

目錄

► 01 程式的編譯與執行

- 1-1 .NET Framework..... 1-1
- 1-2 主控台應用程式 1-2
- 1-3 Windows Forms 應用程式 (Windows Forms Application)..... 1-5

► 02 基本語法

- 2-1 關鍵字與識別字 2-1
- 2-2 資料型別..... 2-3
- 2-3 運算子 2-13
- 2-4 敘述與運算式 2-21

► 03 基本輸入與輸出

- 3-1 Console 類別..... 3-1
- 3-2 Form 3-6
- 3-3 Label(標籤)..... 3-9
- 3-4 TextBox 3-10
- 3-5 Button 3-12
- 3-6 PictureBox 3-22
- 3-7 Timer 3-26
- 3-8 Random 類別 3-28
- 3-9 輸出入對話框 3-31
- 3-10 實例探討..... 3-35

► 04 決策敘述

- 4-1 if...else 4-1
- 4-2 switch...case 4-10
- 4-3 實例探討..... 4-18

► 05 迴圈敘述

5-1	for	5-1
5-2	while 與 do	5-12
5-3	goto.....	5-21
5-4	實例研究.....	5-22

► 06 陣列

6-1	一維陣列.....	6-1
6-2	二維或多維陣列	6-15
6-3	陣列成員.....	6-33
6-4	控制項陣列.....	6-38
6-5	實例研究.....	6-44

► 07 方法

7-1	方法的建置.....	7-1
7-2	參數的傳遞.....	7-9
7-3	方法多載 (Over loading Methods).....	7-17
7-4	遞迴	7-18
7-5	抽象化 (Abstraction).....	7-23
7-6	綜合範例.....	7-24

► 08 公用類別庫

8-1	數值	8-1
8-2	字串	8-7
8-3	時間	8-34
8-4	集合	8-47
8-5	串流與檔案.....	8-77

► 09 類別與物件

9-1	類別與物件的建置.....	9-2
9-2	建構子 (Constructors)	9-10
9-3	屬性成員.....	9-13

9-4	索引子 (indexers) (非資訊本科系可先跳過)	9-17
9-5	運算子多載	9-19
▶ 10	列舉、結構、介面	
10-1	列舉 (非資訊本科系可先跳過)	10-1
10-2	結構 (非資訊本科系可先跳過)	10-3
10-3	介面	10-7
▶ 11	繼承	
11-1	繼承介面	11-1
11-2	繼承類別	11-4
11-3	自訂類別庫	11-17
▶ 12	委派與事件	
12-1	委派 (非資訊本科系可先跳過)	12-1
12-2	事件 (非資訊本科系可先跳過)	12-5
12-3	.NET 事件 (非資訊本科系可先跳過)	12-12
▶ 13	泛型	
13-1	泛型類別的使用 (非資訊本科系可先跳過)	13-1
13-2	自訂泛型類別 (非資訊本科系可先跳過)	13-3
▶ 14	進階輸出入控制項	
14-2	RadioButton	14-3
14-3	GroupBox	14-5
14-4	ListBox	14-8
14-5	ComboBox	14-11
14-6	HScrollBar 與 VScrollBar	14-14
14-7	功能表與工具列	14-17
14-8	NumberUpDown	14-24
14-9	RichTextBox	14-24
14-10	對話框	14-30

14-11 WebBrowser 控制項	14-37
14-12 DateTimePicker	14-38
14-13 MonthCalendar	14-39
▶ 15 控制項事件	
15-1 KeyPress	15-1
15-2 KeyDown 及 KeyUp	15-2
15-3 MouseDown、MouseUp 及 MouseMove	15-6
▶ 16 繪圖	
16-1 繪圖的基本觀念	16-1
16-2 繪圖屬性與方法	16-7
16-3 繪圖相關類別	16-19
16-4 實例探討 (一)	16-28
16-5 Image 與 Bitmap 類別	16-42
16-6 實例探討 (二)	16-53
▶ 17 C# 影像處理與字幕機專題製作	
▶ 18 網路應用程式	
18-1 Dns 與 IPHostEntry 類別	18-1
18-2 主從傳輸程式	18-5
▶ 19 專題製作	
19-1 梭哈遊戲	19-1
19-2 超級記憶卡	19-12
19-3 十點半遊戲	19-24
19-4 指針式時鐘	19-34

基本語法

本章將介紹一些程式語言的基本語法，例如識別字的命名、資料型別、運算子及運算式等。

2-1 關鍵字與識別字

本單元先介紹關鍵字與識別字。

＜ 關鍵字 (Keywords)

關鍵字（或稱保留字）是任一程式語言已事先賦予某一字元或字串一個特別意義，所以程式設計者不得再重複賦予不同的用途。例如，if 已被賦予決策敘述，程式設計者當然不得再定義 if 為另外的用途。以下是 C# 的關鍵字。

abstract	event	new	struct
as	explicit	null	switch
base	extern	object	this
bool	false	operator	throw
break	finally	out	true
byte	fixed	override	try
case	float	params	typeof
catch	for	private	uint
char	foreach	protected	ulong
checked	goto	public	unchecked

class	if	readonly	unsafe
const	implicit	ref	ushort
continue	in	return	using
decimal	int	sbyte	virtual
default	interface	sealed	volatile
delegate	internal	short	void
do	is	sizeof	while
double	lock	stackalloc	
else	long	static	
enum	namespace	string	

✕ 識別字(Identifier)

真實的世界裏，每個人、事及物都有一個名稱，程式設計亦不例外，於程式設計時我們必須為每一個變數、常數、方法類別及物件等命名，以上所有變數、常數、方法等名稱，統稱為程式語言的識別字。C# 的識別字命名規則如下：

1. 識別字必須是以字母（大小寫的 A 至 Z）、@ 或底線（_）開頭。例如，以下是一些合法的識別字。

```
a  
i  
sum  
_sum  
Income  
@if
```

以下是一些非法的識別字。

```
+a  
$sum  
#Age3  
7eleven // 不能由數字開頭
```

2. 識別字由字母開頭後，可由字母、數字組合而成，但不得包含空白。
例如，以下是一些合法的識別字。

```
a123  
a123b
```

以下是一些非法的識別字。

```
a c    // 不能含空白  
c+3    // 不得含有加號  
Age#3  // 不得含有# 號
```

3. 識別字的長度不限，但是太長也徒增人工識別與鍵入的困擾。
4. 識別字的大小寫均視為不同，例如 `Score`、`score` 及 `SCORE` 皆代表不同的識別字。
5. 識別字不得使用保留字，如 `if`、`for` 等。但前面加上 `@` 則不在此限，例如 `@if`。
6. 識別字要用有意義的單字，例如 `StudentNumber` 或 `AverageIncome`。除非生命週期極短的變數才用 `x`、`i` 或 `a` 等當識別字，也千萬不要用 `k23erp` 這種沒意義又難記的識別字。
7. 識別字有多個單字時，中間可以加上底線 (`_`)，例如上例的 `StudentNumber` 可寫成 `Student_Number`，若擔心打字不靈光亦可寫成 `Stu_Num`、`stu_num`、`StuNum` 或 `stumum`，其中 `StuNum` 又稱駱駝表示法，因為大寫字母起來像駝峰一樣，可以避免鍵入底線的困擾、且提昇閱讀效率。

2-2 資料型別

✧ 資料的表示

電腦的主要工作就是處理資料，C# 所能處理的資料為整數、浮點數、布林值、字元與字串，分別說明如下：

整數

C# 可以處理的整數有兩種進位方式，分別是十進位與十六進位。十進位的表示方式同我們平常的書寫方式，如 25；十六進位的表示方式為在數字的前面加上 0x，例如，0x12 則為十進位的 18。

浮點數

數字中含有小數點或指數的稱為浮點數。以指數為例，E 或 e 表示 10 的次方，例如 0.0023、2.3E-3 及 2.3e-3 均是表示相同的浮點數；又例如，2.3E+2 則代表 230 且加號 (+) 不可省略。其次，C# 的浮點數分為二種表示方式，分別是 float 與 double。兩者的差別是後者的有效數字及值域較大，所以為了明確表示浮點數的種類，應於浮點數後以 f 或 F 指定為 float；或以 d 或 D 指定為 double。其次，若未指定 float 或 double，則 C# 預設為 double。

字元 (Char)

C# 的字元必須使用單引號 (') 於單一字元兩旁括住。例如，以下是一些合法的字元。

```
'a'  
'洪'
```

其次，字元 a 的 Unicode 碼是 0061(十元進制)，所以，

```
'\u0061'    // Unicode  
'\x0061'    // 十六進位的無正負號短整數
```

亦代表字元 a。此外，若試圖將兩個或兩個以上的字元以單引號括住，則會出現錯誤信息。例如，以下是非法的字元。

```
'ab'
```

字串 (String)

