

.NET 程式設計入門（使用 C#）

Outline

- **ArrayList** 集合類別
- 陣列類別
- 字串類別
- 其它常用類別
- 引用外部類別
- 程式偵錯
- 例外處理

MSDN Library

- **MSDN Library** 含括大量的專業程式設計資訊，其中包含範例程式碼、技術文件、白皮書及參考指南…
- 我們可以藉由工具列上 [說明] 中的 [內容]、[索引]、[搜尋]，來查詢 **MSDN Library**，了解欲使用的類別中擁在那些成員及各成員的用法等資訊
- 在程式碼撰寫視窗中，將游標放於欲查詢的類別或成員名稱上，可利用快速鍵 **F1** 查詢之
- 線上文件
 - 英文 –
<http://msdn.microsoft.com/library/>
 - 中文 –
<http://msdn.microsoft.com/library/cht/>

Object 型別

- **object** 型別是所有參考型別的基礎，因此所有類別都是直接或間接繼承自 **object** 型別
- 任何參考型別的值皆可指定至 **object** 型別
- **Boxing**
 - 將實值型別轉換成參考型別的動作稱為 **boxing**
 - **ex** : `object o = (int) 1;`

Object 型別

- **UnBoxing**
 - 將物件的值轉換成實值型別的值的过程稱之為 **UnBoxing**
 - **ex : int i = (int) o;**

ArrayList 類別 (1)

- 使用陣列時我們必須先設定好陣列的大小才可以使用，相當不方便
- **ArrayList** 屬於集合，集合大小會隨資料量大小動態改變
- 陣列與集合比較

項目	陣列	集合
自動調整大小	不可	可
儲存型別	相同	不同
執行速度	較快	較慢

ArrayList 類別 (2)

- 命名空間
 - **System.Collections;**
- 常用屬性
 - **Count** – 實際包含的元素個數
- 常用方法
 - **Add** – 將物件加入至末端
 - **Insert** – 將物件插入至指定位置
 - **Remove** – 移除第一個符合指定物件的元素
 - **Clear** – 清除所有元素
 - **Sort** – 以遞增方式排序元素
 - **Reverse** – 將元素次序反轉

實例探討 sample5-a1 (1)

- 程式功能
 - 將元素加入 **ArrayList** 中
 - 列印 **ArrayList** 中各元素值

- 程式內容

```
class car
{
    public car(int x)
    {
        id = x;
    }
    public int id;
}
```


實例探討 sample5-a1 (2)

- 程式內容

```
static void Main(string[] args)
{
    ArrayList myAL = new ArrayList();
    car myCar = new car(10);

    myAL.Add(12);
    myAL.Add("ives");
    myAL.Add(myCar);
    Console.WriteLine("myAL[{0}] : {1}",0 ,myAL[0]);
    Console.WriteLine("myAL[{0}] : {1}",1 ,myAL[1]);
    Console.WriteLine("myAL[{0}] : {1}",2 ,((car)myAL[2]).id);

    ((car)myAL[2]).id = 100;
    Console.WriteLine("myCar : " + myCar.id);
}
```

課堂練習 sample5-b1

- 程式功能
 - 請使用者輸入正整數資料 (不限筆數)
 - 當使用者輸入為負數停止
 - 將使用者輸入的資料排序後列印至螢幕
- 基本概念
 - 利用迴圈及 **ArrayList** 來儲存使用者輸入的資料
 - 利用 **Sort** 方法將資料進行排序動作

Outline

- ArrayList 集合類別
- 陣列類別
- 字串類別
- 其它常用類別
- 引用外部類別
- 程式偵錯
- 例外處理

陣列類別常用成員

- 常用屬性
 - **Rank** – 陣列的維度大小
 - **Length** – 陣列元素個數
- 常用方法
 - **GetLength** – 回傳某一維度的長度
 - **Clone** – 複製陣列內容至一新陣列實體
(不會複製參考所參考的物件)
- 靜態方法
 - **Sort** – 排序陣列元素
 - **IndexOf** – 回傳第一個符合參數值的索引位置

