第二章 主控台應用程式與資料型別

- 2.1 C# 程式架構
- 2-2 C# 資料型別
- 2.3 C# 運算子
- 2.4 主控台應用程式
- 2.5 主控台應用程式格式化輸出入
- 2.6 資料型別轉換
- 2.7 列舉資料型別
- 2.8 結構資料型別

2.1 C# 程式架構

- 主控台應用程式的架構

```
01 using System;
02 using System.Collections.Generic;
03 using System.Linq;
04 using System.Text;
05 using System.Threading.Tasks;
06
07 namespace ConsoleApplication1
08 {
09     class Program
10     {
11         static void Main(string[] args)
12         {
13             // 在主控台印出 "歡迎光臨 C# 的世界" 訊息
14             Console.WriteLine("歡迎光臨 C# 的世界");
15             Console.Read();
16         }
17     }
18 }
```

程式架構說明

1. **Using 指示詞**

   System.Console.WriteLine ("歡迎光臨 C# 的世界")

2. **namespace ConsoleApplication1**

```
namespace 命名空間名稱
{
    類別, 介面, 結構, 列舉, 委派等型別定義於此處
}
```

在上面命名空間範圍內，也可再包含另一個namespace (命名空間)、class(類別)、interface(介面)、struct(結構)、enum(列舉)、delegate(委派)。

下圖是一個包含所有上面項目的 C# 程式基本架構：

程式架構說明

```
namespace MyNamespace                    // 命名空間名稱為 MyNamespace
{
  class MyClass                          // 定義名稱為 MyClass 的類別
  {

  }
  struct MyStruct                        // 定義名稱為 MyStruct 的結構
  {

  }
  interface MyInterface                  // 定義名稱為 MyInterface 的介面
  {

  }

  delegate int MyDelgate();              // 宣告名稱為 MyDelgate 的委派

  enum MyEnum                            // 定義名稱為 MyEnum 的列舉
  {

  }
  class Program                          // 定義名稱為 Program 類別
  {
    public static void Main(string[] args)      // Main 方法為程式開始執行的起點
    {

    }
  }
}
```

程式架構說明

3. 註解符號

// 在主控台印出 "歡迎光臨 C# 的世界 “

```
/*
   在主控台印出 "歡迎光臨 C# 的世界" 訊息
*/
```

4. class Program

- 第9行使用 class 關鍵字定義名稱為 Program 的類別。為 C# 預設類別名稱
- 依需求將 Program更名為較意義的類別名稱

程式架構說明

5. Main()方法

- 程式開始執行進入點。
- C# 程式允許由多個類別組成。
- 多個類別中只允許有一個類別內含有 Main()方法。
- 程式執行時會將此 Main() 方法視為程式開始執行的進入點。

```
class Program
{
   static void Main(string[] args)  ⇦  程式開始執行的進入點
   {
      // 敘述區段;
   }
}
```

程式架構說明

① static

一般類別中所定義的方法必須先建立該類別的物件實體(簡稱物件)後才能使用該物件的方法。C# 預設在 Main()方法前面加上 static 主要是希望不用先建立 Main()方法的物件實體，在執行階段(RunTime)就能直接叫用。若未加上 static，在執行階段就必須先建立該類別的物件實體後才能呼叫。

② void

Main()方法前面加上 void 表示此方法不會傳回任何值。

③ string[] args 引數

表示 args 是屬於 string 資料型別的一個陣列物件(string 為字串類別)。它代表執行 Main()方法時會將接在專案執行檔後面的參數置入 args 字串陣列。

程式架構說明

6. Console.WriteLine() 和 Console.Read() 方法

- .NET Framework 類別程式庫中屬於 Console 類別的一種輸出入方法。

2.2 C# 資料型別

2.2.1 識別字(Identifier)

- 由一個或多個字元組成。用來對程式中的一個方法、變數或其他使用者定義的項目給予名稱，以便程式中識別。

- **識別字命名規則：**

1. 識別字名稱以 A-Z、a-z 或(底線)等字元開頭，不允許以數字開頭。第1個字元後可接大小寫字母、數字、底線。
2. 識別字將字母大小寫視為不同字元。
3. 識別字命名具意義、名稱和資料關連，可讀性高且易記。
4. 允許用中文字當變數名稱，程式中易混淆，建議不用為宜。
5. 關鍵字不允許當識別字。

系統保留字或稱關鍵字

abstract	as	base	bool	break
byte	case	catch	char	checked
class	const	continue	decimal	default
delegate	do	double	else	enum
event	explicit	extern	false	finally
fixed	float	for	foreach	goto
if	implicit	in	int	interface
internal	is	lock	long	namespace
new	null	object	operator	out
override	params	private	protected	public
readonly	ref	return	sbyte	sealed
short	sizeof	stackalloc	static	string
struct	switch	this	throw	true
try	typeof	uint	ulong	unchecked
unsafe	ushort	using	virtual	volatile
void	while			

2.2.2 C# 的基本資料型別

資料型別	.NET Framework	說明
bool 布林	System.Boolean	佔 1-Byte 布林值，其值為 true 或 false。
byte 8 位元無號整數	SystemByte	佔 1-Byte，無正負整數。 範圍：0~255
sbyte 8 位元有號整數	System.Sbyte	佔 1-Byte，有正負符號整數。 範圍：-128 至 127
short 短整數	System.Int16	佔 2-Bytes，有正負符號整數。 範圍：-32,768~+32,767
int 整數	System.Int32	佔 4-Bytes，有正負號整數。 範圍：-2,147,483,648 至 +2,147,483,647
long 長整數	System.Int64	佔 8-Bytes，有正負號整數。範圍： -9,223,372,036,854,775,808 至 +9,223,372,036,854,775,807
ushort 無號短整數	System.Uint16	佔 2-Bytes，無正負號整數 範圍：0~65,535
uint 無號整數	System.Uint32	佔 4-Bytes，無正負號整數 範圍：0~4,294,967,295

資料型別	.NET Framework	說明
ulong 無號長整數	System.Uint64	佔 8-Bytes，無正負號整數 範圍：0 ~ 18,446,744,073,709,551,615
float 浮點數	System.Single	佔 4-Bytes，單精確度浮點數字。 範圍：1.5x10-45 至 3.4x1037 精確度 7 個數字
double 倍精確數	System.Double	佔 8-Bytes，倍精確度浮點數字。 範圍：5.0x10-324 至 1.7x10308 精確度 15-16 個數字
decimal 貨幣	System.Decimal	佔 16 Bytes 為十進位數字，有效位數 28。 範圍：1.0x10-28~7.9x1028
char 字元	System.Char	佔 2-Bytes 字元，其值是一個 Unicode 的字元，該字元以單引號括住。範圍：0~65,535。
string 字串	System.String	字串型別，資料頭尾以雙引號括住
object 物件	System.Object	物件型別，可以存放任意資料型別的資料
使用者定義變數	結構/類別/列舉	成員大小長度的總和。

2.2.3 常值、常數和變數的關係

- 常值 (Literal) 表示本身的值，而非變數值或運算式的結果。
  例如：號碼 5 或字串 “Hello”。
  將程式中使用資料的常值，依程式執行時是否
  允許改變其值分成 常數 和 變數 兩種。
- 常數(Constant)是以有意義的名稱取代常值，在整個程式執行中其值維持不變。
- 變數 (Variable)是以意義的名稱取代常值，允許在整個程式執行中變更其值。
- C# 提供的常值包括：數值常值、字串常值、日期常值、布林常值、物件常值。

2.2.4 常數 (Constant)

- 使用時機
  程式執行時，有些值到程式結束前都一直保持不變且重複出現，為方便在程式中辨識，可使用一個有意義的 常數名稱 來取代這些不變的數字或字串常值。
- 譬如：稅率、圓周率、或常用的字串、日期。
- 程式執行過程中
  ⇨ 變數隨時因敘述指定常值而更改變數值。
  ⇨ 常數經宣告，在整個程式中一直保有宣告時所指定的常數值。
- 常數名稱
  是用 const 來宣告，在宣告同時即指定一個常值。
  程式中使用常數名稱可增加程式可讀性和易修改。

```
const  double  PI = 3.14 ;
int r ;
r = 10 ;
label1.Text = "2 * PI * r = " + (2* PI * r).ToString() ;
```

例 下例示範將常值加入適當型別字元成為常數的正確用法：

1. 常值預設為整數常數

   ```
   public const int DefaultInteger  = 100;
   ```

2. 常值預設為倍精確常數

   ```
   public const double  DefaultDouble  = 54.3345612;
   ```

3. 常值強制為長整數常數 (採附加型別字元 L 或 l)

   ```
   public const long MyLong  = 45L;
   ```

4. 常值強制為單精確度常數(採附加型別字元 F 或 f)

   ```
   public const float MySingle = 45.55F;
   ```

2.2.5 變數

一. 變數的宣告

語法

資料型別 變數名稱；

資料型別 變數名稱 1，變數名稱 2 ，..... ；

宣告 var1 和 var2 為整數變數，str1 為字串變數，寫法：

```
int var1, var2;
string str1;
```

二. 變數的初始化

- 初始化(Initialize)
  - ⇨ 就是給予變數初值。
  - ⇨ C# 是不允許使用未初始化過的變數
  - ⇨ 變數經宣告後須再經初始化才能在程式中使用。
  - ⇨ 初始化就是使用指定運算子來設定變數的初值。

```
int score;
score = 90;
```

合併成一行 `int score = 90 ;`

宣告變數時也可同時指定變數初值，寫法：

```
int score= 100;          // 宣告 score 為整數變數代表成績初值為 100
double price = 90.5;     // 宣告 bcc 為浮點變數單價初值為 90.5
char sex = 'M';          // 宣告 sex 為字元變數代表性別初值為 'M'
char c1 = '\x0061';      // 宣告 c1 為字元變數，以 a 字元的 ASCII 碼表示 61(16)
char c2 = (char)97;      // 宣告 c2 為字元變數，以 a 字元的 ASCII 碼表示 97(10)
char c3 = '\u0061';      // 宣告 c2 為字元變數，以 a 字元的 UniCode 表示
```

字元和字串

- 字元(character)資料
  在程式中以單引號將字元頭尾括起來。如：‘A’、‘B’、‘1’、‘2’等。
- 字串(string) 資料
  一個以上字元合併形成字串資料。
  字串資料在程式中須用 雙引號 將字串頭尾括起來。

1. 字串型別變數宣告及初值設定：

```
string str1 ;
str1 = "Visual C# 2013 程式設計經典";
string name = "史瑞克 4" ;
```

