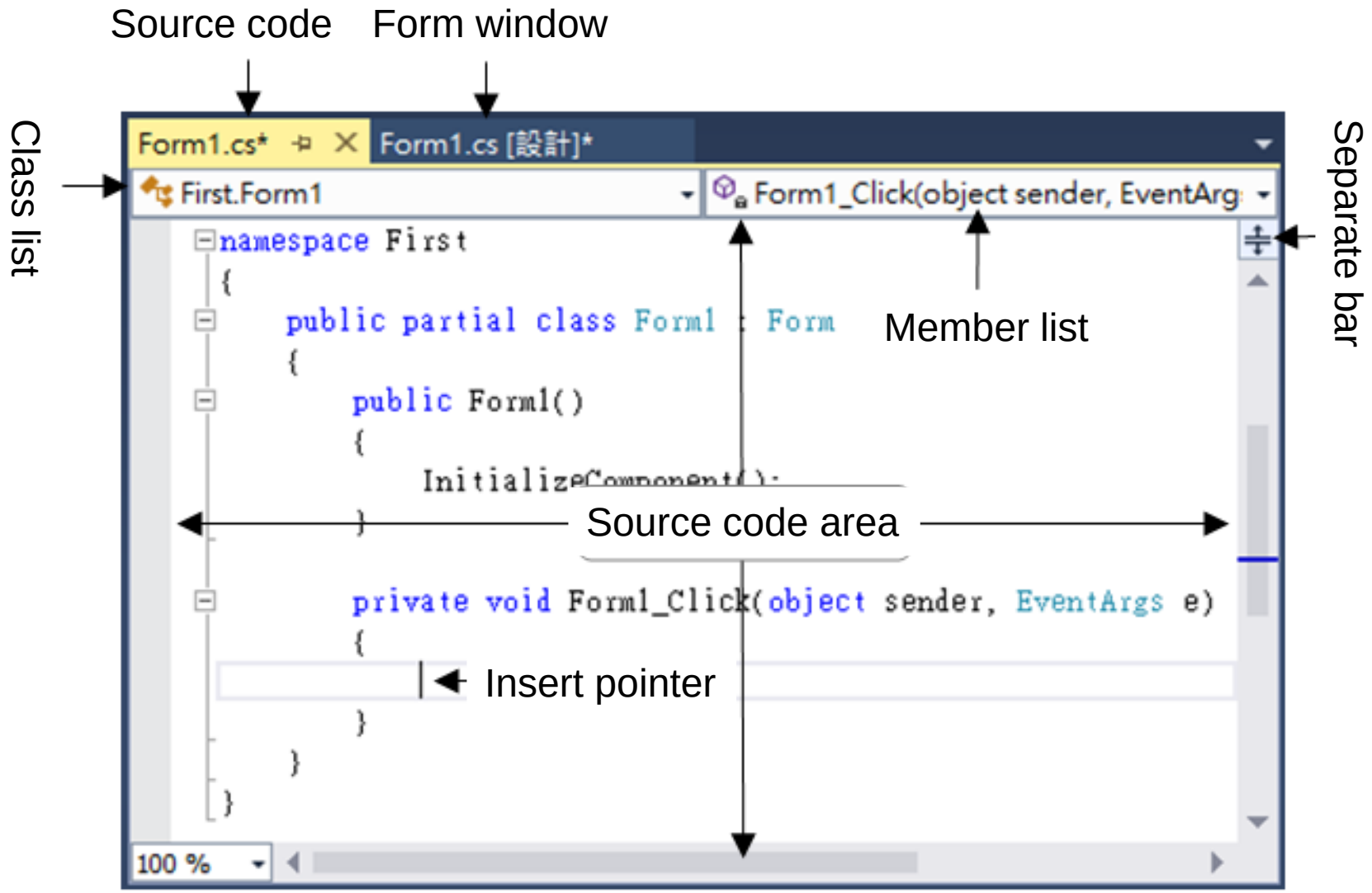
3. Property Window



2. Event Window



4. Source Code Window



6-3 Phases of Windows Application Design

4 Phases of Windows Application Design

1. Input and output interface creation
2. Property assignment

} Form design phase

3. Source code editing