實例探討 sample5-a2 (1)

- 程式功能
 - 陣列類別屬性方法測試
 - 比較實值型別與參考型別

- 程式內容

```
class car
{
    public int id;
}
```

實例探討 sample5-a2 (2)

```
static void Main(string[] args)
{
    int[,] x = new int[2,3];
    Console.WriteLine("Length : " + x.Length);
    Console.WriteLine("Rank : " + x.Rank);
    Console.WriteLine("GetLength(0) : " +
        x.GetLength(0));
    Console.WriteLine("GetLength(1) : " +
        x.GetLength(1));
}
```

實例探討 sample5-a2 (3)

```
car[] myCar = new car[5];  
for(int i=0; i<myCar.Length; i++)  
{  
    myCar[i] = new car();  
    myCar[i].id = i;  
}  
Console.WriteLine();
```

實例探討 sample5-a2 (3)

```
//實值型別與參考型別比較  
x[0,0] = 100;  
myCar[0].id = 100;  
int[,] y = (int[,]) x.Clone();  
car[] myMoto = (car[]) myCar.Clone();  
y[0,0] = 0;  
myMoto[0].id = 0;
```


實例探討 sample5-a2 (4)

```
Console.WriteLine("x[0,0] : " + x[0,0]);  
Console.WriteLine("y[0,0] : " + y[0,0]);  
Console.WriteLine("myCar[0].id : " +  
myCar[0].id);  
Console.WriteLine("myMoto[0].id : " +  
myMoto[0].id);  
}
```

課堂練習 sample5-b2

- 程式功能
 - 建立一個型別為 **int** 的一維陣列，內含 10 個元素，元素值利用亂數來決定 (1~10)
 - 請使用者輸入一個數字 (1~10)
 - 列印出陣列當中是否包含該數字，若有則列印該數字在陣列當中的位置
 - 最後將陣列中所有元素列印出來
- 基本概念
 - 利用 **IndexOf** 靜態方法找出所在位置

Outline

- ArrayList 集合類別
- 陣列類別
- 字串類別
- 其它常用類別
- 引用外部類別
- 程式偵錯
- 例外處理

字串類別常用成員

- 常用屬性
 - **Length** – 字串字元個數 (中文長度計算同英文)
- 常用方法
 - **ToUpper / ToLower** – 字串大小寫轉換
 - **Insert** – 字串插入

字串類別常用成員

- **PadLeft / PadRight** – 以指定字元補足指定長度
- **IndexOf** – 找出指定文字在字串中第一次出現的位置
- **Split** – 將字串依指定分隔字元切割依序存入字串陣列
- **Replace** – 將字串中部份文字以其它文字取代
- **SubString** – 取得字串中指定範圍的文字

實例探討 sample5-a3

- 程式功能
 - 字串類別屬性方法測試

- 程式內容

```
string str = "Alice is a good student!!";  
Console.WriteLine(str);  
Console.WriteLine("ToUpper : " +  
    str.ToUpper());
```

實例探討 sample5-a3

```
Console.WriteLine("ToLower : " +  
    str.ToLower());
```

```
Console.WriteLine("Insert : " +  
    str.Insert(6,"Wang "))
```

```
string[] strArray = str.Split(' ');  
for(int i=0; i<strArray.Length; i++)  
    Console.WriteLine(strArray[i]);
```

課堂練習 sample5-b3

- 程式功能
 - 請使用者輸入多個整數值 (以逗號區隔)
 - 列印輸入的整數值總和
- 基本概念
 - 利用 **Split** 方法取得輸入字串中的各數值

格式化字元

- 貨幣格式 - **C** 或 **c**
 - `Console.WriteLine("{0:c}",4.5);` //NT\$4.50
- 十進位數 - **Dn** 或 **dn**
 - `Console.WriteLine("{0:d5}",4);` //00004
- 科學記號 - **E** 或 **e**
 - `Console.WriteLine("{0:e}",4);` //4.000000e+001
- 定點格式 - **Fn** 或 **fn**
 - `Console.WriteLine("{0:f3}",4);` //4.000
- 數值格式 - **Nn** 或 **nn**
 - `Console.WriteLine("{0:n2}",4000);` //4,000.00
- 十六進位 - **X** 或 **x**
 - `Console.WriteLine("{0:x}",10);` //a