2. 字串型別的運算子

字串變數的資料可以做一些簡單的運算，如下表：

運算子	說明	例子
=	指定字串給字串變數	string str1= "C#程式設計" ;
+	兩個字串的合併	string str2= "VB"+ "2013" ;

```
string str1, str2, str3 ;
str1 = "史瑞克 4！" ;
str2 = "真好看～" ;
str3 = str1 + str2 ;        結果： str3="史瑞克 4！真好看～"
```

2.2.6 實質型別與參考型別

- C# 依資料在記憶體存放管理機制的不同分成：

  1. 實值型別 (Value types)
  2. 參考型別 (Reference types)

一. 實質型別 (Value Type)

- 實值型別的執行個體存放在稱為堆疊區的記憶體中。
- 實值型別資料名稱的記憶位址內存放是資料本身。
- 實值型別的資料在程式執行階段可快速建立、存取和移除該執行個體。
- 由於存取此型別的資料時，記憶體中所記錄的是真正的資料，所以稱為實質型別。
- C# 提供的內建資料型別包含數值型別、結構型別及可為 null 的型別都屬實值型別。

一. 實質型別 (Value Type)

- 所有實值型別都是自 System.Value.Type 隱含衍生而來。
- 實值型別變數直接包含其值
  表示在宣告任何內容的變數時，便會內嵌(Inline)配置記憶體。
- 系統沒對實值型別變數進行個別的堆積(Heap)配置或負荷記憶體回收。
- 實質型別包括：
  內建(基本和一般)型別、結構(使用者定義型別)struct、列舉 enum三種。其中 struct 是在堆疊區存放自己的資料。

C# 提供的內建(基本和一般)型別

內建型別	.NET別名	C#別名	位元組	數值範圍
System.SByte	SByte	sbyte	1	-128 ~ 127
System.Byte	Byte	byte	1	0 ~ 255
System.Int16	Short	short	2	-32768 ~ +32767
System.Int32	Integer	int	4	-2147483648 ~ +2147483647
System.UInt32	UInteger	uint	4	0 ~ 4294967295
System.Int64	Long	long	8	-9223372036854775808 ~ +9223372036854775807
System.Single	Single	float	4	-3.402823E+38 ~ +3.402823E+38
System.Double	Double	double	8	-1.79769313486232E+308 ~ +1.79769313486232E+308
System.Decimal	Decimal	decimal	16	-79228162514264337593543950335 ~ +79228162514264337593543950335
System.Char	Char	char	2	單一 Unicode 字元
System.Boolean	Boolean	bool	4	true/false
System.DateTime	Date	date	8	1/1/0001 12:00:00 AM ~ 12/31/9999 11:59:59 PM

二. 參考型別 (Reference Type)

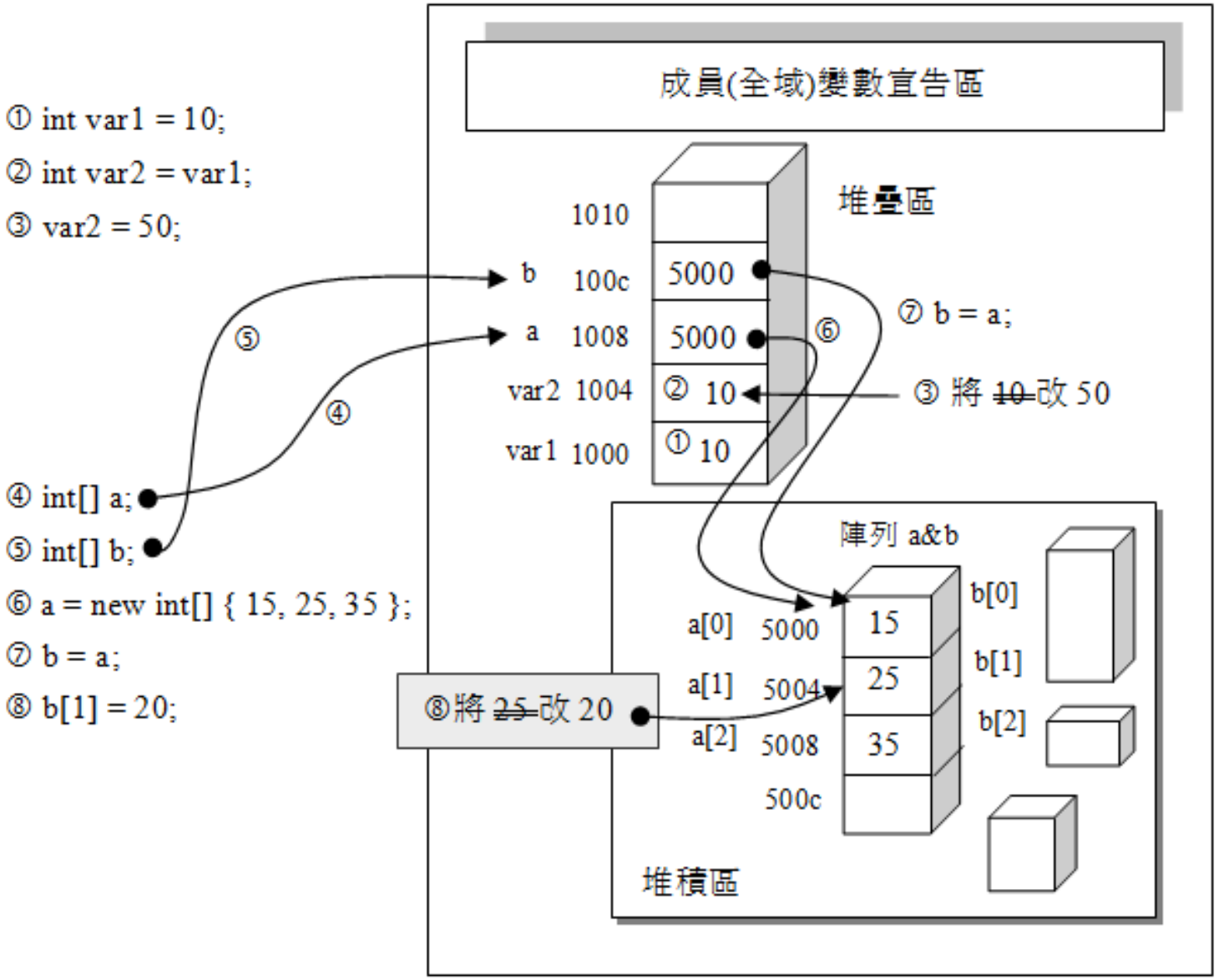
- 參考型別的執行個體(資料)
  是存放在一塊稱為堆積區(Heap)的記憶體中。
- 參考型別的資料名稱非資料本身
  ⇨ 裡面所存放的是資料的位址(即資料的指標)
  ⇨ 用來告知真正資料是存放在堆積記憶體區塊內哪個位址上。
- 參考型別是配置在堆積上的資料型別
  ⇨ 宣告為參考型別的變數會指向儲存資料的位置。

二. 參考型別 (Reference Type) Continue…

- 將變數宣告為參考型別時
  變數在執行階段的一開始會包含 null 值，直到用
  new 運算子明確建立物件執行個體，或其指派
  使用 new 在其他地方所建立的物件為止。
- 參考型別占用記憶體是堆積區
  此塊記憶體由Common Language RunTime動態配置。
- 參考型別包含：
  字串型別及定義為類別、委派、陣列或介面的型別
  都屬參考型別。
- .NET Framework 內的物件，都屬參考型別
- 字串屬參考型別，對字串的任何變動，都會在執行階段
  去產生一個變動後的新字串。

三. 堆疊(Stack)與推積(Heap)

- 記憶體中的堆疊可想像成是一個由下而上疊起來的盒子
- 實值型別的資料就依宣告次序一個一個由下而上依序存放在盒子內。
- 當資料的生命週期結束時將存放該資料的盒子移掉。
- 記憶體中的堆積就配置一個空間給其使用，裡面有一堆盒子亂七八糟擺放，若盒子上可標明該盒子目前是誰在使用，也可多個參考型別的資料共同使用同一個盒子。


① int var1 = 10;
② int var2 = var1;
③ var2 = 50;
④ int[] a;
⑤ int[] b;
⑥ a = new int[] { 15, 25, 35 };
⑦ b = a;
⑧ b[1] = 20;
成員(全域)變數宣告區
堆疊區
1010
b 100c 5000
a 1008 5000
var2 1004 ② 10
var1 1000 ① 10
⑦ b = a;
⑥
③ 將 10 改 50
⑤
④
陣列 a&b
a[0] 5000
a[1] 5004
a[2] 5008
500c
15
25
35
b[0]
b[1]
b[2]
⑧將 25 改 20
堆積區

2.3 C# 運算子

運算子(Operator)是用來指定資料做何種運算。運算子按照運算時所需要的運算元(Operand)數目分成：

1. 單元運算子(Unary Operator) 如：-5、k++。
2. 二元運算子(Binary Operator) 如：a+b。
3. 三元運算子(Tenary Operator)

   如：max = (a>b) ? a：b，

   表示若 a>b 為真，則 max=a ；否則 max = b。

2.3.1 算術運算子

運算子符號	運算子	運算式	若 j=20 k=3 下列 i 結果
+	相加運算子	i = j + k	i ⇦ 23 (20+3)
-	相減運算子	i = j – k	i ⇦ 17 (20-3)
*	相乘運算子	i = j * k	i ⇦ 60 (20*3)
/	相除運算子	i = j / k	i ⇦ 6 (20/3，整數相除結果取整數) 若 j = 20.0 k = 3.0 (i=j/k 則 i=6.666666666666667)
%	取餘數運算子	i = j % k	i ⇦ 2 (20 % 3)

```
int i, j, k;
i = 16;
j= i / 3;
k= i % 3;
```

2.3.2 關係運算子

關係運算子	意義	數學式	關係運算式
==	相等	A=B	A==B
!=	不相等	A≠B	A!=B
>	大於	A>B	A>B
<	小於	A<B	A<B
>=	大於或等於	A≧B	A>=B
<=	小於或等於	A≦B	A<=B

2.3.3 邏輯運算式

邏輯運算子	意義	邏輯運算式	用法
&&	且(And)	A && B	當 A、B 皆為真時，結果才為真。
||	或(Or)	A || B	若 A、B 其中只要有一個為真，結果為真。
!	非(Not 反相)	! A	若 A 為真，結果為假； 若 A 為假，結果為真。

A	B	A && B	A || B	! A
true	true	true	true	false
true	false	false	true	false
false	true	false	true	true
false	false	false	false	true