使用雙引號 (") 括起的若干字元，稱為字串。例如，以下是一合法的字串。

```
"This is a book."  
"張中立"
```

跳脫字元 (Escape Characters)

有些字元已被編譯器賦予某些功能，若要使用這些字元，則應使用跳脫字元 (\)。例如，單引號已被編譯器賦予括住字元的功能，若一定要使用此符號，則應於此字元前加上跳脫字元 (\)，例如以下敘述將印出一個單引號 (')。

```
d='\ ' ;  
Console.WriteLine(d)
```

以下敘述；將會印出 This's a book.。

```
e="This\'s a book. ";  
Console.WriteLine(e);
```

下表是一些常見的跳脫字元。

Escape sequence	字元名稱	Unicode 編碼
\'	單引號	0x0027
\"	雙引號	0x0022
\\	反斜線符號	0x005C
\0	Null	0x0000
\a	發出一聲警告	0x0007
\b	倒退一個字元	0x0008
\f	Form Feed	0x000C
\n	新的一行	0x000A
\r	換行並移到最前方	0x000D
\t	水平方向的 Tab	0x0009
\v	垂直方向的 Tab	0x000B

✧ 資料型別

C# 所能處理的資料如上所述，接下來要討論的是電腦應如何儲存以上資料。因為不同性質的資料需要的記憶體空間均不同，而 C# 為了有效地使用記憶體儲存資料，於是規劃了以下的資料型別，分別是數值型別的整數、浮點數、decimal、char、bool 及參照型別的 object 與 string。這些型別又稱為預設的資料型別，以便提昇資料處理效率，也就是讓小東西用小箱子裝、大東西用大箱子裝的道理，如此才能更有效率地管理記憶體。

整數型別 (Integer)

整數型別共包含 sbyte、byte、short、ushort、int、uint、long 及 ulong 等型別，這些型別所對應的類別名稱、所需記憶體大小及值域（所能表示的範圍，稱為值域），如下表所示：（進一步的資料，請線上查詢 "資料型別"）

別名	類別名稱	所需記憶體 大小(位元組)	值域 (所能表示的範圍)
sbyte	System.Sbyte	1	-128 ~ 127
byte	System.Byte	1	0 ~ 255
short	System.Int16	2	-32768 ~ 32767
ushort	System.UInt16	2	0 ~ 65535
int	System.Int32	4	-2,147,483,648 ~ 2,147,483,647
uint	System.UInt32	4	0 ~ 4,294,967,295
long	System.Int64	8	-9,223,372,036,854,775,808 ~ 9,223,372,036,854,775,807
ulong	System.UInt64	8	0 ~ 18,446,744,073,709,551,615

什麼是 " 別名 "

所有預先定義的型別，它們在 System 的命名空間中均有已定義的類別。例如：int 型別其背後的類別是 System.Int32。所以，以下敘述的效果相同。

```
int i=5;
System.Int32 i=5;
```

以上敘述的 `int` 則是 `System.Int32` 的別名。

浮點數 (floating-point) 型別

可以儲存實數的型別稱為浮點數型別。C# 的浮點數型別分別是 `float` 與 `double`，此兩種型別的類別名稱、所需記憶體大小、有效位數及值域如下表所示。

別名	類別名稱	所需記憶體 大小(位元組)	有效位數	值域
float	System.Single	4	7	負值約 $-3.4 \times 10^{38} \sim -1.5 \times 10^{-45}$ 正值約 $1.5 \times 10^{-45} \sim 3.4 \times 10^{38}$
double	System.Double	8	15 ~ 16	負值約 $-1.8 \times 10^{308} \sim -5.0 \times 10^{-324}$ 正值約 $5.0 \times 10^{-324} \sim 1.8 \times 10^{308}$

decimal 型別

前面的浮點數型別，所能代表的有效位數至多 16 位，若要表示精密度非常高的浮點數，例如 10000000000.00000000001，則無法表示。所以 C# 另外提供 `decimal` 型別，此型別的有效數字可達 28 位。

但是，你要求的精確度是如此高，你也要付出 16 位元的代價，才能儲存一個 `decimal` 的變數。`decimal` 的類別名稱、所需記憶體大小、有效位數及值域，如下表所示。

別名	類別名稱	所需記憶體 (位元組)	有效位數	值域
decimal	System.Decimal	16	28 ~ 29	約 $\pm 1.0 \times 10^{-28} \sim \pm 7.9 \times 10^{28}$

char 型別

char 型別可儲存一個 Unicode 字符。下表是 char 型別的類別名稱與所佔記憶體大小。

別名	類別名稱	所需記憶體(位元組)
char	System.Char	2

object 型別

object 型別的類別名稱與所需記憶體大小如下表所示。

別名	類別名稱	所需記憶體(位元組)
object	System.Object	0 或 8

bool 型別

bool 型別可儲存一個布林值。下表是 bool 型別的類別名稱與所需記憶體的大小。雖然布林值只需一個位元 (0 或 1)，但它卻佔用二個位元組，那是因為位元組是大部分處理器的最小處理單元。

別名	類別名稱	所需記憶體(位元組)
bool	System.Boolean	2

string 型別

C# 的 string 型別可儲存一串 Unicode 的字符序列，它是代表 System.String 類別。雖然，string 是一個類別，但它在程式中卻無所不在，這是因為 string 類別較為常用且特殊，所以給予一些特權。例如，您可以直接使用以下敘述建立一個 string 類別的物件 a。

```
string a = "This is a book. ";
```

下表是 string 型別的類別名稱與所需記憶體大小。

別名	類別名稱	所需記憶體(位元組)
string	System.String	至少 20

變數宣告

變數的功能是用來輸入、處理及儲存外界的資料，而變數在使用以前均要宣告才可使用。在一些舊式的 Basic 語言中，變數並不需要事先宣告，卻也帶來極大的困擾。以下敘述即為變數未宣告的結果，編譯器便無法回應使用者在拼字上的錯誤，而造成除錯上的困難。

```
student = studend+1;
```

以上敘述若事先宣告 student 如下：

```
int student;
```

則編譯器遇到 studend 時，便會提醒使用者此 studend 並未宣告的錯誤訊息，提醒使用者補宣告或注意拼字錯誤。

其次，變數宣告的優點是可配置恰當的記憶體而提高資料的處理效率。例如，有些變數的值域僅為整數，則不用宣告為浮點數或 decimal。甚至有些變數的值域非常小，小到使用 byte 型別即可儲存，此時當然宣告為 byte 型別即可。

C# 變數宣告的語法如下：

```
型別名稱      變數名稱 [= 初值];
```

例如，以下敘述可宣告變數 a 的型別為 int。

```
int a ;
```

若有兩個以上的變數有相同的型別，亦可同時給予宣告，但是變數之間應以逗號(,) 隔開。例如，以下敘述可同時宣告變數 b、c 的型別為 char。

```
char b, c ;
```

其次，C# 並不允許使用未經初始化的變數。所以，亦可於變數宣告時一併給予初值。例如，以下敘述為宣告變數 d 的型別是 string，其初值

是 "Mary" 。

```
string d = "Mary";
```

以下敘述宣告變數 `e` 的型別為 `float`，且其初值是 `0.0023`。其次，浮點數的預設型別是 `double`。所以，若欲指定為 `float`，則需要於浮點數後面加上 `f` 或 `F`。

```
floate=0.0023f;    //請務必於 0.023後面加上 f 或 F
```

以下敘述可同時宣告變數 `a` 與 `b`，並設定其初值。

```
int a=3, b=5;
```

✕ 型別轉換

每一個變數宣告之後，即有屬於自己的型別，往後此變數均只能指定給相同型別的變數儲存，若執行階段欲指定給不同型別的變數儲存，則稱此為型別轉換。C# 的型別轉換分為隱含轉換 (Implicit Conversion) 與強制轉換 (Explicit Conversion)，分別說明如下：

隱含轉換 (Implicit Conversion)

將值域小的型別轉為值域大的型別，稱為自動轉換或轉型 (Convert)。此種轉換，系統可自動處理並確保資料不會流失。例如，將 `short` 轉為 `int` 或 `long`，則因後者的值域均比前者大，所以可順利的轉換。以下敘述可將型別為 `short` 的變數 `a` 指定給型別為 `int` 的變數 `b`，且原值不會改變。

```
short a=23;
int b;
b=a ;
Console.WriteLine(a); // 結果是23
```

強制轉換 (Explicit Conversion)

將值域大的轉為值域小的型別 (如 int 轉為 short)，或不同值域之間的互轉 (如 byte 轉為 sbyte)，則稱此為強制轉換或稱為鑄型 (Cast)。強制轉換的語法如下：

```
變數1 = (變數的型別) 變數2;
```