自訂數值格式輸出字串

- 0
 - 顯示數值位數比設定格式位數小時，資料左邊以 0 填滿位數
- #
 - 顯示數值位數比設定格式位數小時，資料右邊以空白填滿位數
- .
 - 以小數點右方位數決定小數位數，超過位數以四捨五入處理

自訂數值格式輸出字串

- ,
 - 使用千位分隔符號，千位分隔放在最後，則數值除以 1000 顯示，放置二個除以 1000000 顯示
- %
 - 數值乘以 100 顯示

實例探討 sample5-a4 (1)

- 程式功能
 - 測試自訂數值格式輸出字串
- 程式內容

```
double myvar1 = 0801234567;  
Console.WriteLine(" 1." +  
    myvar1.ToString("(0##)###-#####"));
```

實例探討 sample5-a4 (1)

```
int myvar2 = -12345;  
Console.WriteLine(" 2." +  
    myvar2.ToString("#####"));  
Console.WriteLine(" 3." +  
    myvar2.ToString("000000"));
```

實例探討 sample5-a4 (2)

- 程式內容

```
double myvar3 = -2.455;  
Console.WriteLine(" 4." +  
    myvar3.ToString("#.##"));  
Console.WriteLine(" 5." +  
    myvar3.ToString("00.00"));  
Console.WriteLine(" 6." +  
    myvar3.ToString("00.00000"));
```

```
double myvar4 = 1234567890;
```

實例探討 sample5-a4 (2)

```
Console.WriteLine(" 7." +  
    myvar4.ToString("#,#"));  
Console.WriteLine(" 8." +  
    myvar4.ToString("#,"));  
Console.WriteLine(" 9." +  
    myvar4.ToString("#,,"));  
Console.WriteLine("10." +  
    myvar4.ToString("#,,,"));  
  
double myvar5 = 0.086;  
Console.WriteLine("11." +  
    myvar5.ToString("#.#%"));
```

Outline

- ArrayList 集合類別
- 陣列類別
- 字串類別
- 其它常用類別
- 引用外部類別
- 程式偵錯
- 例外處理

其它類別

- **DateTime**
 - 提供時間與日期相關的設定 (含閏年查詢靜態方法)
- **Math**
 - 提供許多關於三角函數、指數、對數等常見數學函數運算
- **Random**
 - 提供亂數方面的功能

實例探討 sample5-a5

- 程式功能
 - DateTime 及 Math 類別靜態方法測試
- 程式內容

```
Console.WriteLine("現在時刻：" +  
    DateTime.Now);
```

```
Console.WriteLine("2 的 10 次方為 " +  
    Math.Pow(2,10));
```

```
Console.WriteLine("-2 的絕對值為 " + Math.Abs(-  
    2));
```

```
Console.WriteLine("Sin(pi/2) 為 " +  
    Math.Sin(Math.PI/2));
```