【例 1】 ① (4 > 3) && ('a'=='b') ⇨ 真 && 假 ⇨ false(假)

② (4 > 3) || ('a'=='b') ⇨ 真 || 假 ⇨ true(真)

③ ! (4 > 3) ⇨ !(true) ⇨ 假 ⇨ false(假)

【例 2】 年齡介於 20 ≦age < 60 之間的條件式：

(age >= 20) && (age < 60)

【例 3】 score 不為零的條件式：

score != 0

2.3.4 位元運算子

A	B	A & B	A | B	A^B	~A
1	1	1	1	0	0
1	0	0	1	1	0
0	1	0	1	1	1
0	0	0	0	0	1

1. 5 & 3 ⇨ 結果為 1，運算過程如下圖：

```
  0101  ←5 的二進位
& 0011  ←3 的二進位
------
  0001  ←1 的二進位
```

2. 5 | 3 ⇨ 結果為 7，運算過程如下圖：

```
  0101  ←5 的二進位
| 0011  ←3 的二進位
------
  0111  ←7 的二進位
```

3. 5 ^ 3 ⇨ 結果為 6，運算過程如下圖：

```
  0101  ←5 的二進位
^ 0011  ←3 的二進位
------
  0110  ←6 的二進位
```

4. ~5 ⇨ 結果為-6，運算過程如下圖：

```
~ 0101  ←5 的二進位
------
  1010  ←-6 的二進位
```

2.3.5 移位運算子

移位運算子可用來做數值運算，做法是先將指定的運算元轉成二進位，接著再使用 ">>" 右移運算子指定該運算元往右移幾個位元(bit)，或是使用 "<<" 左移運算子指定該運算元往左移幾個位元(bit)。例如：$(20)_{10} = (0010100)_2$

1. 20_{10}>>1 ⇨ 20_{10} 右移 1 個 bit ⇨ 00010100_2>>1 ⇨ 000010102 = 10_{10}
2. 20_{10}<<1 ⇨ 20_{10} 左移 1 個 bit ⇨ 00010100_2<<1 ⇨ 001010002 = 40_{10}

2.3.6 複合指定運算子

運算子符號	意義	實例
=	指定	i = 5;
+=	相加後再指定	i += 5;
-=	相減後再指定	i -= 5;
*=	相乘後再指定	i *= 5;
/=	相除後再指定	i /= 5;
%=	餘數除法後再指定	i %= 5;
^=	作位元的 XOR 運算	i ^= 5;
&=	作位元的 AND 運算後再指定	i &= 5;
|=	作位元 OR 運算後再指定	i |= 5;
<<=	左移指定運算	i <<= 5
>>=	右移指定運算	i >>= 5

2.3.7 遞增及遞減運算子

- ++ 遞增和 --遞減運算子兩者都屬單元運算子。
- i = i + 1 ⇨ i++ ;

  i = i – 1 ⇨ i - - ;
- 將遞增/遞減運算子
  放在變數之前 表前遞增；放在之後表 後遞增；

運算子符號	意義	實例
!	邏輯 NOT 運算	if(!true)
++	遞增運算子	i++; (與 i=i+1 同)
--	遞減運算子	i--; (與 i=i-1 同)

【例 1】

```
int i = 10,k ;
k = i++ ;
```

【例 2】

```
int i = 10, k ;
k = ++i ;
```

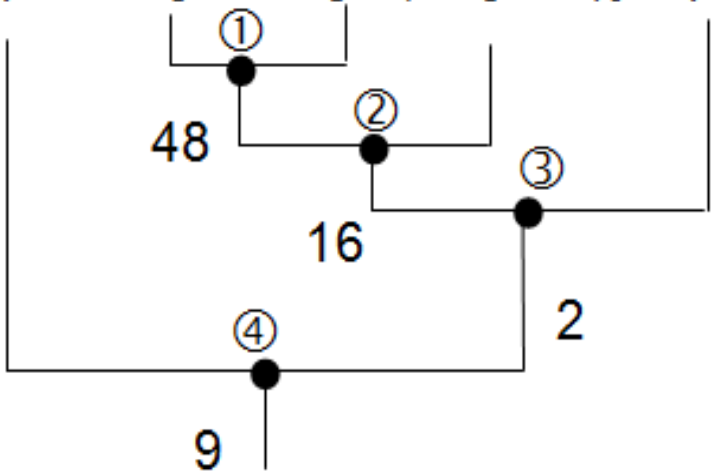
【例 3】

```
int i, j, k ;
i = j = 10 ;
k = ++i*5 ;
j = k++*2 ;
```

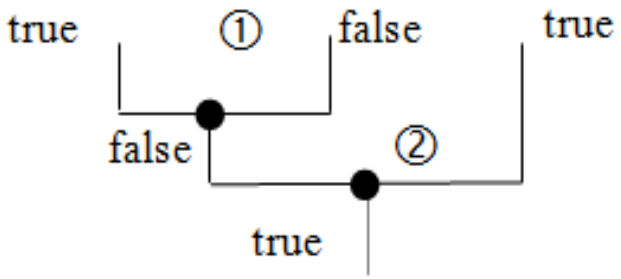
2.3.8 運算子的優先順序

優先次序	運算子(Operator)	運算次序
1	x.y、f(x)、a[x]、x++、x--、new、 、(括號)	由內至外
2	! 、~、(cast)、+(正號)、-(負號)、++x、--x	由內至外
3	*(乘)、 / (除)、 %(取餘數)	由左至右
4	+(加)、 -(減)	由左至右
5	<< (左移) 、 >> (右移)	由左至右
6	< 、 <=、 >、 >= (關係運算子)	由左至右
7	== (相等) 、 != (不等於)	由左至右
8	& (邏輯 AND)	由左至右
9	^ (邏輯 XOR)	由左至右
10	| (邏輯 OR)	由左至右
11	&&(條件式 AND)	由左至右
12	|| (條件式 OR)	由左至右
13	?: (條件運算子)	由右至左
14	=、+=、-=、*=、/=、%=、<<=、>>=、&=、^=、!=	由右而左
15	, (逗號)	由左至右

【例 1】7 + 6 * 8 / 3 % 7 = 9



【例 2】 A && B || C (假設 A 和 C 為 true，B 為 false)



2.4 主控台應用程式

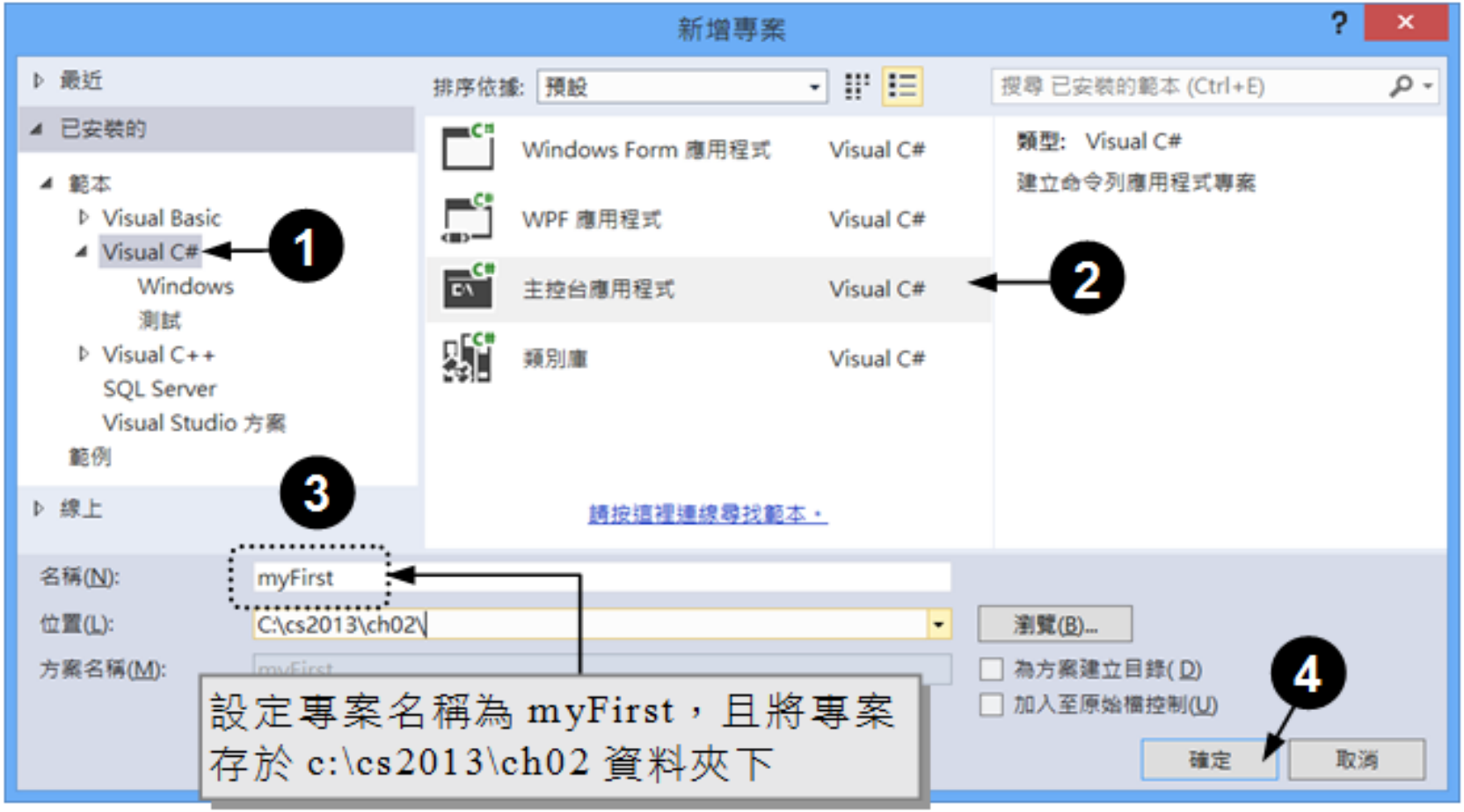
2.4.1 主控台應用程式的撰寫與執行

- Console 是系統命名空間(Namespace)內所定義類別之一。
- 主要用來處理有關在主控台應用程式(Console Application)模式下的輸入、輸出以及錯誤資料串流(Streams)。