例如，以下敘述可將型別是 int 的 b 變數指定給型別是 short 的 a 變數。

```
int b=23;
short a;
a=(short) b;
Console.WriteLine(b); // 結果是23
```

其次，強制轉換的風險比較大，有可能資料流失或溢位。例如，以下敘述將 float 型別強制轉換為 byte 型別，將造成小數點後面的數字流失。

```
float c= 3.4f;
byte d;
d=(byte)c;
Console.WriteLine(d); // 結果是3
```

以下敘述將 byte 型別的 d 變數轉為 sbyte，將造成溢位，輸出為 -1。

範例 2-2a

示範變數的型別轉換。

程式列印

```
//Implicite Conversion
short a=23;
int b;
b=a;
Console.WriteLine ("1:  "+b); //23
//Emplicite Conversion)
```

```
a=(short)b;  
Console.WriteLine ("2:  "+a);    //23  
float c=3.4f;  
byte d;  
d=(byte)c;  
Console.WriteLine ("3:  "+d);    //3  
d=255;  
sbyte e;  
e=(sbyte) d;  
Console.WriteLine ("4:  "+e);    //-1
```

✧ 常數符號宣告

跟變數一樣，常數符號（以下簡稱常數）亦需要配置記憶體，與變數不同的是，常數符號正如其名稱所示，在整個程式中都不會改變其值，故稱為常數。程式設計中，有兩種表示常數的方式，一種是文字式 (Literal)，例如直接以 15 或 3.14159 表示某一常數；另一種是常數符號式 (Symbolic)，則是本單元所要介紹的。之所以需要常數符號，是因為有些數字在程式中會不斷的重複出現，為了增加程式的可讀性及降低程式的維護成本，此時即可用一個有意義的符號代替，但必須在符號之前加上保留字 **const**，則該符號的值將永遠保持在所宣告的符號中，程式中任何位置均不能改變其值，此稱為常數符號，簡稱常數。

例如令 **const float PI=3.14159f**，則每次要使用 3.14159 時，只要填入 PI 即可（註：常數符號通常將每一個字母均用大寫表示）。又例如，於定存利息的計算，利率應以常數符號代替，程式中只要引用此利率時，均應使用此常數符號，則當利率改變時，只要修改程式前頭的利率常數即可。若未使用常數符號統一此值，則因此常數散落程式各地而無法確保此值的一致性。

2-3 運算子

可以對運算元 (Operand) 執行特定功能的特殊符號稱為運算子。一般而言，運算子可分為以下幾類：指派 (Assignment) 運算子、算術 (Arithmetic) 運算子、關係 (Relational) 運算子、邏輯 (Logical) 運算子、字串 (String) 運算子、複合指定運算子及位元操作 (Bitwise) 等運算子。而每一種運算子都可以再細分為一元 (Unary) 運算子與二元 (Binary) 運算子。一元運算子只需要一個運算元就可以操作，而二元運算子則需要兩個運算元才能夠操作。在以下單元中，我們除了檢視各種不同的運算子功能外，還將討論運算子的優先順序 (Precedence) 與順序關聯性 (Associativity)。優先順序是用來決定同一運算式擁有多個運算子時，每一個運算子進行運算的順序；而順序關聯性則決定了在同運算式中，相同優先順序的運算子執行順序。

✂ 指派運算子 (Assignment operator)

指派 (或稱指定) 運算子的符號為 = (只有 Pascal 與 Delphi 採用 :=)，其作用為將運算符號右邊運算式的值指派給運算符號左邊的運算元。所以，以下敘述的 `sum = a+b;` 是將 `a+b` 的值指派給 `sum`。

```
sum = 0, a = 3, b = 5;  
sum = a+b;
```

上式與數學的等號是不同的，所以不要一直懷疑為什麼 0 會等於 8。其次，你不能將常數放在指派運算子的左邊，例如，

```
8 = x;
```

為一個不合法的敘述，但以下敘述將常數 8 指派給變數 `x` 為合法的。

```
x = 8;
```

✧ 算術運算子 (Arithmetic operators)

算術運算子用來執行一般的數學運算，包括取負數 (-)、取正數 (+)、加 (+)、減 (-)、乘 (*)、除 (/)、取餘數 (%)、遞增 (++) 及遞減 (--) 等，如下表所示：

運算子符號	功能	範例	結果
+	取正數	+2	
-	取負數	-2	
+	加	6+2	8
-	減	6-2	4
*	乘	6*2	12
/	除法	6/2	3
%	取餘數	6 % 2	0
++	遞增	x=1; x++; Console.Write(x);	2
--	遞減	x=1; x--; Console.Write(x);	0

/(除法)

傳回兩數相除的結果，只有被除數與除數都是整數，才傳回整數，只要被除數或除數有一實數，都傳回實數。

```
int a = 7, b = 3; double d = 3; double e = 7;
Console.WriteLine(a/b);
Console.WriteLine(a/d);
Console.WriteLine(e / b);
Console.WriteLine(e / d);
```

++/-- (遞增 / 遞減)

遞增 (++) 及遞減 (--) 又分為前置與後置，前置是運算子在運算元之前，如 ++a，後置是運算子在運算元之後，如 a++，原則上不論 ++a 或

`a++` 都是將 `a` 值加 1 並放回 `a`，但若是 `b=++a` 和 `b=a++`，則其 `a` 值均會加 1，但 `b` 值會有差異，前置 `b` 值會得到加 1 的結果，後者只能得原 `a` 值。請執行以下程式，並觀察執行結果。

```
int a = 1, b = 1;
a++;
++b;
Console.WriteLine(a);
Console.WriteLine(b);
a = 1;
b = ++a;
Console.Write(a); Console.WriteLine(b);
a = 1;
b = a++;
Console.Write(a); Console.WriteLine(b);
```

其次，遞增減運算子是一種單獨存在的運算子，我不希望有人寫成 `c=a+++2`，然後考別人 `c` 值為何，雖然這式子有解，但確是沒有意義的。因為遞增減運算子是因為程式設計者常會撰寫 `a=a+1` 或 `a=a-1` 之類的敘述，而發展出來的簡略式子，所以千萬不要以此發展一些只有自己看得懂的東西為樂。

✂ 關係運算子 (Relational operators)

關係運算子又稱為比較運算子，用於資料之間的大小比較，比較的結果可得到邏輯的 `True` 或 `False`，下表是 C# 的關係運算子符號。

運算子符號	功能	範例	結果
<code>==</code>	等於	<code>2 == 3</code>	<code>False</code>
<code>!=</code>	不等於	<code>2 != 3</code>	<code>True</code>
<code><</code>	小於	<code>2 < 3</code>	<code>True</code>
<code>></code>	大於	<code>2 > 3</code>	<code>False</code>
<code><=</code>	小於或等於	<code>2 <= 3</code>	<code>True</code>
<code>>=</code>	大於或等於	<code>2 >= 3</code>	<code>False</code>

例如：

```
Console.WriteLine(2 == 3);
```

得到 False。

✧ 邏輯運算子 (Logical operators)

當同一個運算式同時具有兩個以上的關係運算子時，則每兩個關係運算子之間必須使用邏輯運算子連結。C# 的邏輯運算子如下表所示：

運算子符號	功能	範例	結果
&&	AND	(2>3)&&(3==1)	False
	OR	(2>3) (3==1)	False
!	NOT	!True	False
		!False	True

例如，要如何判斷 x 滿足 $1 < x < 5$ 呢？那就是 $(1 < x) \&\& (x < 5)$ 。

✧ 字串運算子 (String operator)

前面關係運算子的 ==、!=、>、<、>= 及 <= 均適用於字串的大小比較，而下表的字串連結運算子 (+)，則可連結兩個字串。

運算子符號	功能	範例	結果
+	字串連結	"abc"+"xyz"	"abcxyz"

✧ 複合指派運算子

結合指派與算術、關係及邏輯的運算子稱為複合指派運算子。例如，程式設計者常會鍵入 `sum=sum+5`，為了簡化此敘述而定義此一複合指派運算子 +=。所以，以上敘述 `sum=sum+5`，即可寫成 `sum+=5`。下表是 C# 常用的複合指派運算子。

運算子符號	功能	範例	結果
+=	相加後指定	a=1 ; a+=2;	3
-=	相減後指定	a=1 ; a-=2;	-1
=	相乘後指定	a=2 ; a=3;	6
/=	相除後指定	a=6 ; a/=3;	2
%=	取餘數後指定	a=7 ; a%=2;	1
&=	AND 運算後指定	a=6 ; a&=2;	2
=	OR 運算後指定	a=6 ; a =2;	6

✂ 位元運算子

將指定的運算元先轉為二進位，再逐一執行位元之間的運算，稱為位元運算子。C# 常用的位元運算子說明如下：

& (AND)