課堂練習 sample5-b4

- 程式功能
 - 請使用者輸入一年份
 - 列印該年是否為閏年
- 基本概念
 - 利用 **DateTime** 中判斷閏年的方法來完成

Outline

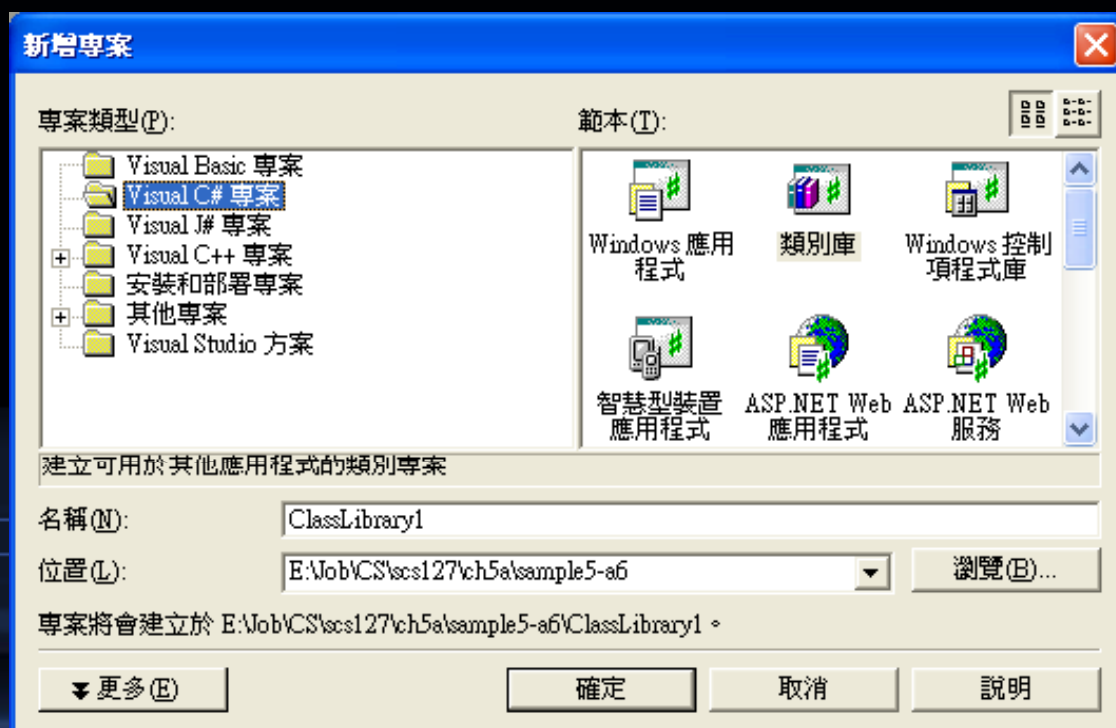
- ArrayList 集合類別
- 陣列類別
- 字串類別
- 其它常用類別
- 引用外部類別
- 程式偵錯
- 例外處理

引用外部類別 (1)

- 我們可以將常用的類別寫在類別庫專案當中，提供自己或是外部專案來引用
- 要引用外部類別必須將該類別所屬的組件檔 (.dll) 加入至欲引用的專案參考中
- 欲使用某一類別時，尚需利用 **using** 關鍵字加入該類別所屬命名空間

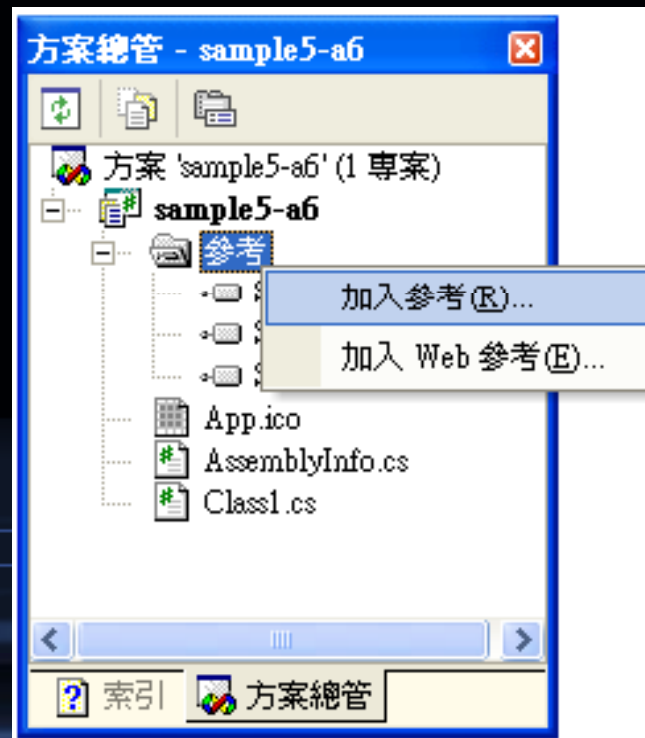
引用外部類別 (2)