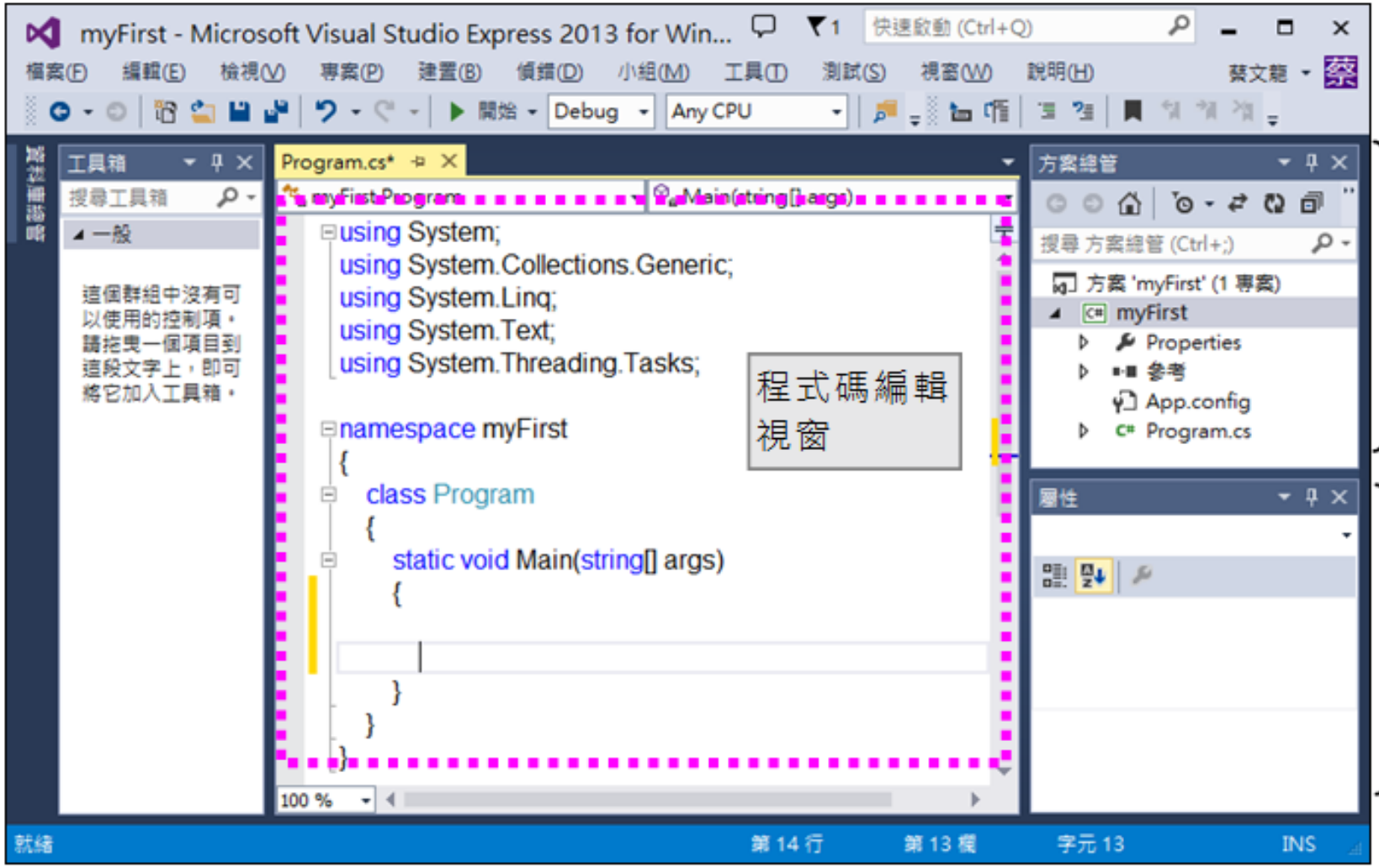
進入整合開發環境


起始頁 - Microsoft Visual Studio Express 2013 for Wind...
快速啟動 (Ctrl+Q)
檔案(F) 編輯(E) 檢視(V) 偵錯(D) 小組(M) 工具(T) 測試(S) 視窗(W) 說明(H)
蔡文龍
附加...
起始頁
工具箱
伺服器總管
Express 2013 for Windows Desktop
開始
新增專案...
開啟專案...
從原始檔控制開啟...
最近
WpfApplication1
DBAp2
DBAp3
DBAp2
探索 Express 2013 for Windows Desktop 的新功能
檢閱下列區段，就可以找到 Express 2013 for Windows Desktop 中各項新增功能和增強功能的相關資訊。
了解 Express 2013 for Windows Desktop 的新功能
查看 NET Framework 4.5.1 的新功能
瀏覽 Team Foundation Service 的新功能
重新配置新功能資訊
Windows 桌面的新功能
Visual C#
Visual Basic
Visual C++
Windows Presentation Foundation (WPF)
Windows Form
就緒