兩個運算元皆為 1，才能得到 1，否則為 0。例如：

```
a=6;    //(00000110)
b=255;  //(11111111)
c=a&b;
Console.Write(c);
```

結果是 6，其二進位是 00000110。

| (OR)

兩個運算元只要有一個 1，就得到 1，否則為 0，例如：

```
a=6;    //(00000110)
b=255;  //(11111111)
c=a|b;
Console.Write(c);
```

結果是 255，其二進位是 11111111。

^ (XOR，互斥或)

兩個運算子不同時，就得到 1，否則為 0。例如：

```
a=6;    //(00000110)
b=255;  //(11111111)
c=a^b;
Console.Write(c);
```

結果是 249，其二進位是 11111001。

~ (取補數)

此為單一運算子，將運算元的 1 轉為 0，0 轉為 1。例如：

```
a= ~a; //(a=00000110)
Console.Write(a);
```

結果是 249，其二進位是 11111001。

>> (右移)

將運算元右移，左邊補 0。例如：

```
b=255;
b=b>>2;           //右移2個位元
Console.Write(b);
```

結果是 63，其二進位是 00111111。

<< (左移)

將運算元左移，右邊補 0。例如：

```
b=63;              //00111111
b=b<<2;            //左移2個位元
Console.Write(b);
```

結果是 252，其二進位是 11111100。

✧ 雜項運算子

以下是一些無法歸類的運算子，列表如下。這些運算子將會在往後各章陸續介紹。

運算子符號	使用範例	說明
.	System.String	連結物件與類別成員
[]	A[]	一維陣列
[,]	A[,]	二維陣列
()	Work()	方法的叫用

✧ 運算子的優先順序 (Precedence)

於較複雜的運算式中，通常同時存在許多運算子，此時就需要定義其優先順序，下表即是 C# 關於運算子的優先順序表。

分類	運算子	優先順序	順序關聯性
主運算 (Primary)	. , a[] , a() , x++ , x--	1(最高)	
一元運算子 (Unary)	+ , - , ! , ~ , ++x , --x	2	
乘除	* , / , %	3	左
加減	+ , -	4	左
移位	<< , >>	5	左
關係	< , > , <= , >= , is , as	6	左
相等	== , !=	7	左
Bitwise AND	&	8	左
Bitwise XOR	^	9	左
Bitwise OR		10	左
Logical AND	&&	11	左
Logical OR		12	左
指派與複合指派	= , *= , /= , %= , += , -= , <<= , >>= , &= , ^= , =	13	右

例如：

```
x=x+y*z;
```

等效於以下敘述。

```
x=(x+(y*z)); //乘法優先順序較加法高
```

又例如：

```
z=x>2 & y<3; //關係優先順序較邏輯高
```

等效於

```
z=(x>2) & (y<3)
```

➤ 順序關聯性

當同一敘述，相鄰的運算子擁有相同的優先順序時，即需定義運算子是左結合或右結合。例如：

```
x=a-b-c;
```

同樣是減號(-)，優先順序相同，此時就要靠定義結合律，減法結合律是由左至右，所以以上同義於

```
x=((a-b)-c);
```

而

```
x=y=z=2;
```

指派運算子的結合律是由右至左，所以以上式子同義於

```
(x=(y=(z=2))) ;
```

所以以上式子，x、y、z的結果都是2。

範例 2-3a

請將右列數學敘述以 C# 敘述表示。

$$1. x = \frac{a+b+c}{3}$$

$$2. y = \frac{a+b}{a-b}$$

 程式列印

```
1. x=(a+b+c)/3;  
2. y=(a+b)/(a-b);
```

2-4 敘述與運算式

✧ 運算式 (Expression)

任何可求得值的式子，均稱為運算式。例如， $5+3$ 會傳回一個數值，所以 $5+3$ 是一個運算式。一般而言，可以放在等號右邊的東西，都可以稱為運算式。例如，以下 `Console.Write()` 括號內的東西均稱為運算式。

```
i=0, a=3, b=5;  
Console.Write(i+1);    // 1  
Console.Write(a+b);    // 8  
Console.Write(a>b);    // False  
Console.Write(i==0);   // True
```

✧ 敘述 (Statement)

凡是控制執行的順序、對運算式取值或不作任何事，均可稱敘述或陳述式，以下式子即是一個敘述。其次，每一敘述均應以分號(;)結束。

```
sum=sum+1;
```

✧ 敘述區塊 (Block Statement)或複合敘述 (Compound Statement)

任何可以放置單一敘述的地方，也就能放敘述區塊，敘述區塊亦稱為複合敘述。一個敘述區塊是由左右兩個大括號組合而成，如下所示：

```
{  
    t=a;  
    a=b;  
    b=t;  
}
```

✧ 註解 (Comments)

適當的程式註解能增加程式的可讀性。其次，註解是給人看的，編譯器均不予理會註解的內容。C# 的註解有二種表示方式，分別是 `(/*)` 與 `(//)`，說明如下。

`/*` (多列註解)

`/*` 後面的文字視為註解，直到遇到 `*/` 為止。例如，以下敘述的 `sum=sum+x` 將會被視為註解，而未執行。

```
x=x+1;      /* x 值加 1  
sum=sum+x;  
將 x 值累加至sum */
```

`//` (單列註解)

凡是放在雙斜線 `(//)` 後面的一整列文字，均視為註解。例如，以下敘述的 "x 值加 1" 即為註解，不會被執行。

```
x=x+1;      //x 值加 1  
sum=sum+x;  //將 x 值累加至 sum
```

✧ 命名空間 (Namespace)

C# 的類別非常多，常會造成命名的困擾，且程式編譯時若全部載入，將大大影響程式的執行效率，所以乃將功能相近的類別合併，並給予

一個資料夾名稱，此資料夾名稱即稱為命名空間，待程式設計者要使用時，再個別載入即可。例如，`Byte`、`Sbyte`、`Int16` 等結構及 `String`、`Math`、`Console` 等類別都非常接近與常用，所以另以 `System` 歸類，使用時則以資料夾名稱 . 類別名稱叫用，如以下敘述：

```
System.Console.WriteLine();
```

但每次都寫資料夾名稱，則有點麻煩，所以一般都在程式前頭，先行宣告所要引用的命名空間，例如，開啓主控台應用程式時，均已引用 `System` 命名空間，如以下敘述：

```
using System;
```

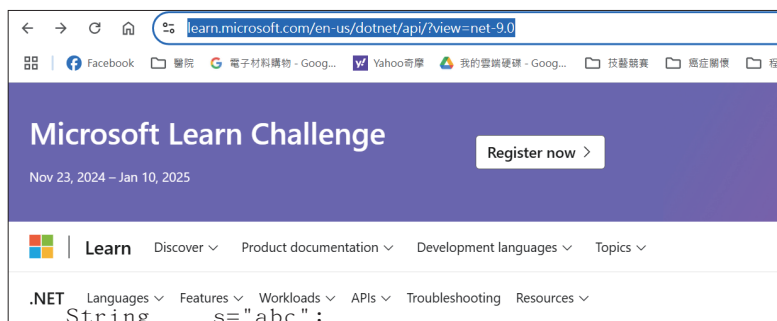
所以

```
System.Console.WriteLine();
```

就可直接簡化寫成

```
Console.WriteLine();
```

下圖則是 .NET Framework 的命名空間 (<https://learn.microsoft.com/en-us/dotnet/api/?view=net-9.0>)，例如，`System` 及 `System.Data` 等等都稱為命名空間，點選後內部都包含很多功能相近的類別。



自我練習

1. 請練習查詢 `Console` 類別，練習找到 `WriteLine()` 方法，下一章會介紹此類別。

2. 請練習查詢 Random 類別，練習找到 Next() 方法，下一章會介紹此類別。

✧ 類別成員 (Class Member)

常見的類別成員有常數、資料、方法及屬性等成員，且都分為靜態與實例兩種，分別說明如下：

靜態 (Static) 與實例 (Instance) 成員

類別的成員均可分成靜態與實例等兩類。成員名稱前若加上大寫的 S 或 static 保留字，則稱此成員為靜態成員。靜態成員的存取方式為「類別名稱. 成員名稱」。例如，Console 類別的 WriteLine() 即為一個靜態方法，所以此方法的使用方式如下：

```
Console.WriteLine();
```

Math 類別的 PI 亦為靜態資料成員，所以，以下敘述可印出 π 之值。

```
Console.WriteLine(Math.PI);
```

實例成員於使用前必先樣例此類別而得到一個類別變數，此類別變數亦稱為物件，往後即以「物件. 類別成員」存取類別的實例成員。例如，以下敘述可建立一個 s 物件，此物件的內容是「abc」。

```
String s="abc";
```