- 將常用類別撰寫於類別庫當中
- 利用工具列中『建置』『建置方案』進行編譯產生組件檔 (.dll)



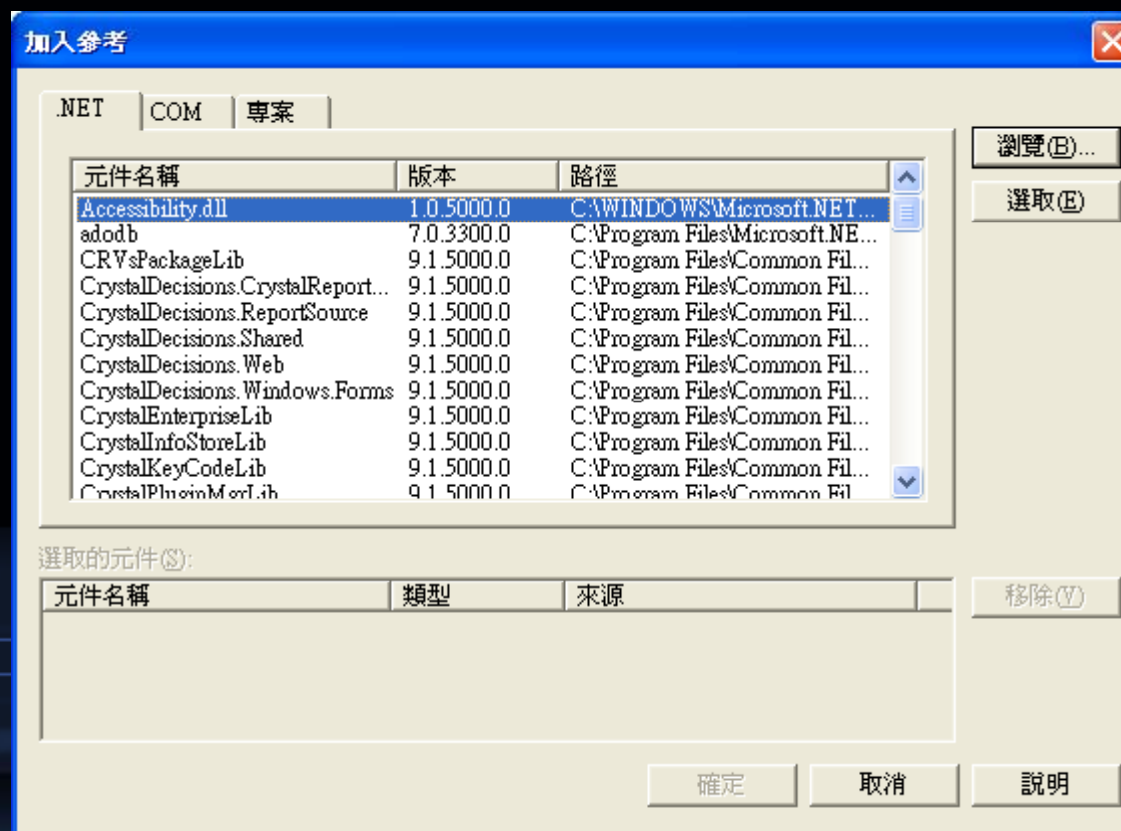
引用外部類別 (3)