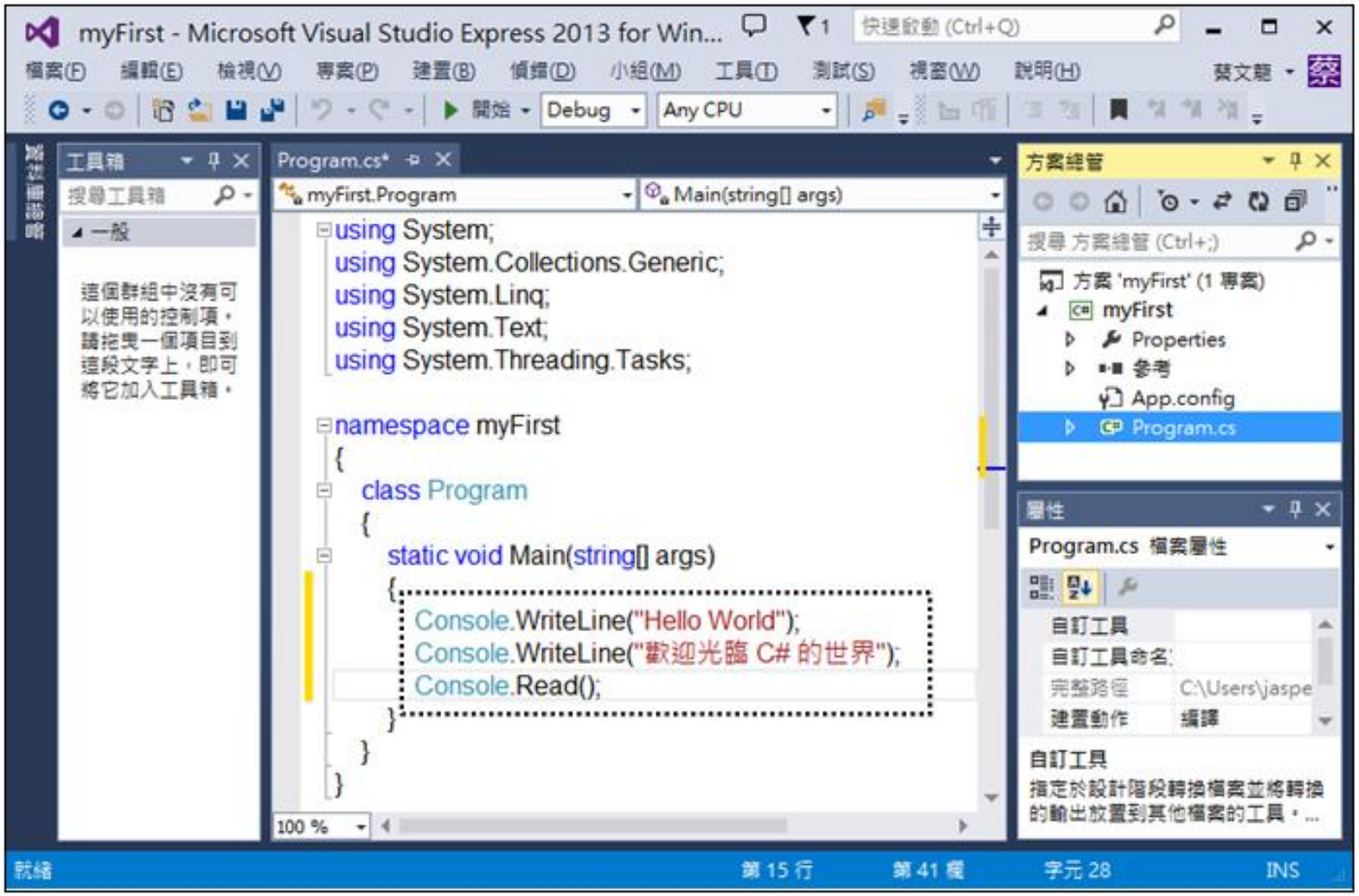
在Console模式下建立名稱為myfirst 專案

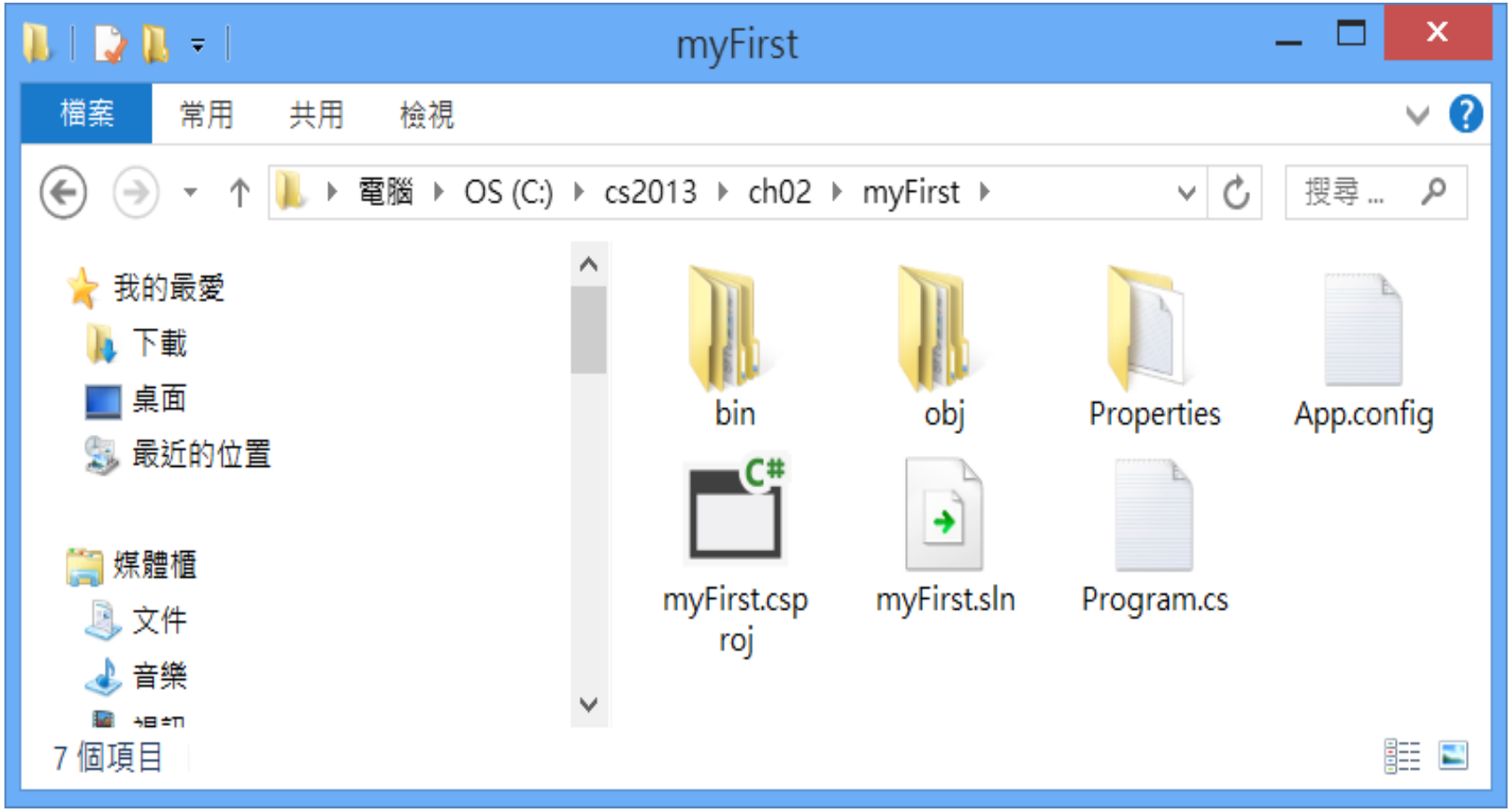


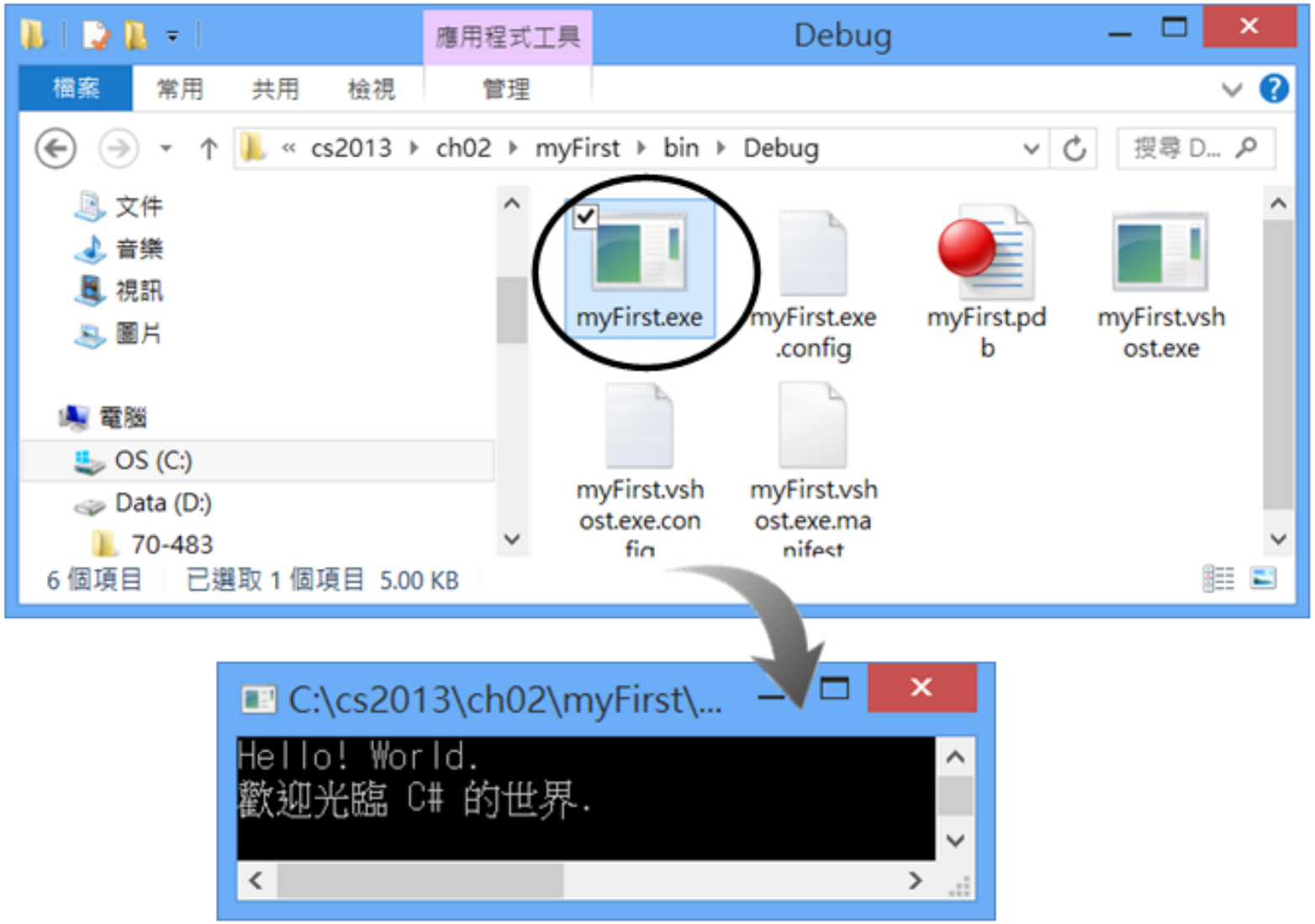
程式編碼視窗


myFirst - Microsoft Visual Studio Express 2013 for Win...
檔案(F) 編輯(E) 檢視(V) 專案(P) 建置(B) 偵錯(D) 小組(M) 工具(T) 測試(S) 視窗(W) 說明(H)
Debug
Any CPU
工具箱
Program.cs*
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;
namespace myFirst
{
class Program
{
static void Main(string[] args)
{
}
}
}
程式碼編輯視窗
方案總管
方案 'myFirst' (1 專案)
myFirst
Properties
參考
App.config
Program.cs
屬性
方案總管
屬性視窗
就緒
第 14 行
第 13 欄
字元 13
INS

撰寫相關程式碼


myFirst - Microsoft Visual Studio Express 2013 for Win...
Debug
Any CPU
Program.cs*
myFirst.Program
Main(string[] args)
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;
namespace myFirst
{
class Program
{
static void Main(string[] args)
{
Console.WriteLine("Hello World");
Console.WriteLine("歡迎光臨 C# 的世界");
Console.Read();
}
}
}
工具箱
方案總管
myFirst
Properties
App.config
Program.cs
就緒


myFirst
檔案
常用
共用
檢視
電腦
OS (C:)
cs2013
ch02
myFirst
搜尋 ...
我的最愛
下載
桌面
最近的位置
媒體櫃
文件
音樂
bin
obj
Properties
App.config
myFirst.csproj
myFirst.sln
Program.cs
7 個項目


應用程式工具
Debug
檔案
常用
共用
檢視
管理
cs2013 ▸ ch02 ▸ myFirst ▸ bin ▸ Debug
搜尋 D...
文件
音樂
視訊
圖片
電腦
OS (C:)
Data (D:)
70-483
myFirst.exe
myFirst.exe.config
myFirst.pdb
myFirst.vshost.exe
myFirst.vshost.exe.config
myFirst.vshost.exe.manifest
6 個項目
已選取 1 個項目 5.00 KB
C:\cs2013\ch02\myFirst\...
Hello! World.
歡迎光臨 C# 的世界.