4. Project building and debugging

} Program design and execution phase

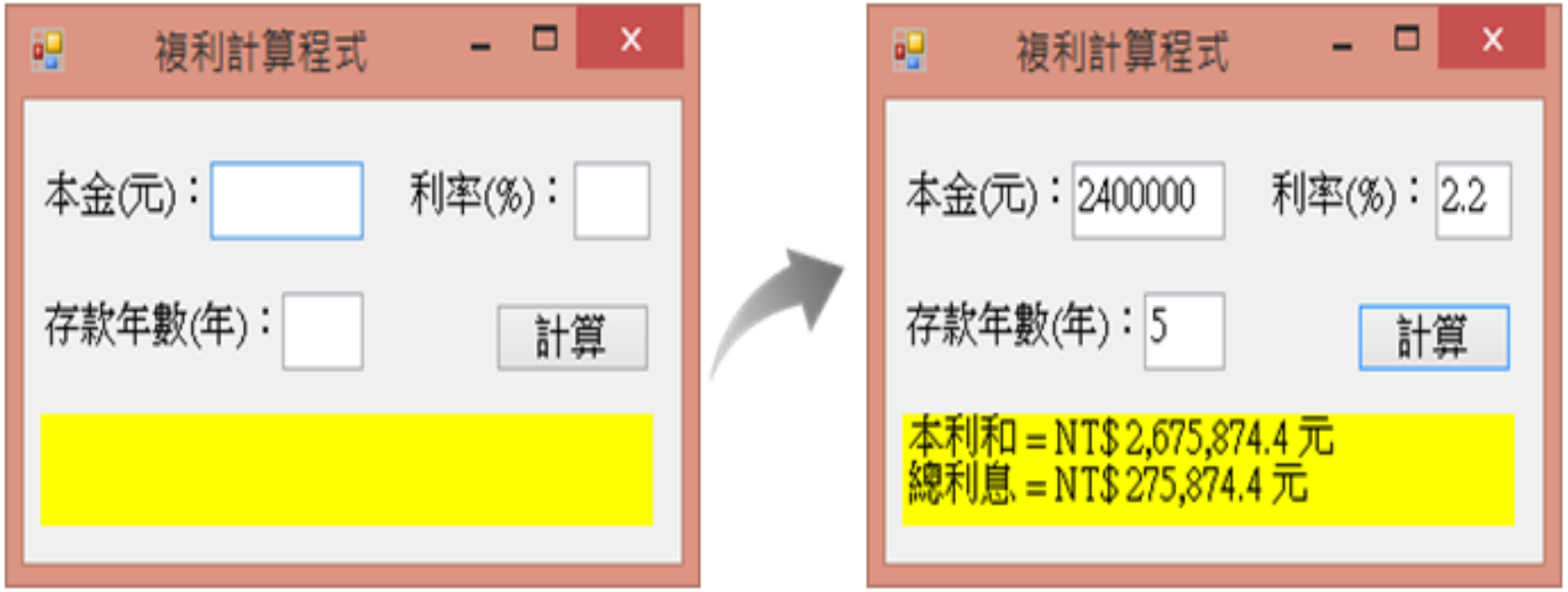
Practice(CompoundInterestCalc):

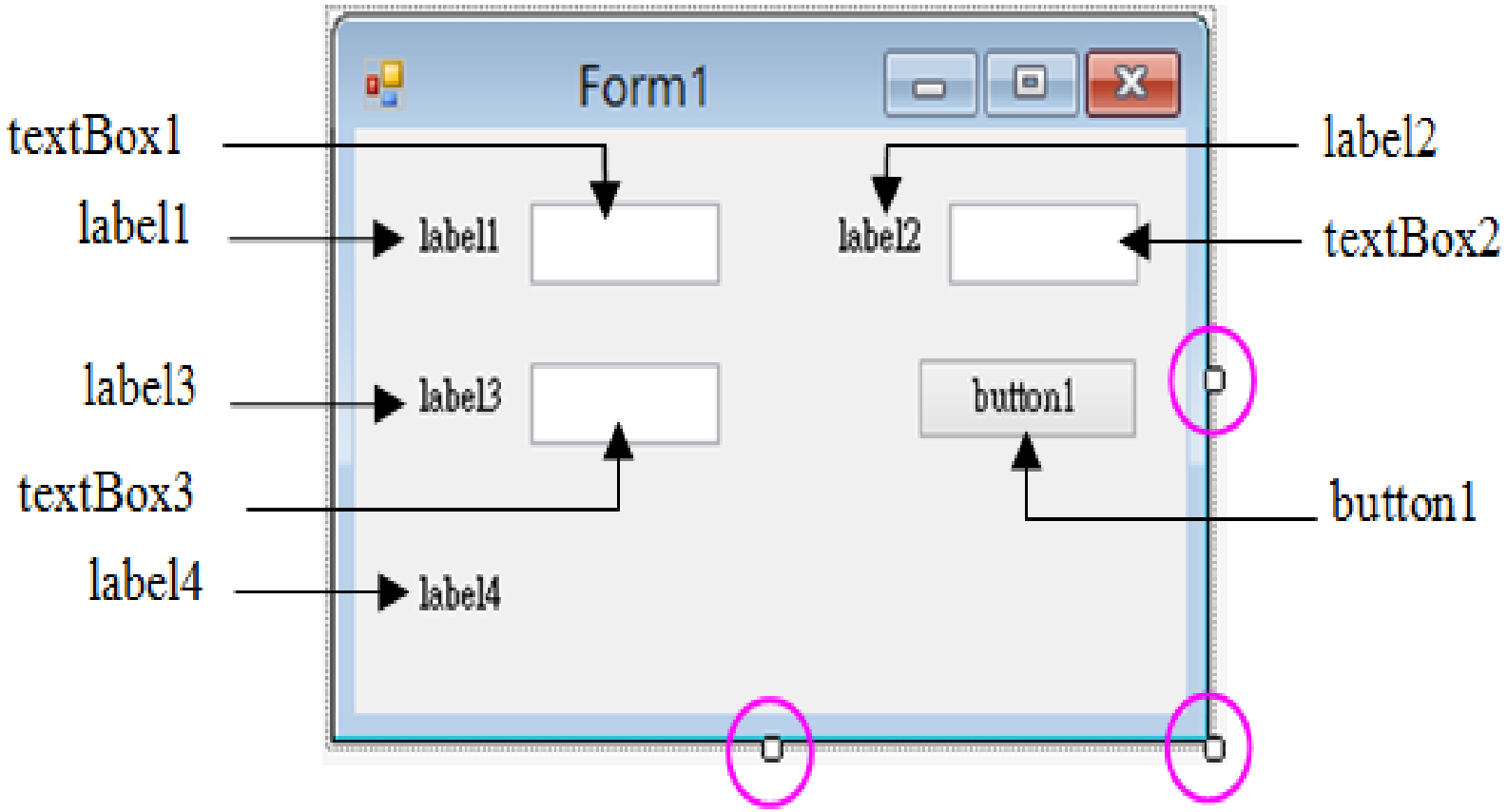
Follow the 4 phases mentioned just now to design a compound-interest-calculation Windows Application. The requirements are:

1. Provide 3 pairs of label and textbox control items for showing messages and data inputation
2. Provide a “Calculate” button control item
3. Provide a label control item for showing the sum of principal and interest, the sum of all interest in 2 lines.
4. When the principal, interest rate, saving years, press “Calculate” button to calculate the sum of principal and interest with the formula. The formula is:

   sum of principal and interest = principal x $(1 + \text{interest rate})^{\text{years}}$
   function: Math.Pow(2, 3) = 2^3
   interest = sum of principal and interest – principal

Result:


複利計算程式
本金(元)：
利率(%)：
存款年數(年)：
計算
複利計算程式
本金(元)：2400000
利率(%)：2.2
存款年數(年)：5
計算
本利和 = NT$ 2,675,874.4 元
總利息 = NT$ 275,874.4 元

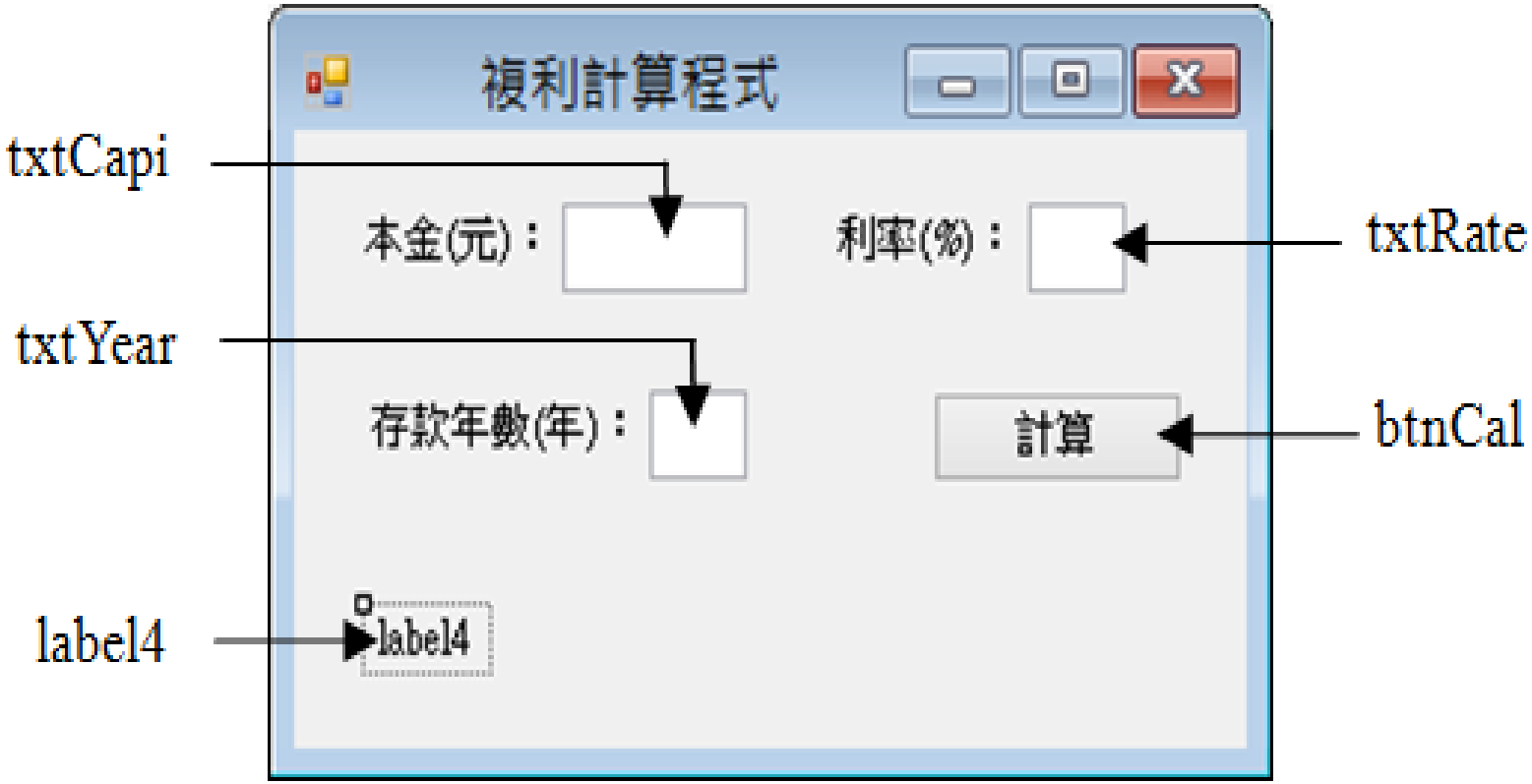

Form1
textBox1
label1
label3
textBox3
label4
label1
label2
label3
button1
label4
label2
textBox2
button1