- 若要引用該類別庫類別，在欲引用專案的方案總管參考部份點選右鍵



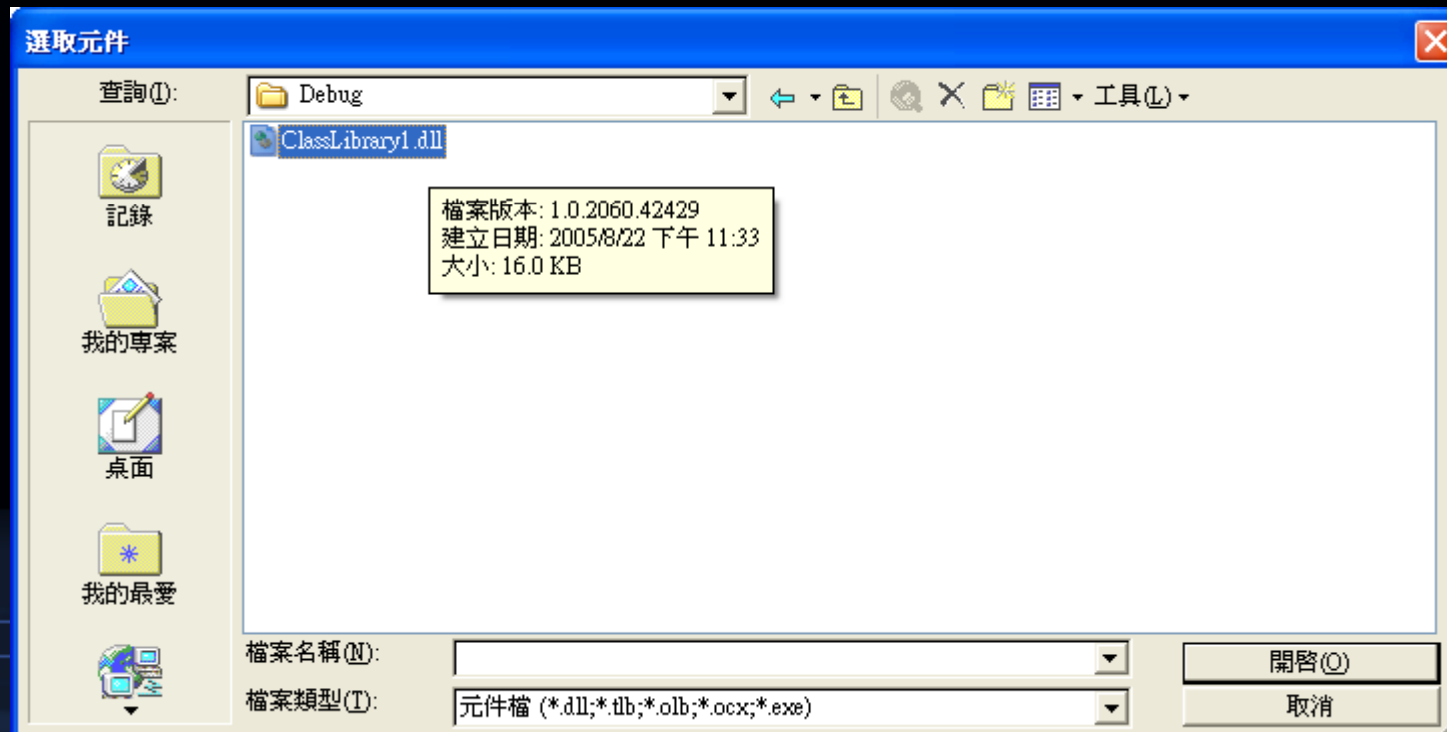
引用外部類別 (4)

- 透過瀏覽按鈕加入該類別庫組件



引用外部類別 (5)

- 選取組件檔



實例探討 sample5-a6 (1)

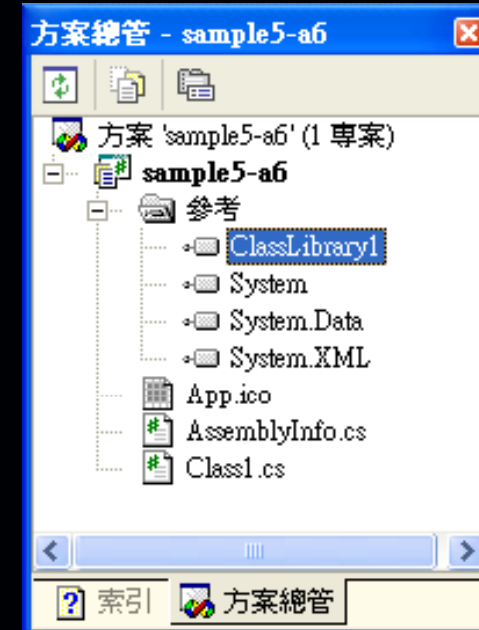
- 程式功能
 - 以類別庫建立一類別包含二數相加靜態方法
 - 在專案中引用該類別庫呼叫該相加運算方法
- 程式內容 – **ClassLibrary1**

```
namespace ClassLibrary1
{
    public class myClass
    {
        public static int Add(int a, int b)
        {
            return a+b;
        }
    }
}
```