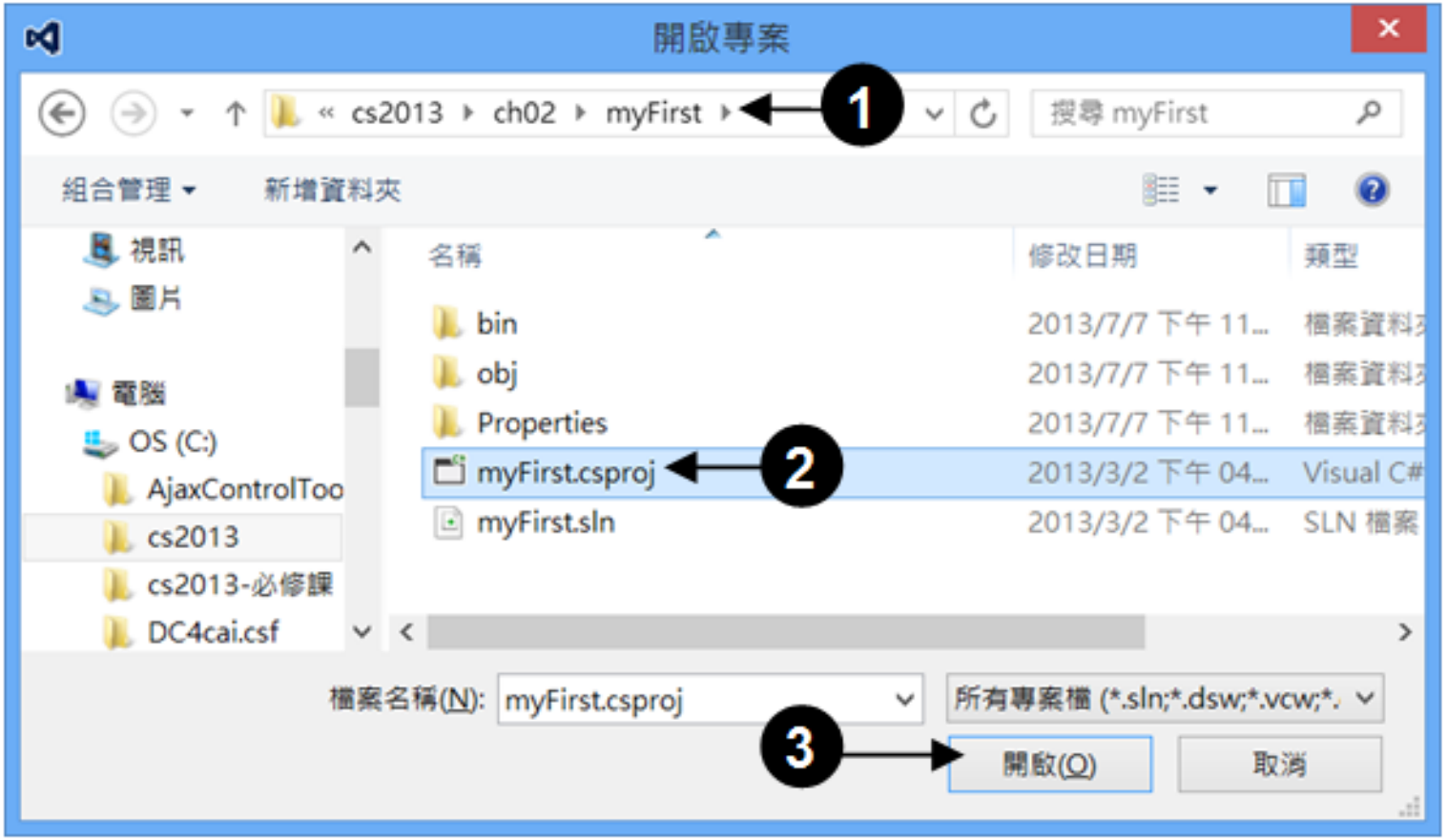
產生的相關檔案

相關檔案	說明
myFirst.sln	為方案檔(位於 ch02\myFirst 資料夾下)，副檔名為 .sln。
myFirst.csproj	為專案檔(位於 ch02\myFirst 資料夾下)，副檔名為 .csproj。
Program.cs	為主控台預設的 C# 程式檔(位於 ch02\myFirst 資料夾下)，副檔名為 .cs。
bin/Debug/ myFirst.exe	當專案編譯後，在 ch02\myFirst\bin\Debug 資料夾下會產生和專案同名的 .exe 執行檔。

2.4.2 關閉專案檔

- 執行功能表的【檔案(F)/全部儲存(L)】指令
  將目前專案內的相關檔案進行儲存的動作。
- 若不離開整合開發環境，繼續編輯另外專案時
  執行功能表的【檔案(F)/關閉方案(T)】指令來關閉目前的專案或方案。
- 若要離開 C# 整合開發環境
  執行功能表的【檔案(F)/結束(X)】指令或
  按 關閉鈕離開 C# 整合開發環境。

2.4.3 開啟專案檔


開啟專案
cs2013 ▸ ch02 ▸ myFirst ▸
1
搜尋 myFirst
組合管理
新增資料夾
視訊
圖片
電腦
OS (C:)
AjaxControlToo
cs2013
cs2013-必修課
DC4cai.csf
名稱
修改日期
類型
bin
2013/7/7 下午 11...
obj
2013/7/7 下午 11...
Properties
2013/7/7 下午 11...
myFirst.csproj
2
2013/3/2 下午 04...
Visual C#
myFirst.sln
2013/3/2 下午 04...
SLN 檔案
檔案名稱(N): myFirst.csproj
所有專案檔 (*.sln;*.dsw;*.vcw;*.
3
開啟(O)
取消

2.5 主控台應用程式格式化輸出入

2.5.1 Console 類別常用方法

Console 類別的方法	說明
Write	將輸出串流(Output Stream)由指定的輸出裝置(預設為螢幕)顯示出來。
WriteLine	將輸出串流(Output Stream)及換行控制字元(Carriage return)由指定的輸出裝置(預設為螢幕)顯示出來
Read	由指定輸入裝置(預設為鍵盤)將鍵入的一個字元讀進來形成一個輸入串流(Input Stream)並放入指定的變數。
ReadLine	由指定輸入裝置(預設為鍵盤)將鍵入的一行字串讀進來形成一個輸入串流(Input Stream)並放入指定的變數。也就是ReadLine()方法允許接受一連串的輸入串流(一行字元)一直到按下  鍵為止。

範例 ：ConsoleEx1.sln

現以一個簡單程式來說明 Write()/WriteLine()/ReadLine()的用法。當程式開始執行時畫面先出現 "請輸入品名："，當使用者由鍵盤鍵入 "碁峰小平板" 按 鍵後，緊接著在第二行出現 "請輸入單價："，當你由鍵盤鍵入 "9000" 按 鍵後，在第三行會顯示 "品名：碁峰小平板 單價：9000 這筆記錄儲存成功"。

執行結果

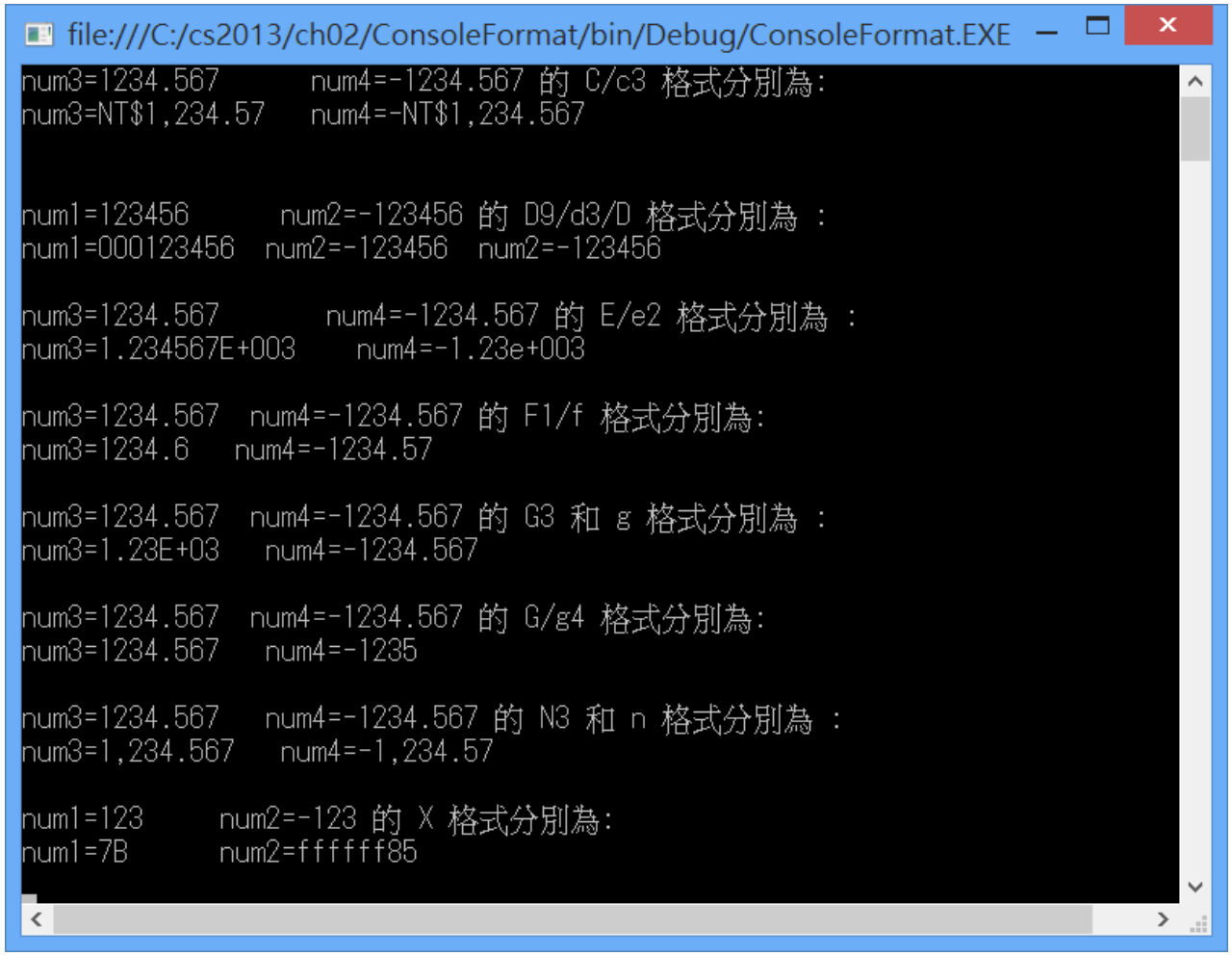
```
10        static void Main(string[] args)
11        {
13            string ProductName;
15            int Price;
16            Console.Write("請輸入品名：");
18            ProductName = Console.ReadLine();
19            Console.Write("請輸入單價：");
22            Price = int.Parse(Console.ReadLine());
23            Console.WriteLine("品名：{0}  單價：{1}  這筆記錄儲存成功",
                                ProductName, Price);
24            Console.Read();
25        }
```

2.5.2 如何進行格式化輸出

字元	功能
C 或 c	以貨幣方式顯示資料，以$開頭。
Dn 或 dn	以指定 n 位數顯示十進位資料，空白處補 0。
E 或 e	以指數方式顯示資料。
Fn 或 fn	以小數有 n 位來顯示資料。
G 或 g	以一般格式顯示。
N 或 n	使用千位符號但不包含$符號，若為 N1 表保留小數至第一位，若為 N 和 N2 都保留小數至第二位。
X 或 x	資料以十六進制顯示。