String 類別有一實例資料成員 Length。以下敘述可用 s 物件存取此實例成員。

```
Console.WriteLine(s.Length); //3
```

✧ 物件

類別建構完成之後，就如同是一個模型，若要存取類別的實例成員，則必須先樣例此類別，才可存取類別成員。例如：

```
int a;  
string b;
```

即是使用變數 a、b 各樣例一個 System.Int32 結構與 System.String 類別，變數 a、b 即是類別的樣例，我們定義此類別的樣例為「物件」。往後，我們均以「物件・成員」存取類別的實例成員。例如，上圖的 Int32 結構已實作 ToString 實例方法，可將數值資料轉為字串。所以：

```
int a=10;  
string b;  
b=a.ToString()+ "aa";  
Console.WriteLine (b);
```

結果是 10aa 。

其次，當我們於工具箱的某一圖項按兩下，其時就是在樣例某一類別，而產生一個類別變數，此一類別變數，我們稱為物件。例如，form1、label1、label2 與 textBox1 等都稱為物件。就如同現實社會中，每一個人 (people1、people2、people3…) 都是人類 (People) 的樣例，每一隻貓 (cat1、cat2…) 都是貓類 (Cat) 的樣例。其次，樣例後的物件名稱就以類別名稱加上樣例的先後順序序號為預設的物件名稱。物件誕生之後，當然可以重新取一個比較有意義的名稱，例如 txtx1、txty1、btnStart 等。

✧ 樣例

樣例類別的方式有兩種，分別是於工具箱使用按兩下的方式與使用者寫程式樣例物件。例如，於工具箱的 Label 按兩下，即可得 label1 物件。其次，亦可自行寫程式樣例類別。例如，以下程式，可樣例 Pen 類別，並指派其顏色與筆寬。

```
Pen pen1; //宣告 pen1 為 Pen 類別的物件  
pen1 = new Pen(Color.Red, 1); //此筆顏色是紅色，且筆寬為 1
```

✕ 繼承

任何新產品的開發，均不是無中生有，而是從舊有的產品中繼承部份或所有功能，再加入新的零件或修改部份零件而成一項新的產品。例如：SENTRA180 正是繼承 SENTRA160 而來，只是排氣量提高了、內裝豪華了，但是原來的輪胎、方向盤及座椅還是用原來 SENTRA160 的東西，這就是繼承的道理，使得新產品的開發得以縮短時程。軟體的開發何嘗不應如此？繼承的另一優點是同一方法得以讓數個新舊版本同時存在。因為當你 有新產品時，你不可能同時讓你的新舊客戶同時更新，所以您必須讓這些不同版本的方法同時存在，以滿足不同年代產品需求。就如同每次修車時，老闆會問您的出廠年代，以便找出對應的零件。

✕ 基礎類別 (Base class) 與衍生類別 (Derived class)

第一代所開發的類別，我們稱為基礎類別，有些書稱為父類別 (Parent class)、表面類別 (Super class)。繼承後的類別則稱為衍生類別，有些書稱為子類別 (Child class)、延伸類別 (Extend class) 或副類別 (Sub class)。

基本輸入與輸出

輸入與輸出是程式設計的第一步，本章將介紹 C# 的基本輸入與輸出。首先，若使用主控台開發應用程式，則僅能使用 Console 類別，關於 Console 類別請看 3-1 節。其次，若開發 Windows 應用程式，則可使用 Form、Label、TextBox 及 Button 等控制項，關於以上控制項的使用，請看 3-2 至 3-10 節。

3-1 Console類別

Console 類別的命名空間為 System，其定義如下：(如何線上查詢，請看 2-4 節的命名空間)

```
public static class Console
```

此為靜態 (static) 類別，靜態類別的使用是直接取用，例如

類別.成員

其可用方法如下圖所示 (如何查詢，請看 2-5 節)。由於都是靜態 (static) 方法，所以都是使用 " 類別.成員 " 存取這些方法。本單元僅介紹 Write、WriteLine、Read 及 ReadLine 等四個方法。

Methods

[Expand table](#)

Beep()	Plays the sound of a beep through the console speaker.
Beep(Int32, Int32)	Plays the sound of a beep of a specified frequency and duration through the console speaker.
Clear()	Clears the console buffer and corresponding console window of display information.

✕ Write 方法

Write 方法共有 18 種多載（多載請看 7-3 節），如下圖所示（僅列出 6 種多載）。它可將指定的資料輸出於主控台。本單元僅介紹兩種代表性的方法，分別是 Write(int) 與 Write(string, object)。

Write(Boolean)	Writes the text representation of the specified Boolean value.
Write(Char)	Writes the specified Unicode character value.
Write(Char[], Int32, Int32)	Writes the specified subarray of Unicode characters.
Write(Char[])	Writes the specified array of Unicode characters.
Write(Decimal)	Writes the text representation of the specified Decimal value.

```
public static void Write(Int32)
```

靜態方法，可將 int 型別的資料輸出。例如：

```
int a=5;
Console.Write(a);
```

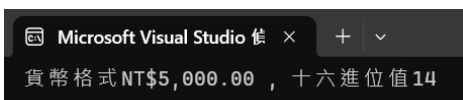
結果是 5。

```
public static void Write (String, Object)
```

此方法可將指定資料以格式化字串輸出。例如：

```
int a = 5000, b = 20;
Console.Write("貨幣格式{0:c},十六進位值{1:x}", a, b);
```

的輸出結果如下圖。



其中 {0:c} 與 {1:x} 的 0 與 1 分別表示後續所要對應的資料順序，本例 0 對到 a, 1 對到 b; c 與 x 代表輸出格式，常用的輸出格式如下表所示。

符號 (大小寫均可)	說明
c	貨幣符號
d	十進位
e	科學記號
f	小數的位數。例如 F5 表示小數取 5 位
n	顯示千分位
x	十六進位

以上 0, 1 代表後續資料的順序，也可以套上「\$」符號，直接填入變數名稱，例如，以上敘述可簡化如下：

```
Console.WriteLine($" 貨幣格式{a:c}, 十六進位值{b:x}");
```

✎ WriteLine 方法

WriteLine 方法的使用方式均同 Write 方法，只是資料輸出之後即跳一行。例如：

```
Console.WriteLine("a");  
Console.WriteLine("aa");
```

的結果是

```
a  
aa
```

而

```
Console.Write("a");  
Console.Write("aa");
```

的結果是 aaa。

✧ Read 方法

從標準輸入資料流讀取下一個字元，此方法的語法如下：

```
public static int Read();
```

例如，以下敘述可讀取單一字元，並印出其 ASCII Code。例如，鍵入 a，則印出 97；但是鍵入 abc，亦只印出 97。

```
int a;  
a=Console.Read();  
Console.WriteLine(a);
```

若要輸出所鍵入的字元，則要使用轉型。例如，以下敘述可將所鍵入的字元，以字元的形式輸出。如鍵入 a，可輸出 a。

```
a=Console.Read();  
char b=(char) a;  
Console.WriteLine(b);
```

✧ ReadLine 方法

從標準輸入資料流讀取下一行字元，此方法的語法如下：

```
public static string ReadLine();
```

以下敘述則可讀取使用者所輸入的字串。例如，輸入 mary 則輸出 mary；輸入 12 則輸出 12。

```
string s=Console.ReadLine();  
Console.WriteLine(s);
```

以下敘述可將所輸入的字串轉為整數數值。往後即可將此數值進行加減乘除等運算。

```
string s=Console.ReadLine();  
int a=int.Parse(s);  
a++;  
Console.WriteLine(a);
```

以下敘述，則可轉為 double 型別的數值。

```
double a=double.Parse(Console.ReadLine());
```

範例 3-1a

請寫一個程式，可以輸入任意三角形的三邊長 a、b、c，並輸出面積。
(本例假設輸入之三邊長可構成一三角形。三角形面積的計算公式如下：)

$$A = \sqrt{d(d-a)(d-b)(d-c)}, \text{ 其中 } d = \frac{a+b+c}{2}$$

執行結果

```
請輸入第一邊長：3
請輸入第二邊長：4
請輸入第三邊長：5
三角形的面積為：6.0
```

程式列印

```
using System;
string sa, sb, sc;
int a, b, c;
double d, Area;
// input
Console.Write(" 請輸入第一邊長：");
sa = Console.ReadLine ();
Console.Write(" 請輸入第二邊長：");
sb = Console.ReadLine();
Console.Write(" 請輸入第三邊長：");
sc = Console.ReadLine ();
//process
a = int.Parse(sa);
b = int.Parse(sb);
c = int.Parse(sc);
d = (a + b + c) / 2;
Area = Math.Sqrt(d * (d - a) * (d - b) * (d - c));
// output
Console.WriteLine(" 三角形的面積為：{0:F1} ", Area);
```