實例探討 sample5-a6 (2)

- 程式內容 - sample5-a6

```
using System;
using ClassLibrary1;
namespace sample5_a6
{
    class Class1
    {
        static void Main(string[] args)
        {
            Console.WriteLine("2+3=" + myClass.Add(2,3));
            Console.ReadLine();
        }
    }
}
```



課堂練習 sample5-b5

- 程式功能 - 類別庫
 - 以類別庫建立一類別包含一靜態方法
 - 該靜態方法可傳入一字串回傳一布林型別
 - 此靜態方法可檢查傳入字串是否為合法 **E-mail** 帳號
- 程式功能 - 主程式
 - 引用上述類別庫
 - 請使用者輸入 **E-mail** 帳號
 - 列印輸入結果是否正確

課堂練習 sample5-b5

- 基本概念
 - 假設合法 **E-mail** 帳號需包含 @ 符號並且 @ 符號後至少需有一個 . 符號
 - 可利用 **string** 類別中的 **IndexOf** 方法來檢查

Outline

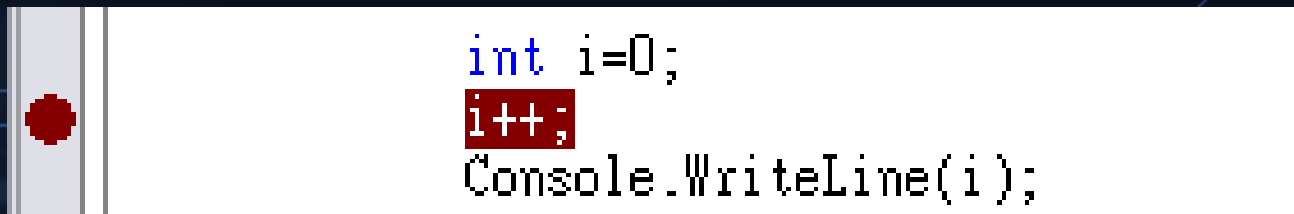
- ArrayList 集合類別
- 陣列類別
- 字串類別
- 其它常用類別
- 引用外部類別
- 程式偵錯
- 例外處理

程式偵錯

- 語法錯誤
 - 程式的語法撰寫不正錯，編譯時即會指出錯誤的地方
- 邏輯錯誤
 - 程式撰寫邏輯觀念有誤，也是最難察覺的錯誤，可利用偵錯工具來協助找出錯誤之處
- 執行時期錯誤
 - 使用者輸入的資料型別不符、計算過程分母為 0、磁碟中無指定檔案等執行時期錯誤，可利用 **try catch** 敘述來解決

中斷 (1)

- 中斷
 - 當程式執行至中斷點處即會暫停，此時可藉由其它工具來查詢各變數目前數值
- 設定中斷點
 - 將滑鼠移到欲設置中斷點程式碼左方，按下滑鼠左鍵即可設定中斷點

A screenshot of a code editor with a white background. On the left side, there is a vertical toolbar with a red circular icon containing a black dot, representing a breakpoint. To the right of this icon, the following C# code is displayed:

```
int i=0;  
i++;  
Console.WriteLine(i);
```

The line `i++;` is highlighted with a red background, indicating that a breakpoint has been set at this line of code.