範例：ConsoleFormat.sln

利用上表 C# 所提供的各種格式化字元，來熟悉各種格式化字元輸出的結果。



2.5.3 如何自訂數值格式輸出字串

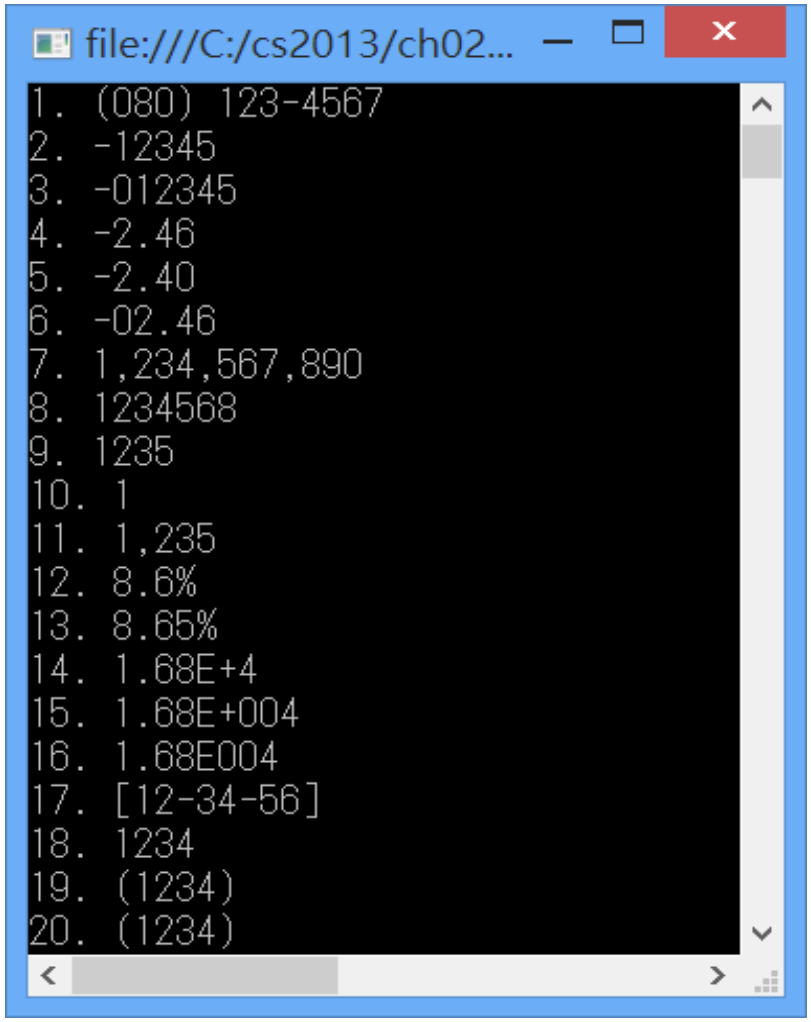
格式字元	功能
0：零值預留位置	若顯示數值的位數比設定格式的位數小，資料向右靠齊顯示，未用到的位數以 0 取代。若數值資料比設定格式位數大，則以該數值大小直接顯示。
#：數字預留位置	若顯示數值的位數比設定格式的位數小，資料向左靠齊顯示，未用到的位數以空白取代。若數值資料比設定格式位數大，則以該數值大小直接顯示。
.：小數點	由小數點右邊的位數來決定小數的位數，若資料的小數位數超過設定的小數位數，會做四捨五入。小數前後可使用#或 0 格式字元。
，：千位分隔符號和數值縮放	使用千位分隔符號，若千位分隔符號放一個在最後面，該數值會除以 1000 顯示，連續兩個千位分隔字元除以 1000000，以此類推。

2.5.3 如何自訂數值格式輸出字串

格式字元	功能
%：百分比預留位置	使用%格式字元會將數值除以 100 顯示。
E：科學標記法	使用科學記號會以指數方式顯示。
\：逸出字元	使用逸出字元，接在此逸出字元後面的字元會被當作指定的特殊意義來處理。
'ABC' | "abc" 常值字串	使用單引號或雙引號，會將括住的字串複製到輸出字串內直接顯示出來。

範例 ：ConsoleToString.sln

練習使用 ToString()方法所提供的自訂格式輸出字元，將數值轉換成字串並輸出，結果如下圖：

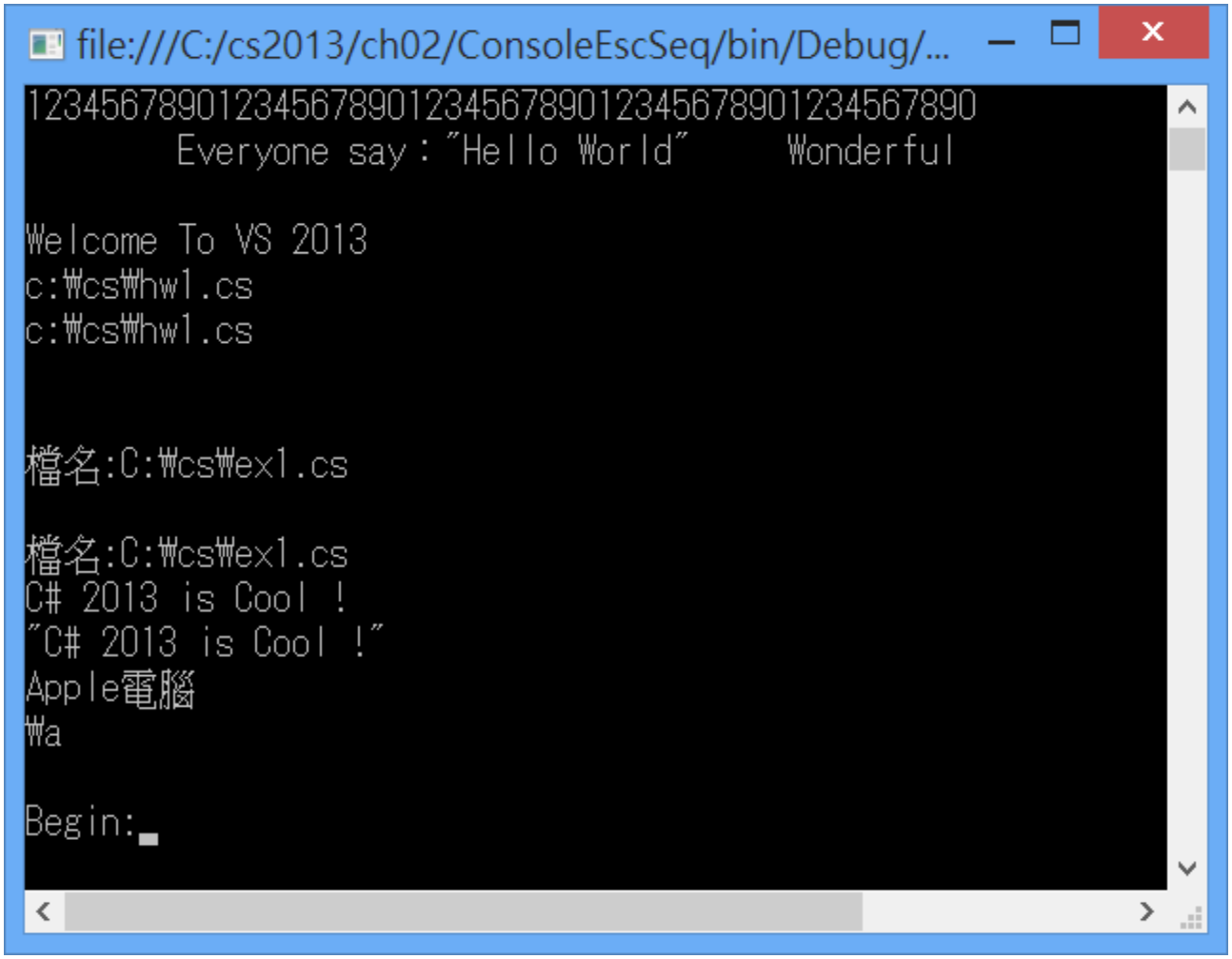


2.5.4 Escape sequence控制字元

逸出序列字元	說明
\'	插入一個單引號。
\"	插入一個雙引號。
\\	插入一個倒斜線，當程式定義檔案路徑時有用。
\a	觸發一個系統的警告聲。
\b (Backspace)	退一格。
\n (New line)	換新行。
\r (Return)	游標移到目前該行的最前面。
\t (Tab)	插入水平跳格到字串中。
\udddd	插入一個 Unicode 字元。
\v	插入垂直跳格到字串中。
\0 (Null space)	代表一個空字元。

範例 ： ConsoleEscSeq.sln

在範例程式中使用 \t , \n , \" , \\ 等逸出序列，觀察其輸出結果。



2.6 資料型別轉換

2.6.1 隱含轉換(Implicit Conversion)

- 隱含轉換亦稱自動轉換
- 當下列兩種情況允許自動轉換：

  1. 兩者的資料型別相容。
  2. 目的資料型別比原始資料型別範圍大時。

2.6.1 隱含轉換

資料型別	可自動轉換至下列資料型別
sbyte	short/int/long/float/double/decimal
byte	short/unshort/int/uint/long/ulong/float/double/decimal
char	ushort/int/uint/long/ulong/float/double/decimal
int	long/float/double/decimal
uint	long/ulong/float/double/decimal
short	int/long/float/double/decimal
ushort	int/uint/long/ulong/float/double/decimal
long	float/double/decimal
ulong	float/double/decimal
float	double

2.6.1 隱含轉換

下例說明資料型別以擴展轉換或縮小轉換做自動轉換的結果：

```
int i;
long l;
float f;
double d;
f = i;   // 擴展轉換，允許自動轉換
d = i;   // 擴展轉換，允許自動轉換
d = l;   // 擴展轉換，允許自動轉換
i = d;   // 縮小轉換，出現 "無法將型別 'double' 隱含轉換成 'int'" 錯誤訊息
l = d;   // 縮小轉換，出現 "無法將型別 'double' 隱含轉換成 'long'" 錯誤訊息
```

2.6.2 明確轉換

- 明確轉換就是強制轉換
- 是利用型態轉換(Type Cast) 方法來強迫資料轉換成其他指定的資料型別
- 它是一個指令用來告知編譯器將某個資料型別轉換成另一個資料型別。
- 明確轉換可能導致轉換結果產生不精確，或產生 “throwing exception ”。
- 轉型運算式語法下：

  (cast)變數名稱或運算式

2.6.2 明確轉換 Continue...

- 如 x 和 y 都是 double 變數，以縮小轉換成 int，由於 double 無法使用自動轉換成 int，因此必須使用 cast 做明確轉換
- 寫法：

```
int i=(int)x;       // 將 x 浮點數轉型成整數再指定給 i 整數
int n=(int)y;       // 將 y 浮點數轉型成整數再指定給 n 整數
```

2.6.2 明確轉換

Continue…

資料型別	可明確轉換至下列資料型別
sbyte	byte/ushort/uint/ulong/char
byte	sbyte/char
char	sbyte/byte/short
uint	sbyte/byte/short/ushort/int/char
int	sbyte/byte/short/ushort/uint/long/ulong/char
short	sbyte/byte/ushort/uint/ulong/char
ushort	sbyte/byte/short/char
long	sbyte/byte/short/ushort/int/uint/ulong/char
ulong	sbyte/byte/short/ushort/int/uint/long/char
float	sbyte/byte/short/ushort/int/uint/long/ulong/char/decimal
double	sbyte/byte/short/ushort/int/uint/long/ulong/char/float/decimal
decimal	sbyte/byte/short/ushort/int/uint/long/ulong/char/float/double

2.6.2 明確轉換 Continue…

資料型別	可轉換型別	轉換結果
float/double/decimal	int	轉換結果會將小數部分捨棄，若超出有效範例則結果為 0。
decimal	float/double	四捨五入接近 float 或 double 值。
double	float	四捨五入接近 float，若轉換值太小或太大以 0 和無窮大表之
float/double	decimal	轉換值以十進位數字表之，四捨五入接近 28 位數值