3 ways to set properties

1. Input type: directly set the value of property
2. Menu type: select a value from all available property values
3. Dialog type: show a dialog window to programmer for advanced settings

Set the Name and Text properties

1. label1: Name reserved, Text = “Principal:”
2. label2: Name reserved, Text = “Years:”
3. label3: Name reserved, Text = “Rate(%):”
4. textBox1: Name = txtCapi, Text = “” (empty string)
5. textBox2: Name = txtYear, Text = “”
6. textBox3: Name = txtRate, Text = “”
7. button1: Name = btnCal, Text = “Calculate”
8. label4: Name reserved, Text reserved


複利計算程式
txtCapi
本金(元)：
利率(%)：
txtRate
txtYear
存款年數(年)：
計算
btnCal
label4
label4

3. Source Code

```
// FileName : Compound.sln
using System;
using System.Collections.Generic;
using System.ComponentModel;
using System.Data;
using System.Drawing;
using System.Linq;
using System.Text;
using System.Threading.Tasks;
using System.Windows.Forms;

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

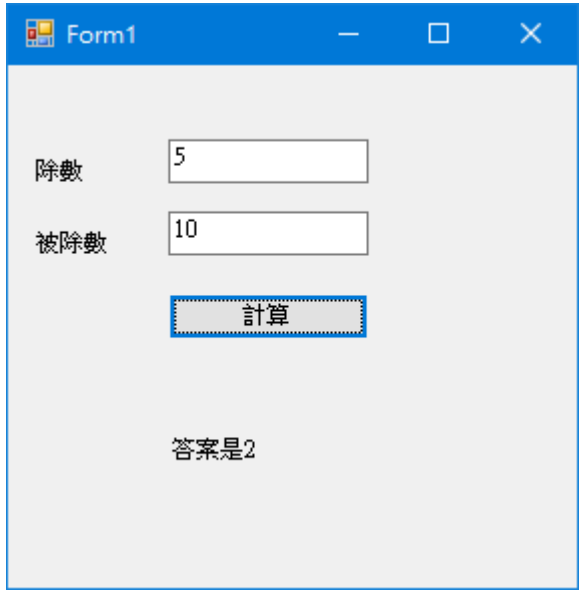
Automatically generated by IDE

```
19
20      private void Form1_Load(object sender, EventArgs e)
21      {
22          label4.Text = "";
23      }
24
25      private void btn_Click(object sender, EventArgs e)
26      {
27          double money = double.Parse(txtCapi.Text);   // 本金
28          double years = double.Parse(txtYear.Text);    // 年期
29          double yrate = double.Parse(txtRate.Text);    // 年利率
30          double total;        // 本利和
31          total = money * Math.Pow((1 + yrate / 100), years);
32          label4.Text = "本利和 = NT$ " + total.ToString("#,#.0") + " 元";
33          label4.Text += "\n 總利息 = NT$ " + (total - money).ToString("#,#.0") + " 元";
34      }
35  }
36 }
```

Practice 3

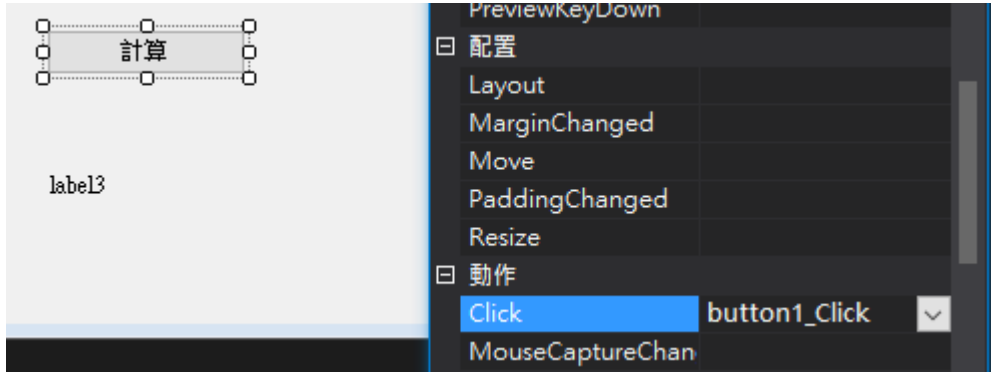
- **Take a simple practice.**

Use the Button event to calculate and give the answer.



Practice 3

- Tips:
  - If you want to give an element an event, click the element and find the  icon in the attribute window.
  - Use the click button event as an example, set the name as button1_Click in the click event. Then you can see the function in the code script.




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

4. Debug

1. **Start debug**
   press F5 or Debug(D)/Start debug(S) to compile and run the program, examine whether every function fits the requirements or not
2. **Stop debug**
   press the close button at the right-top side of window  **or Debug(D)/Stop debug(E) to terminate the program and return to IDE.**

Debug

- 1. press the mouse button and mark the line you feel has problem

```
148                     Console.WriteLine("ID\tName\tScore");
149                     Console.WriteLine("==================================");
150                     for (int i = 0; i < 10; i++)
151                     {
152                         Console.Write("{0}\t{1}\t{2}\n", id[i], allname[i], score[i]);
153                     }
```

- 2. press f5 to start the program, it will stop at that line you mark and press f11 you can run the program step by step and see the detail of all variable.