補充說明

C# 並沒有開根號與次方運算子，所以若要開根號，請使用 `Math` 類別的 `Sqrt` 方法；若要計算次方值，則應使用 `Pow` 方法，詳細請參閱 8_1 節的 `Math` 類別。

自我練習

- 請寫一程式，滿足以下條件：
 - 可以輸入兩個座標。
 - 計算此兩點座標距離。
 - 輸出此兩點距離。
 - 使用兩點式，求出過此兩點的直線方程式 $ax+by+c=0$ 。
- 假設某次考試成績資料如下：（本題先不要用迴圈與陣列）

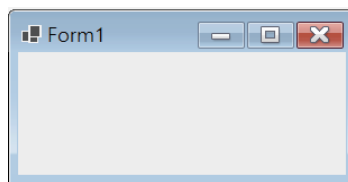
55、66、77、88、99

- 請寫一主控台應用程式輸入以上資料。
- 輸出以上資料。
- 計算總和與平均。
- 輸出總和與平均。

3-2 Form

Form（表單）如右圖所示。

Form 的主要功能是放置其它輸出入控制項，所以又稱為容器控制項（可以再放置其它控制項的控制項，稱為容器控制項）。



屬性

Form 的屬性視窗，如下圖所示。本單元並不一一介紹所有屬性，而是先將常用屬性介紹如下。其次，以下屬性亦是以下各單元控制項的常用屬性，請讀者仔細瀏覽。

BackColor

背景顏色。

Font

字型。

ForeColor

前景顏色。

Text

設定標題的文字。(VB6 、Delphi 與 Java 均使用 Caption，但是 .NET 則已統一使用 Text)。例如，以下敘述可設定標題文字為 "Hello"

```
this.Text = "Hello";    // 類別內的自稱為this
```

Location

控制項相對於容器左上角的位置。

Size

控制項的大小，單位為像素。

Visible

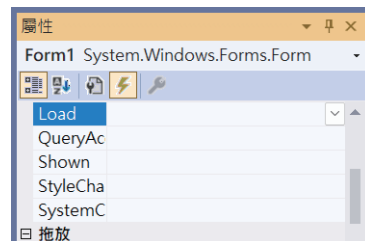
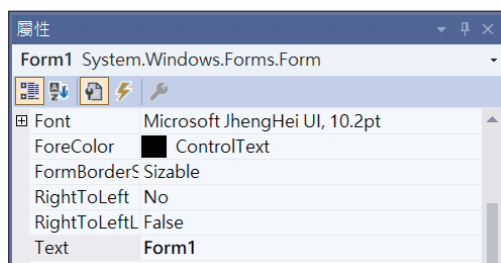
決定控制項是否可見或隱藏。

Enabled

是否啟用控制項。

事件

表單的事件視窗，如右圖所示。本書並不一一介紹，僅介紹常用事件如下：



Load

載入表單時，引發此事件。

Activated

當表單成為作用視窗時，引發此事件。

Closed

當使用者關閉表單，且表單被關閉後引發此事件。

Click

當按一下控制項時引發此事件。

DoubleClick

當按二下控制項時，引發此事件。

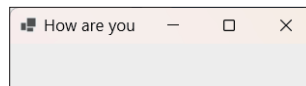
範例 3-2a

請寫一個程式，完成以下動作：

1. 表單載入時，以表單的標題顯示 "Hello"。
2. 按一下表單時，以表單的標題顯示 "How are you"。
3. 按二下表單時，結束程式的執行。

執行結果

1. 下圖左是程式執行後，表單標題出現 "Hello" 的畫面。
2. 下圖右是按一下表單後，表單標題出現 "How are you" 的畫面。
3. 按二下表單，可結束程式執行。



程式列印

```
private void Form1_Load(object sender, System.EventArgs e)
{
    this.Text = "Hello";
}
private void Form1_Click(object sender, System.EventArgs e)
{
    this.Text = "How are you";
}
private void Form1_DoubleClick(object sender, System.EventArgs e)
{
    this.Close();
}
```

3-3 Label(標籤)

Label 如下圖所示。此控制項僅可輸出結果，不能讓使用者輸入任何文字或數字。其主要功能是輸出執行結果或於設計階段填入某些文字，以作為控制項的說明。此控制項的常用屬性分別是 BackColor、ForeColor、Font、Visible、Enabled、Location 及 Size 等，此皆與 Form 相同，本單元不予重覆說明，以下僅介紹以下常用屬性。

Name

物件的名稱。 

Text

控制項所要顯示的文字。(VB6、Delphi 與 Java 均使用 Caption，但是 .NET 則已統一使用 Text)

TextAlign

決定控制項內文字的對齊方式。

AutoSize

依字型大小自動調整控制項的大小，但僅對不含換行的文字有效。

範例 3-3a

請寫一程式，完成以下功能。

1. 於程式設計階段載入一個 Label 控制項，並修改控制項所顯示的文字為 "Hello" 。
2. 表單載入時，Label 控制項顯示 "How are you" 。
3. 按一下表單時，Label 控制項顯示 "Good morning" 。
4. 按一下 Label 控制項時，Label 控制項顯示 "Good night" 。
5. 按二下 Label 控制項時，結束程式的執行。

 程式列印

```
private void Form1_Load(object sender, System.EventArgs e)
{
    label1.Text = "How are you";
}
private void Form1_Click(object sender, System.EventArgs e)
{
    label1.Text = "Good morning";
}
private void label1_Click(object sender, System.EventArgs e)
{
    label1.Text = "Good night";
}
private void label1_DoubleClick(object sender, System.EventArgs e)
{
    this.Close();
}
```

3-4 TextBox

載入 TextBox 控制項後，TextBox 如右圖所示。其預設名稱爲 textBox1，預設顯示的文字亦是 textBox1，主要功能是供使用者輸入文字或數字。本控制項的常用屬性說明如下：



Text

控制項中所包含的文字。

TextAlign

控制項中文字的對齊方式。

PasswordChar

密碼輸入時，所要顯示的文字。

ReadOnly

設定是否唯讀。

TabIndex

決定此控制項定位順序的索引。例如，你共有 3 個 `TextBox` 控制項供使用者輸入，此時可分別設定其定位順序為 0、1 及 2，則當程式執行時，定位點將會預設於 `TabIndex=0` 的控制項，待使用者完成第 0 個控制項輸入時，可按一下 `Tab` 鍵而移至第 1 個控制項。

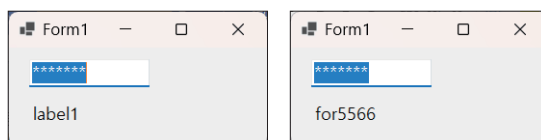
以上是 `TextBox` 控制項的常用屬性。此外，尚有 `ScrollBars`、`MultiLine`、`SelectionStart` 及 `SelectionLength` 等屬性，則待第 14-9 介紹。

範例 3-4a

請設計一密碼輸入程式，由 `TextBox` 控制項輸入密碼，當使用者按一下表單時，由 `Label` 控制項輸出密碼內容。

執行結果

下圖左為輸入密碼的畫面，所輸入的字元均以星號 (*) 顯示，下圖右是按一下表單的畫面，所輸入的密碼已由 `Label` 輸出。



物件說明

物件	類別名稱	屬性	屬性設定值	說明
Form1	Form	Text	Form1	表單
TextBox1	TextBox	PasswordChar	*	輸入字元以 "*" 顯示
		MaxLength	8	設定輸入字元上限
		Text	for5566	
Label1	Label	Text	label1	

程式列印

```
private void Form1_Click(object sender, System.EventArgs e)
{
    label1.Text = textBox1.Text;
}
```

3-5 Button

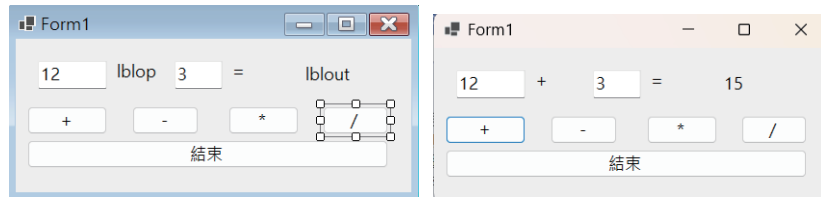
在前面幾節裡，都是使用按一下表單的方式來執行某項命令。但是這與我們的使用習慣不符，也無法同時執行多個命令，本節將介紹一個正式的功能按鈕 (Button)，以解決以上問題。Button 控制項最常使用的功能為 Click 事件，其標題亦為 Text 屬性，右圖 Text 屬性值為 button1，請看以下範例說明。(VB6 為 Caption，.NET 已統一使用 TEXT)