下例說明使用 cast 對資料型別轉換的影響：

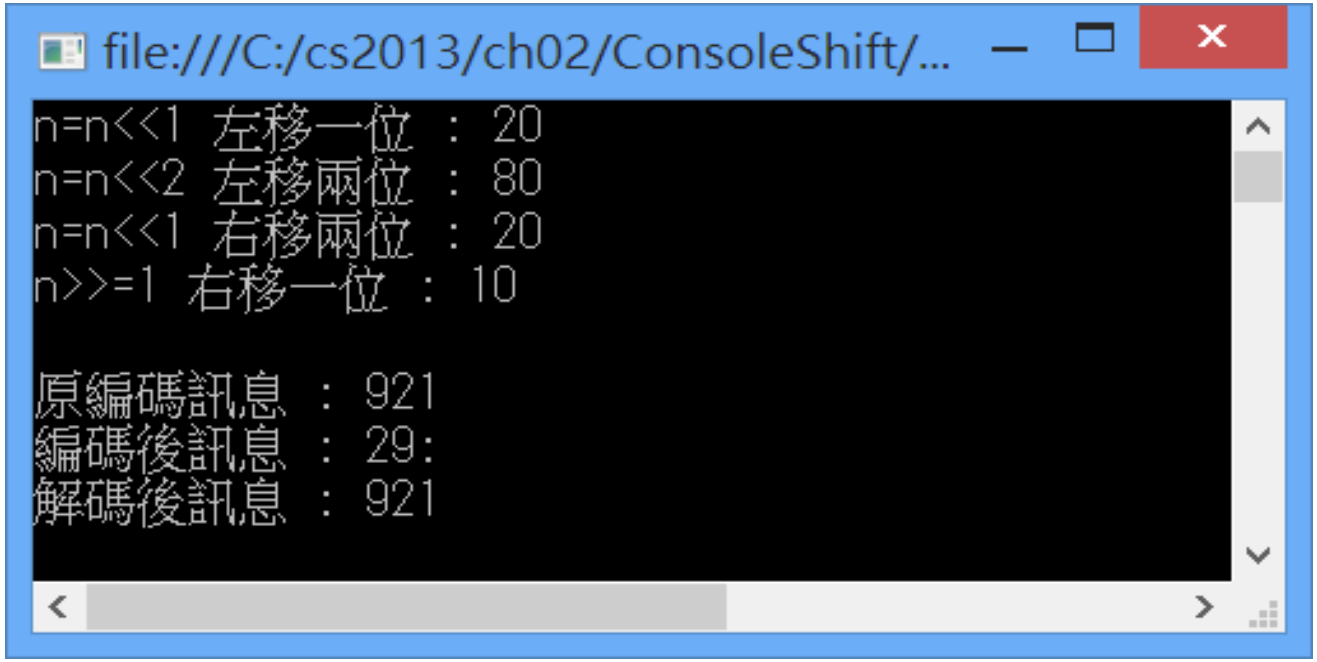
```
int i;
double d = 234.9;
i = (int)d;          // 結果：a=234
i = d;               // 出現 "無法將型別 'double' 隱含轉換成 'int'" 錯誤訊息

byte b;
i =255;
b = (byte) i          // b=255
i = 256;
b = (byte) i          // b=1 轉換值超過範圍資料遺失

char ch;
b = 41;              // ASCII 的 'A'
ch = (char) b;       // ch='A
```

範例 ：ConsoleShift.sln

練習使用移位運算子，右移一位數值乘 2；往左移一位數值除以 2。另外將每個字元與 key 設定值做 XOR 運算加碼並顯示加碼後的字元，再將加碼過的字元做 XOR 運算，解碼還原成原來字元。



2.7 列舉資料型別

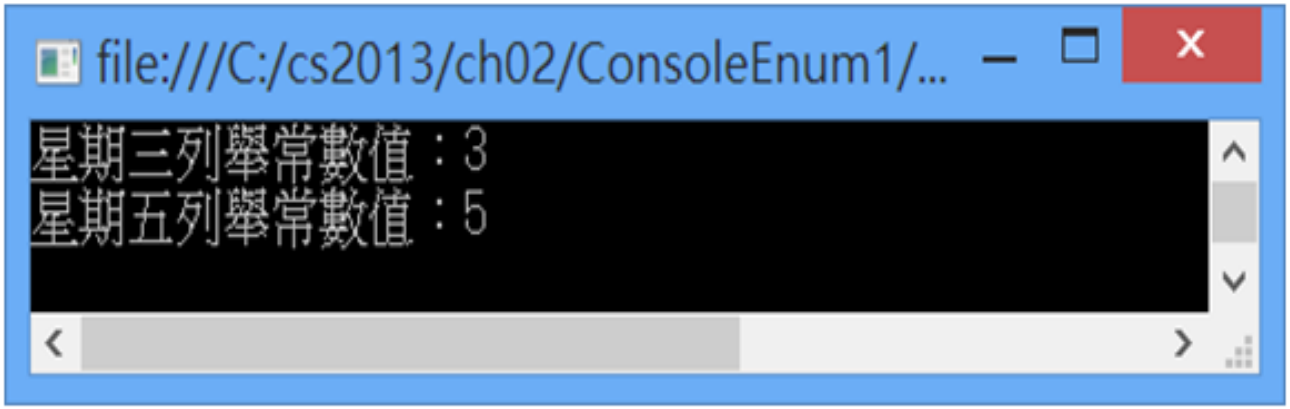
語法

```
[存取修飾詞] enum 列舉名稱 : 資料型別
{
    列舉成員 1=初值 1,
    列舉成員 2=初值 2,
    ……
    列舉成員 N=初值 N
}
```

範例：ConsoleEnum1.sln

定義用來表示一星期七天的 WeekDays 列舉資料型別，WeekDays 列舉的成員依序設為 Monday = 1、Tuesday = 2、Wednesday = 3、Thursday = 4、Friday = 5、Saturday = 6、Sunday = 7。最後請印出 WeekDays.WednesDay 及 WeekDays.Friday 的列舉常數值。

執行結果



2.8 結構資料型別

語法

```
[存取修飾詞] 結構名稱
{
    [存取修飾詞] 資料型別 結構成員 1;    // 結構成員 1

    [存取修飾詞] 資料型別 結構成員 2;    // 結構成員 2
      ……
    [存取修飾詞] 資料型別 結構成員 N;    // 結構成員 N
}
```

2.8.2 結構欄位初值設定

若結構變數的欄位宣告為 private 私有成員，則無法使用「.」來存取。

譬如定義如下 Product 結構：

```
struct Product
{
    public string No, Name;
    public int Price;
}
```

先宣告 cpu 屬於 Product 結構，再設定 cpu 結構變數各欄位的內容：

```
Product cpu ;
cpu.No = "pc01";
cpu.Name = "Xbox One";
cpu.Price =9600;
```

範例 ：ConsoleStruct1.sln

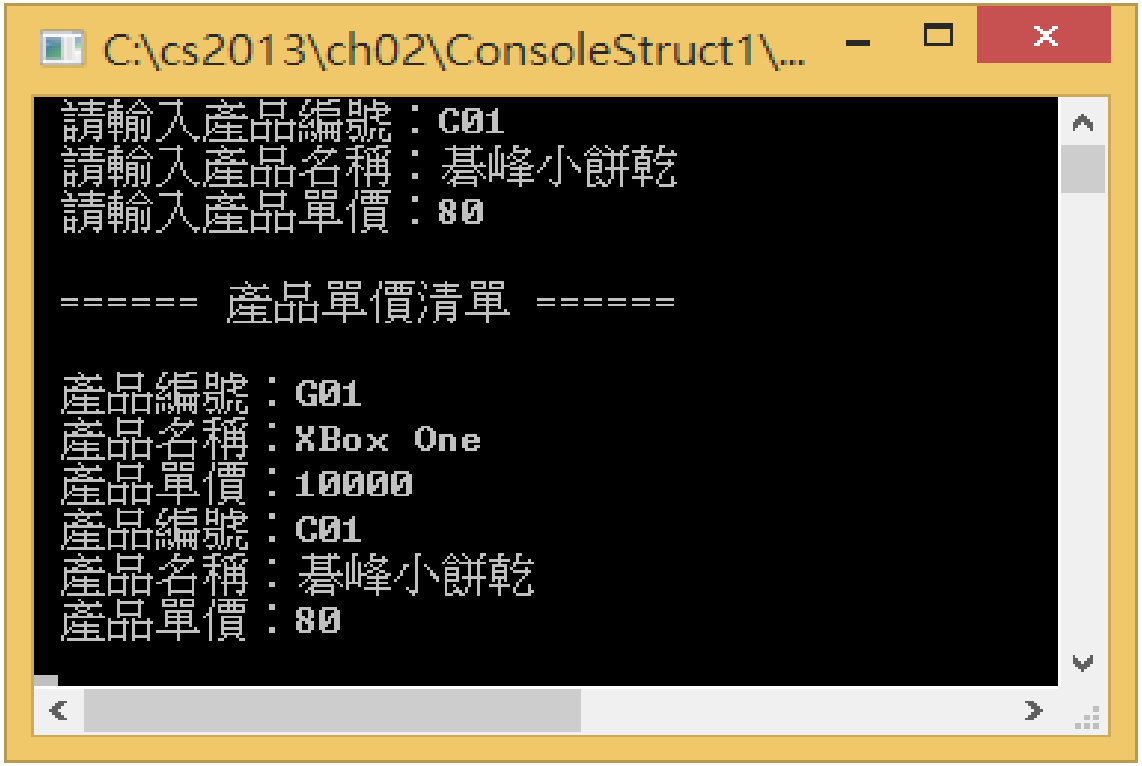
定義一個名稱為 Product 的結構，該結構擁有 No、Name、Price 三個欄位分別用來表示產品編號、品名、單價。並宣告 game 和 cookie 屬於 Product 的結構變數，並要求設定初值：

① game 結構變數使用 "=" 設定初值：編號= "G01 ", 品名="Xbox One", 單價=10000。

② cookie 由鍵盤輸入初值：編號="C01", 品名="碁峰餅乾", 單價=80。

最後再顯示這二件產品各欄位的內容。

輸出結果


C:\cs2013\ch02\ConsoleStruct1\...
請輸入產品編號：C01
請輸入產品名稱：碁峰小餅乾
請輸入產品單價：80
====== 產品單價清單 ======
產品編號：G01
產品名稱：XBox One
產品單價：10000
產品編號：C01
產品名稱：碁峰小餅乾
產品單價：80

本章結束 ...