```
148                     Console.WriteLine("ID\tName\tScore");
149                     Console.WriteLine("==================================");
150                     for (int i = 0; i < 10; i++)
151                     {            i 0
152                         Console.Write("{0}\t{1}\t{2}\n", id[i], allname[i], score[i]);
153                     }
```

Practice-debug

- Hint:
- Use the breakpoint to find out the bug in the code and debug.
- There are several bugs here:
  - Only the integer can be shown.
  - Wrong calculate results.
  - Not able to get rid of the loops
- Please use the debug way (e.g. appointed breakpoint).
- Upload the screenshot of the result to moodle.

6-4 Set Properties in Source Code

Grammar
Object.propertyName = value;

Ex1: set the Text property of button1 to “計算”, the property value is a string, usage:

```
button1.Text = "計算";
```

Ex2: set the Width property of button1 to 75, the property value is an integer, usage:

```
button1.Width = 75;
```

Ex3: set the Enabled property of button1 to false, the property value is a Boolean, usage:

```
button1.Enabled = false;
```

2. Integrated Enumeration Class

Grammar

```
Object.propertyName = Enumeration.member;
```

Example

```
Object.ForeColor = Color.member;

Object.BackColor = Color.member;
```

The list of Color enumeration in common use:

Member	Color	Member	Color	Member	Color
Black	black	Navy	navy blue	Red	red
Blue	blue	Olive	olive	Silver	silver
Brown	brown	Orange	orange	SkyBlue	sky blue
Gold	gold	OrangeRed	orange red	Tomato	tomato
Green	green	Pink	pink	White	white
Gray	gray	Purple	purple	Yellow	yellow

Ex: set the BackColor property of label4 to yellow

```
label4.BackColor = Color.Yellow;
```

Use the principal of 3 primary colors – Red, Green, Blue, use method FromArgb(R, G, B) to mixing color, the number scope of color is 0~255

Grammar
Object.ForeColor = Color.FromArgb(R, G, B); Object.BackColor = Color.FromArgb(R, G, B);

Ex1: set the background color of the form to “blue”

```
this.BackColor = Color.FromArgb(0, 0, 255);
```

“this” stands for the form, this.BackColor stands for the background color of the form

Ex2: set the background color of button1 to “purple” (red + blue)

```
button1.BackColor = Color.FromArgb(255, 0, 255);
```

Ex3: set the background color of the form to white(red + green + blue)

```
this.BackColor = Color.FromArgb(255, 255, 255);
```

Ex4: set the background color of textBox1 to black

```
textBox1.BackColor = Color.FromArgb(0, 0, 0);
```

Ex5: set the background color of the form to gray

```
this.BackColor = Color.FromArgb(125, 125, 125);
```

2. BorderStyle Enumeration: BorderStyle

Member	None	FixedSingle	Fixed3D
Description	No border	Single line border	3-D border line
Example	label1	label1	label1
Code	BorderStyle.None	BorderStyle.FixedSingle	BorderStyle.Fixed3D

Ex: set the border of label1 to 3-D border line, usage:

label1.BorderStyle = BorderStyle.Fixed3D;

3. TextAlign Enumeration: ContentAlignment

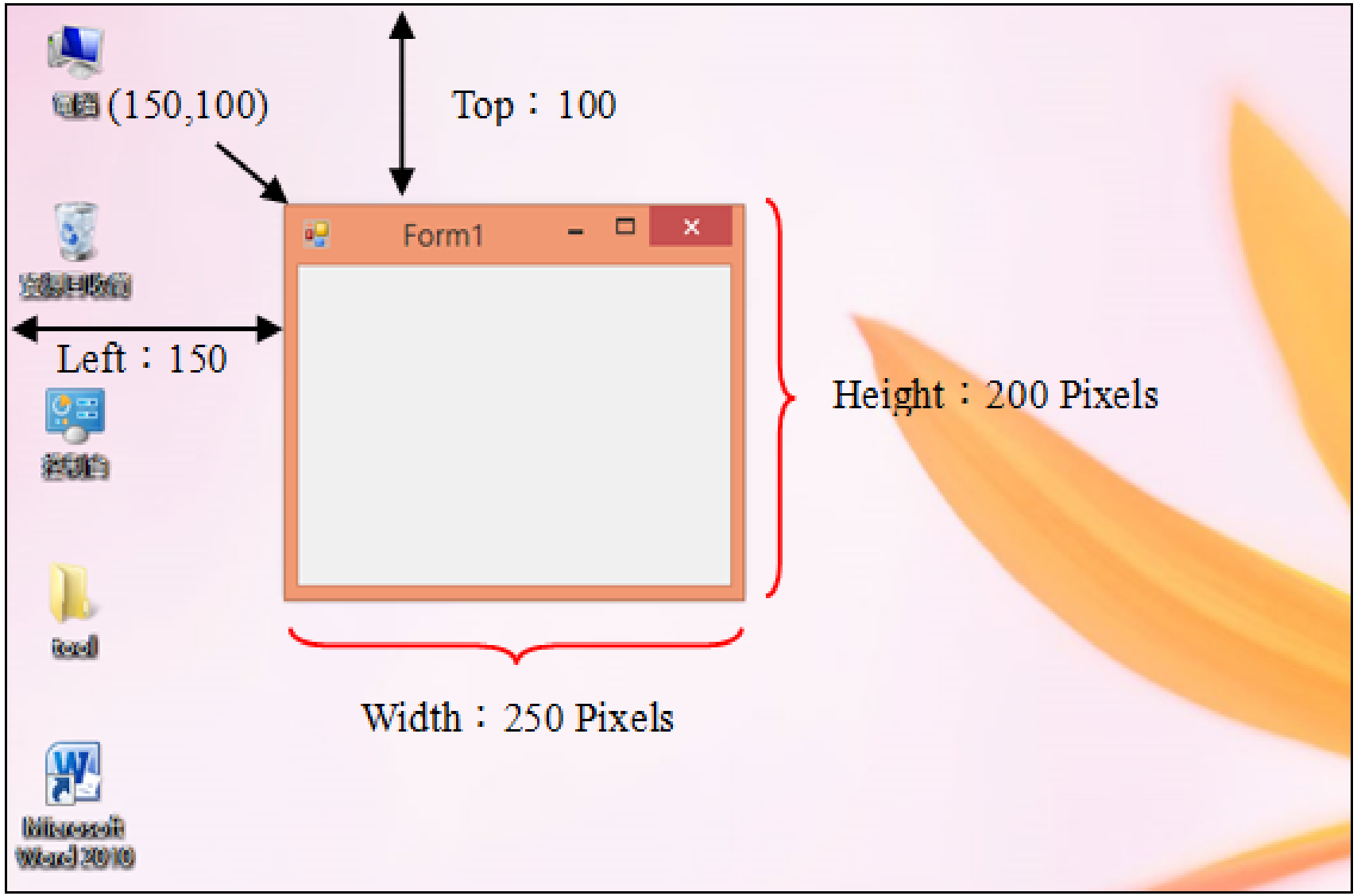
TopLeft	TopCenter	TopRight
MiddleLeft	MiddleCenter	MiddleRight
BottomLeft	BottomCenter	BottomRight

Ex: set the align of the text “計算” of button1 to the right-bottom of the control item, usage:


```
button1.TextAlign = ContentAlignment.BottomRight;
```

3. new

Grammar
Object.propertyName = new className(arg1, arg2, …);


(150,100)
Top：100
Form1
Left：150
Height：200 Pixels
Width：250 Pixels

Ex: use Top and Left properties to set the coordinate, usage:

```
this.Top = 150;
this.Left = 100;
```

Ex: set the coordinate of form to (150, 100) from left-top of the screen, usage:

```
this.Location = new Point(150, 100);
```

Ex: use Width and Height properties to set the size of the form, usage:

```
this.Width = 250;
this.Height = 200;
```

Ex: Size property can include Width and Height properties, usage:

```
this.Size = new Size(250, 200);
```

Image and BackgroundImage Properties

Grammar
Object.Image = Image.FromFile(“imagePath”); Object BackgroundImage = Image.FromFile(“imagePath”);

Grammar
Object.Image = Image.Bitmap(“imagePath”); Object BackgroundImage = Image.Bitmap(“imagePath”);

Ex1: load “C:\cs2013\ch06\duck.jpg” as the background image of the form:

```
this.BackgroundImage = Image.FromFile ("c:\\cs2013\\ch06\\duck.jpg");
this.BackgroundImage = new Bitmap("c:\\cs2013\\ch06\\duck.jpg");
```

Ex2: remove the background image of the form