範例 3-5a

請寫一程式，滿足以下條件。

1. 可以輸入兩個數字。
2. 安排四個 Button 按鈕，其標題分別是 +、-、* 及 /。
3. 按一下以上四個按鈕，則可由 Label 輸出以上兩個數字相加、減、乘及除的結果。

執行結果



程式列印

```
public partial class Form1 : Form
{
    private void button1_Click(object sender, EventArgs e)
    {
        int a, b, c ;
        a = int.Parse(textBox1.Text);
        b = int.Parse(textBox2.Text);
        c = a+b ;
        label1.Text = "+";
        label3.Text = c.ToString();
    }
    private void button2_Click(object sender, EventArgs e)
    {
        int a, b, c;
        a = int.Parse(textBox1.Text);
        b = int.Parse(textBox2.Text);
        c = a - b;
        label1.Text = "-";
        label3.Text = c.ToString();
    }
    private void button3_Click(object sender, EventArgs e)
    {
        int a, b, c;
        a = int.Parse(textBox1.Text);
        b = int.Parse(textBox2.Text);
        c = a* b;
        label1.Text = "*";
        label3.Text = c.ToString();
    }
    private void button4_Click(object sender, EventArgs e)
```

```
{
    int a, b, c;
    a = int.Parse(textBox1.Text);
    b = int.Parse(textBox2.Text);
    c = a / b;
    label1.Text = "/";
    label3.Text = c.ToString();
}
private void button5_Click(object sender, EventArgs e)
{
    this.Dispose();
}
}
```

✎ 字串與數值

textBox1.Text 的型別是 string，若要取其內容進行數值計算，則應使用 int.Parse() 方法轉為數值，而數值要輸出在 string 型別的控制項，亦要使用 toString() 方法轉回 string 型別，如以上敘述。若未先轉為數值而直接執行相加運算，則為字串串接。如以下敘述：

```
string a=textBox1.Text; //12
string b=textBox2.Text; //3
string c=a+b;           //123
```

✎ 物件縮碼

控制項的命名通常是依照物件的產生順序命名，例如 textBox1、textBox2、textBox3，此種方式通常到程式設計階段都會忘記每一控制項的功能，造成不斷翻閱表單與程式的困擾，為了解決此一問題，通常必須將物件按照其功能重新命名。其次，物件名稱若能保有類別的部分名稱，則可提高程式可讀性，所以遂有物件縮碼的構想。例如，TextBox 先以 txt 為開頭命名，Label 先以 lbl 為開頭命名，Button 先以 btn 開頭命名。接著，才依照此物件的功能接續命名。例如，Button1 的功能是相加，所以命名為 btnadd，Button2 的功能是相減，所以是 btnsub，其餘依此類推。所以

以上程式，通常撰寫如下。

```
public partial class Form1 : Form
{
    private void btnadd_Click(object sender, EventArgs e)
    {
        int a, b, c ;
        a = int.Parse(txta.Text);
        b = int.Parse(txtb.Text);
        c = a+b ;
        lblOp.Text = "+";
        lblout.Text = c.ToString();
    }
    private void btnsub_Click(object sender, EventArgs e)
    {
        int a, b, c;
        a = int.Parse(txta.Text);
        b = int.Parse(txtb.Text);
        c = a - b;
        lblOp.Text = "-";
        lblout.Text = c.ToString();
    }
    private void btnmul_Click(object sender, EventArgs e)
    {
        int a, b, c;
        a = int.Parse(txta.Text);
        b = int.Parse(txtb.Text);
        c = a* b;
        lblOp.Text = "*";
        lblout.Text = c.ToString();
    }
    private void btndiv_Click(object sender, EventArgs e)
    {
        int a, b, c;
        a = int.Parse(txta.Text);
        b = int.Parse(txtb.Text);
        c = a / b;
        lblOp.Text = "/";
        lblout.Text = c.ToString();
    }
    private void btnend_Click(object sender, EventArgs e)
```

```
{  
    this.Dispose() ;  
}  
}
```

✕ 共用事件

若事件之間有許多共同的程式片段，則可使用”共用事件”處理。例如，上面的四個按鈕，都有一些程式共用，則可以寫成以下敘述。然後將屬性視窗的 button2 、button3 、button4 等 Click 事件均點選 button1_Click。

```
public partial class Form1 : Form  
{  
    private void button1_Click(object sender, EventArgs e)  
    {  
        Char n ;  
        short a, b;  
        Single c = 0 ;  
        n = Char.Parse((sender as Button).Text); //as 是轉型別運算子  
        a = short.Parse(textBox1.Text);  
        b = short.Parse(textBox2.Text);  
        switch ( n ) {  
            case '+':  
                c=a+b;  
                break;  
            case ' - ' :  
                c = a-b ;  
                break;  
            case '*':  
                c= a*b ;  
                break;  
            case '/':  
                c=a /b ;  
                break;  
        }  
        label1.Text = (sender as Button).Text ;  
        label2.Text = c.ToString();  
    }  
}
```

以上程式的

```
n = Char.Parse((sender as Button).Text);
```

as 代表轉型別運算子，可將來源的 sender 從 Object 型別轉為 Button 型別，然後再取對應到 Button 的 Text 屬性，以便進一步判別應進行何項運算。

自我練習

- 請寫一程式，滿足以下條件：
 - 可以輸入兩個座標。
 - 計算此兩點座標距離。
 - 輸出此兩點距離。
 - 使用兩點式，求出此直線方程式。

(提示：前面都是直接輸出入，現在則要尋找適當的控制項作為輸出入控制項，也要安排按鈕，作為執行的依據。)
- 輸入三角形三邊長 a、b、c，求其面積。

$$A = \sqrt{d(d-a)(d-b)(d-c)}, \text{ 其中 } d = \frac{a+b+c}{2}$$

本例假設所入的三角形三邊長可圍成三角形，例如，輸入 3，4，5 得到面積 6。(C# 並沒有平方與開跟號運算子，請用 Math.pow() 與 Math.sqrt() 代替)

- 輸入三個座標點，求此三角形面積。例如，輸入 (3,0)、(0,4)、(0,0)，將可得面積 6。

提示：

$$\text{面積} = \frac{1}{2} \begin{vmatrix} x_1, y_1 \\ x_2, y_2 \\ x_3, y_3 \\ x_1, y_1 \end{vmatrix} = \frac{1}{2} | (x_1y_2 + x_2y_3 + x_3y_1 - x_2y_1 - x_3y_2 - x_1y_3) |$$

(以上公式可繼續擴充至四點、五點…座標，但要凸多邊形，都可求其面積)

範例 3-5b

寫一個程式，可以輸入一個一元二次方程式，並求其解。(本例假設所輸入的方程式有二解，例如： $2x^2-7x+3=0$ 。其解為 $x_1=0.5, x_2=3$)

演算法則

1. 設有一元二次方程式如下：

$$ax^2 + bx + c = 0$$

2. 令 $d = \sqrt{b^2 - 4ac}$ 。

3. 則方程式二解分別為 $x_1 = \frac{-b+d}{2a}$ ， $x_2 = \frac{-b-d}{2a}$ 。

表配置與輸出結果

前面都是直接輸出入，現在則要尋找適當的控制項作為輸出入控制項，也要安排按鈕，作為執行的依據，本例表單配置如下圖左：

程式列印

```
public partial class Form1 : Form
{
    private void button1_Click(object sender, EventArgs e)
    {
        short a, b, c;
        a = short.Parse(txta.Text);
        b = short.Parse(txtb.Text);
        c = short.Parse(txtc.Text);
        double d;
        d = Math.Sqrt(Math.Pow(b, 2) - 4 * a * c);
        double x1, x2;
        x1 = (-b + d) / (2 * a);
```

```

        //(2*a)的括號若漏掉，則運算結果是錯的
        x2 = (-b - d) / (2 * a);
        lblx1.Text = x1.ToString();
        lblx2.Text = x2.ToString();
    }
}

```

自我練習

1. 寫一個程式，可以輸入一個二元一次方程式，並求其解。（本例假設所輸入的方程式恰有一解）

提示：本例演算法如下：

- (1) 設二元一次方程式如下：

$$\begin{aligned} a_1x + b_1y &= c_1 \\ a_2x + b_2y &= c_2 \end{aligned}$$

$$(2) \text{ 令 } d = \begin{vmatrix} a_1 & b_1 \\ a_2 & b_2 \end{vmatrix} = a_1b_2 - a_2b_1。$$

- (3) 其解分別是

$$x = \frac{\begin{vmatrix} c_1 & b_1 \\ c_2 & b_2 \end{vmatrix}}{d} = (c_1b_2 - c_2b_1)/d \quad y = \frac{\begin{vmatrix} a_1 & c_1 \\ a_2 & c_2 \end{vmatrix}}{d} = (a_1c_2 - a_2c_1)/d$$