中斷 (2)

- 當程式執行至中斷點時，我們可以逐步一行行執行程式觀察每個變數數值的變化
 - 逐步執行
 - 每按一次 [逐步執行] 執行一行程式碼，會執行函式內程式碼
 - 不進入函式
 - 每按一次 [不進入函式] 執行一行程式碼，但不會執行函式內部的程式

中斷 (3)

- 當程式執行至中斷點處，可以開啟 [偵錯] [視窗] 中的 [自動變數] [區域變數] [監看式] 視窗來觀察變數值的變化
 - 自動變數
 - 顯示程式目前執行行號附近的變數
 - 區域變數
 - 顯示程式目前執行區塊所宣告的變數
 - 監看式
 - 自訂想要觀看的變數

實例探討 sample5-a7

- 程式功能
 - 觀察變數值、中斷測試

- 程式內容

```
x = 10;
```

```
int c = 0;
```

```
for(int i=0; i<10; i++)
```

```
    c+=i;
```

```
c += x;
```

```
ArrayList myAL = new ArrayList();
```

```
myAL.Add(c);
```

```
x = (int)myAL[0];
```

Outline

- ArrayList 集合類別
- 陣列類別
- 字串類別
- 其它常用類別
- 引用外部類別
- 程式偵錯
- 例外處理

例外處理 (1)

- 程式在執行時，經常會因為使用的操作或環境因素而出現未預期的錯誤
 - 讀取軟碟，但軟碟機尚未放置磁片
- 我們可以使用 **try catch finally** 來捕捉未預期的錯誤
- 當執行至 **try** 程式區塊內發生錯誤時，會逐一檢查該錯誤符合那個 **catch**，以便執行該 **catch** 內所撰寫的錯誤處理程式
- 不管有沒有符合的 **catch**，最後都會執行 **finally** 內的程式區塊

例外處理 (2)

- 所有的例外類別都是衍生自內建的 **Exception** 例外類別
- 常用的例外類別

類別	說明
Exception	執行時期產生錯誤
DivideByZeroException	除數為 0
IndexOutOfRangeException	索引值超出陣列範圍
InvalidCastException	型別轉換錯誤
OverflowException	溢位

- 常用屬性
 - **Message** – 取得例外的描述訊息

例外處理 (3)

- 語法

```
try
{
    程式區塊;
}
catch (例外類別 變數名)
{
    程式區塊;
}
catch (例外類別 變數名)
{
    程式區塊;
}
finally
{
    程式區塊;
}
```

例外處理 (4)

- 用法

```
try
{
    int x = 0;
    x = 3/x;
}
catch
{
    Console.WriteLine("發生錯誤!!");
}
finally
{
    Console.WriteLine("結束");
}
```


實例探討 sample5-a8 (1)

- 程式功能
 - 請使用者輸入一個數 x
 - 計算 3 除以 x 的值
- 程式內容

```
try
{
    Console.Write("請輸入一個數：");
    int x = int.Parse(Console.ReadLine());
    Console.WriteLine("計算結果為：" + (3/x));
}
```

實例探討 sample5-a8 (2)

- 程式內容

```
Catch( Exception e )  
{  
    Console.WriteLine( e.Message+"發生錯誤!!");  
}  
finally  
{  
    Console.WriteLine("結束");  
}
```

課堂練習 sample5-b6

- 程式功能
 - 請使用者輸入一個數字 x
 - 計算 x 的平方值
 - 輸入字串為文字時列印錯誤訊息
- 基本概念
 - 藉由例外處理來預防使用者輸入文字字串

自訂例外處理

- 我們可以藉由 **throw** 敘述來自訂特殊的錯誤
- **throw** 可指定所使用的例外類別
- 語法
 - **throw new 例外類別();**
- 用法
 - **throw new Exception("error!!");**

實例探討 sample5-a9 (1)

- 程式功能
 - 請使用者輸入月份
 - 月份輸入錯誤產生例外
- 程式內容

```
static int getMonth()
{
    Console.Write("請輸入月份：");
    int month = int.Parse(Console.ReadLine());

    if(month > 12 || month <= 0)
        throw new Exception("您輸入的月份不正確!!");
    else
        return month;
}
```

實例探討 sample5-a9 (2)

- 程式內容

```
static void Main(string[] args)
{
    try
    {
        Console.WriteLine("您輸入的月份爲：" + getMonth());
    }
    catch(Exception e)
    {
        Console.WriteLine(e.Message);
    }
    finally
    {
        Console.ReadLine();
    }
}
```