```
this.BackgroundImage = null;
```

3. Use Font property to set the style of the Text property content, like font, size, style, etc. usage:

Grammar
Object.Font = new Font(fontName, fontSize, fontStyle);

Five font styles:

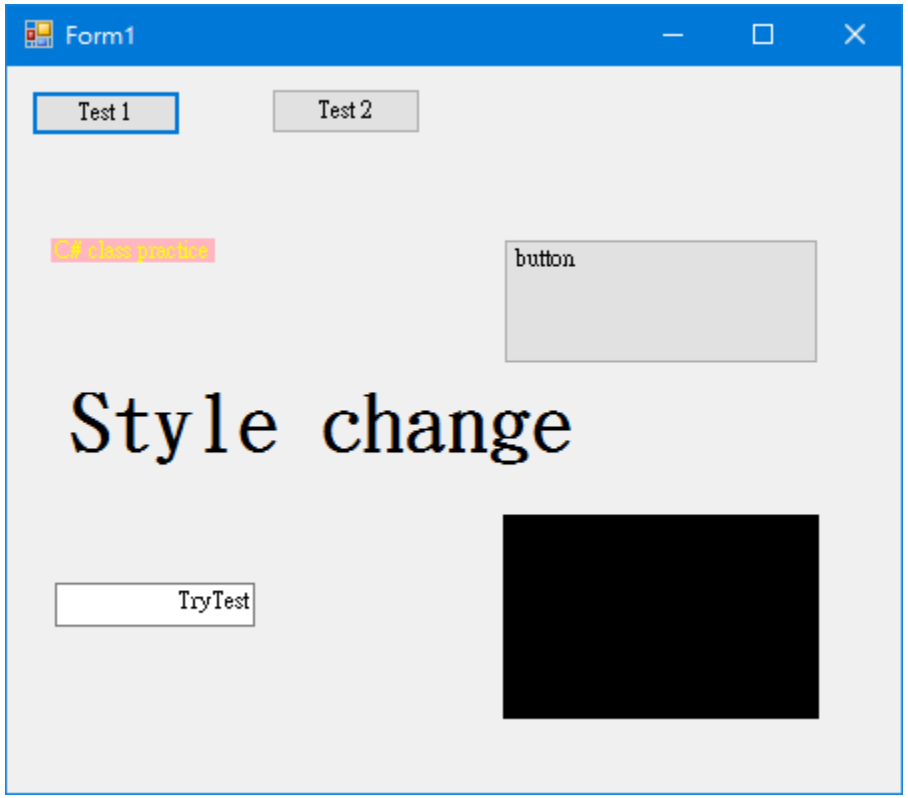
1. FontStyle.Bold
2. FontStyle.Italic
3. FontStyle.Regular
4. FontStyle.Strikeout
5. FontStyle.Underline

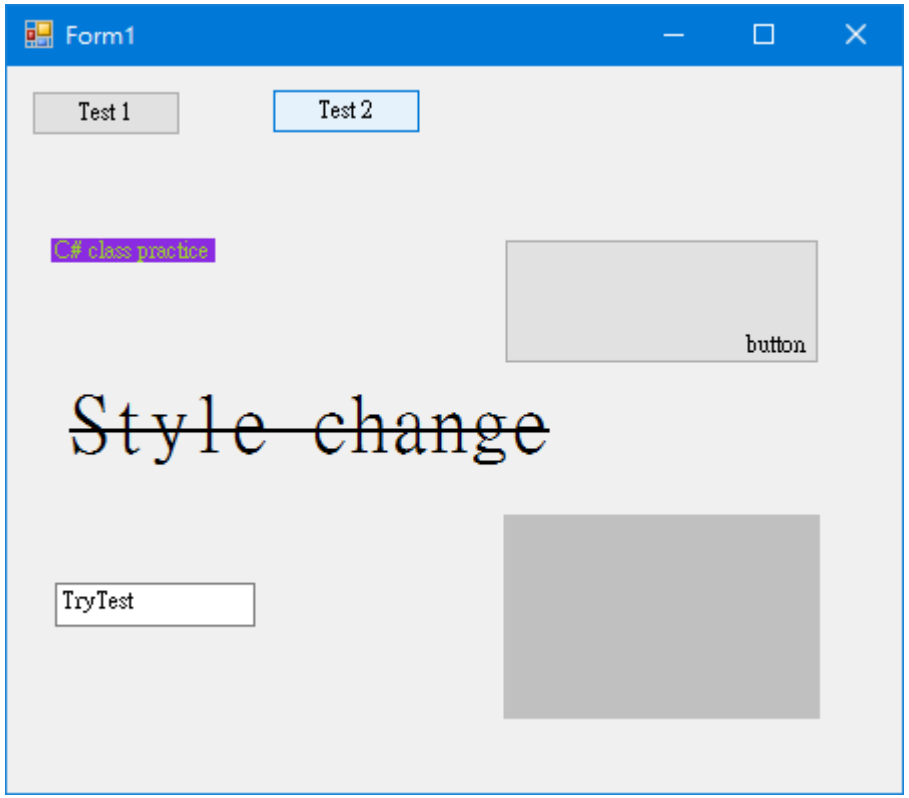
Practice (change)

- **Test 1 :**
  label 1 backcolor: LightPink ForeColor: Yellow
  label 2 – font: 標楷體, 30, Bold
  button3 – text align: top left
  textBox1 – text align : Right
  picture box – background color: Black

Practice (change)

- **Test 2 :**
  label 1 backcolor: BlueViolet ForeColor: YellowGreen
  label 2 – font: 細明體, 30, Strikeout
  button3 – text align: BottomRight
  textBox1 – text align : Left
  picture box – background color: Silver


Form1
Test 1
Test 2
C# class practice
button
Style change
TryTest


Form1
Test 1
Test 2
C# class practice
button
Style change
TryTest

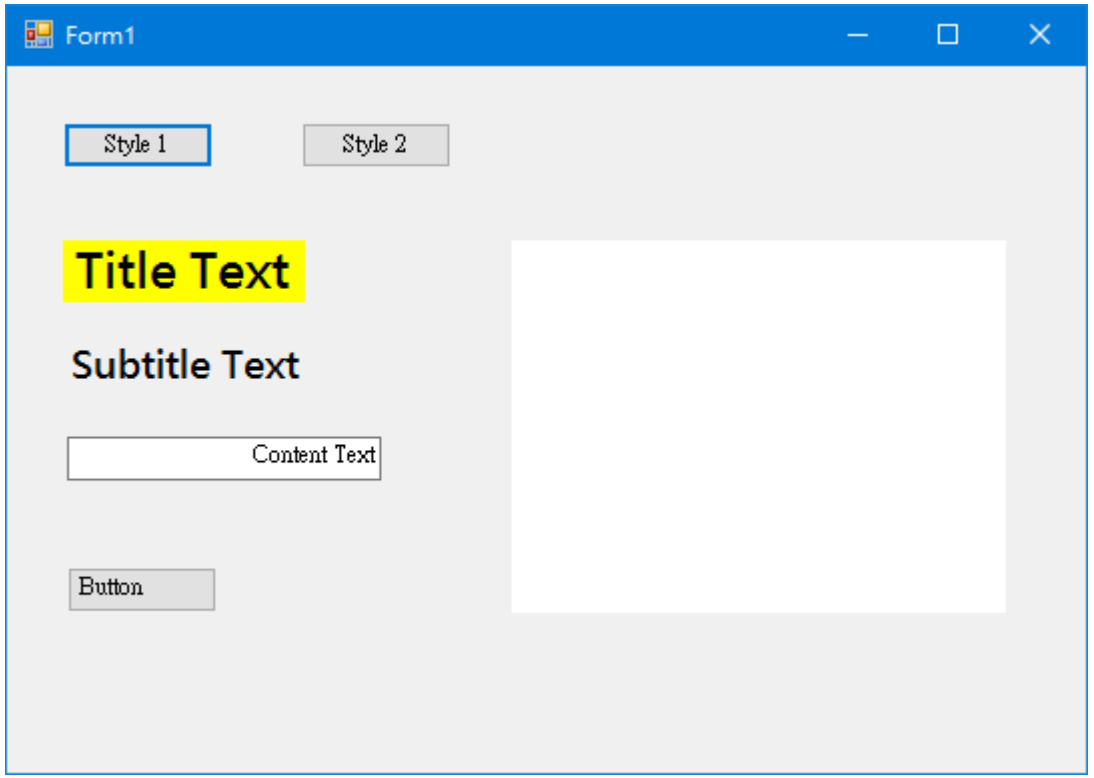
Stu_Practice(WindowStyleChange)

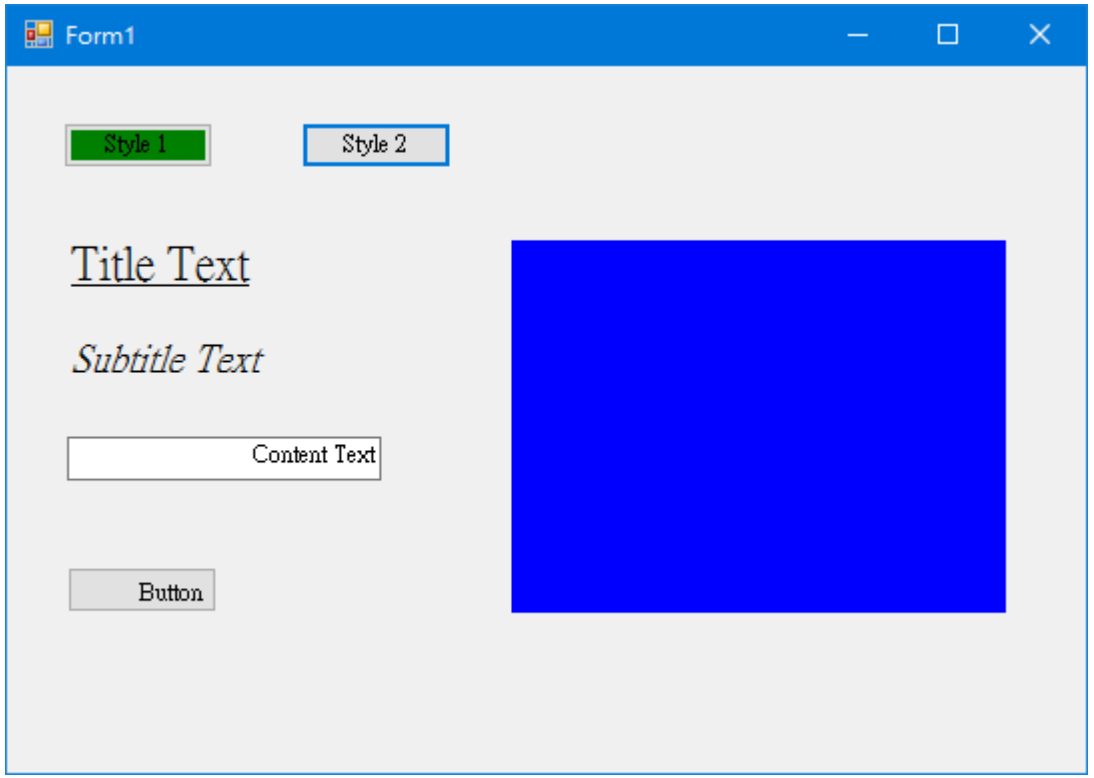
- Design a style previewer, 2 styles
- Style1:
  title – background color: Yellow, font: 微軟正黑體, 18, bold
  subtitle – font: 微軟正黑體, 14, bold
  button – text align: bottom right
  picture box – background color: White

Practice (cont’d)

- **Style2:**
  title – background color: Transparent, font: 新細明體, 18, underline
  subtitle – font: 新細明體, 14, italic
  button – text align: Right
  picture box – background color: blue

Result


Form1
Style 1
Style 2
Title Text
Subtitle Text
Content Text
Button


Form1
Style 1
Style 2
Title Text
Subtitle Text
Content Text
Button

End

Take a Break …