- (4) 例如， $3x+y=5$ ， $x-2y=-3$ ，則其解為 $x=1$ $y=2$

2. 寫一個程式，可以輸入一個三元一次方程式，並求其解。（本例假設所輸入的方程式恰有一解）

提示，解三元一次方程式的演算法如下：

- A. 設三元一次方程式如下： $a_1x + b_1y + c_1z = d_1$

$$a_2x + b_2y + c_2z = d_2$$

$$a_3x + b_3y + c_3z = d_3$$

$$B. \text{ 令 } e = \begin{vmatrix} a_1 & b_1 & c_1 \\ a_2 & b_2 & c_2 \\ a_3 & b_3 & c_3 \end{vmatrix} = a_1b_2c_3 + a_2b_3c_1 + a_3b_1c_2 - a_3b_2c_1 - b_3c_2a_1 - c_3b_1a_2。$$

C. 則其解分別是

$$x = \frac{\begin{vmatrix} d_1 & b_1 & c_1 \\ d_2 & b_2 & c_2 \\ d_3 & b_3 & c_3 \end{vmatrix}}{e}$$

$$y = \frac{\begin{vmatrix} a_1 & d_1 & c_1 \\ a_2 & d_2 & c_2 \\ a_3 & d_3 & c_3 \end{vmatrix}}{e}$$

$$z = \frac{\begin{vmatrix} a_1 & b_1 & d_1 \\ a_2 & b_2 & d_2 \\ a_3 & b_3 & d_3 \end{vmatrix}}{e}$$

D. 例如， $3x+y+z=5$

$$2x-2y+z=1$$

$$x-y+2z=2$$

則其解為 $x=1$ ， $y=1$ ， $z=1$ 此稱為克拉馬公式。

範例 3-5c

計數器。請設計一程式，安排一個 Label 與 Button，每當使用者按一下 Button，則 Label 的顯示值增加一。

程式列印

```
public partial class Form1 : Form
{
    int a=0;
    private void button1_Click(object sender, EventArgs e)
    {
        a++;
        label1.Text = a.ToString();
    }
}
```

🔗 補充說明

本例的變數 `a` 一定要宣告在方法外面，才可保留至下一次此變數被呼叫。若將變數宣告在方法內，如以下敘述，則程式執行後，變數 `a` 永遠僅累加到 1，此即為變數的有效範圍僅止於程式所在區塊。

```
public partial class Form1 : Form
{
    private void button1_Click(object sender, EventArgs e)
    {
        int a=0;
        a++;
        label1.Text = a.ToString();
    }
}
```

🔗 自我練習

1. 請寫一程式，共包含四個按鈕，可分別遞增 1、遞增 10、遞減 1、及遞減 10。
2. 請寫一程式，滿足以下條件。
 - (1) 可以輸入兩個數字。
 - (2) 交換此兩個數字。
 - (3) 輸出交換的結果。

提示：寫出運算思維如下：

兩個數要交換，就如同兩隻手的東西要交換。所以先設定兩隻手的名稱分別為 `a` 與 `b`，現要交換其內容，則其方法如下：

- (1) 先找來第 3 隻手 `t`。
- (2) 將 `a` 手的東西交給 `t`。(此時 `a` 手才是空的)

```
t =a
```

- (3) 將 `b` 手的東西交給 `a`。

```
a =b
```

(4) 將 t 手的東西交給 b，而完成兩隻手上東西的交換。

```
b = t
```

(5) 其次，電腦並無法同時拋出兩隻手的東西，且同時接住另一隻手的東西，所以以下敘述當然得到錯誤的結果。

```
a = b
b = a
```

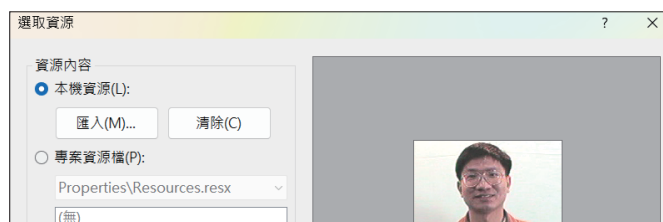
(6) 數字右旋或左旋。請安排四個 Label 與一個按鈕，每當使用者按一下按鈕，可將 label1 的值交給 label2，label2 交給 label3，label3 交給 label4，label4 的值交給 label1，此即為右旋。

3-6 PictureBox

Picture Box(圖片盒) 允許載入點陣圖 (*.bmp)、圖示 (*.ico)、中繼檔 (Metafile, *.wmf)、*.jpeg、可攜式網路圖檔 (*.png) 及 *.gif 等型態的圖形檔案。其常用屬性如下：

Image 屬性

用於設定指派所要顯示的圖形檔。於程式設計階段，只要於 Image 屬性點選所要的圖檔，即可顯示該圖檔。本例點選 D:\CsBook\gwosheng.bmp，因為是本機硬碟所以先點選 " 本機資源 "，如右圖所示：



若要於程式執行階段載入本機資源檔案圖檔，則應該使用 Image.FromFile 方法，如以下敘述：

```
PictureBox1.Image=Image.FromFile("D:\\CsBook\\gwosheng.bmp")
```

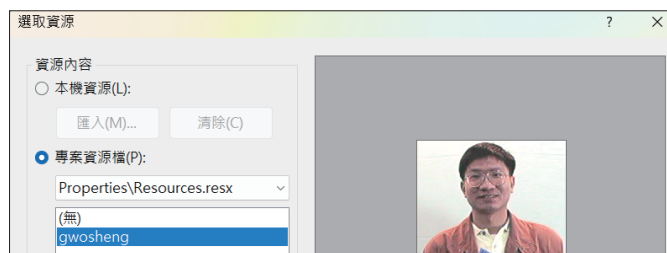
或是使用 `Bitmap` 類別的建構函式，如以下敘述

```
PictureBox1.Image=New Bitmap("D:\\CsBook\\gwosheng.bmp")
```

都可載入圖檔。其次，若要省略圖檔路徑，則應將圖檔複製到專案資料夾下的 `bin/Debug` 資料夾。

資源檔

若於執行階段需要不斷改變圖檔來源，爲了提高程式執行效率與簡化程式撰寫，請先將圖檔匯入專案資源檔，如下圖所示：



待於程式設計階段，若要載入專案資源檔，則程式如下：（請留意副檔名與路徑都已經省略）

```
pictureBox1.Image = Properties.Resources.gwosheng;
```

SizeMode

用來設定圖形的顯示模式，如下表所示：共有 5 種模式，請自行開啓範例 3-6a，觀察執行結果。

成員名稱	說明	執行結果
Normal	影像放置在 PictureBox 的左上角。如果影像大於包含它的 PictureBox，就會裁剪影像	

成員名稱	說明	執行結果
StretchImage	PictureBox 內的影像會延伸或縮小，以調整成最適合 PictureBox 的大小	
AutoSize	將 PictureBox 的大小調整成與其所包含影像的大小	
CenterImage	如果 PictureBox 大於影像，影像即置中顯示。如果影像大於 PictureBox，圖片即放在 PictureBox 的中央，而外緣被裁剪	
Zoom	以原影像的比例伸縮影像	

此屬性可於設計與執行階段調整。例如，以下程式可於執行階段設定為 Normal 顯示。

```
PictureBox1.SizeMode = PictureBoxSizeMode.Normal
```

Left, Top

控制項的位置 (x, y) 屬性，如下圖所示。若於程式執行階段改變其值，則有動畫的效果。例如，以下程式，可將控制項向右移動 10 個單位。

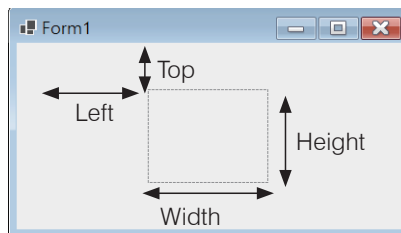
```
PictureBox1.Left = PictureBox1.Left + 10
```

Height, Width

控制項的高度與寬度屬性，如下圖所示。若於程式執行階段改變其值，則有動畫的效果。例如，

```
PictureBox1.Width= PictureBox1.Width+50
```

則可將控制項的寬度增加 50 個單位。



範例 3-6a

示範以上 `SizeMode`。(本範例程式的 `CsBook` 資料夾位置，請依自己電腦的安裝位置調整)

 程式列印

```
namespace e3_6a
{
    private void btnOpen_Click(object sender, System.
        EventArgs e)
    {
        //pictureBox1.Image = Image.FromFile("D:\\CsBook\\joky.bmp");
        pictureBox1.Image = Image.FromFile(@"D:\CsBook\joky.bmp");
        // 以上兩種方式都可以，後面的方法可抑制其使用跳脫字元
    }
    private void btnSave_Click(object sender, System.
        EventArgs e)
    {
        pictureBox1.Image.Save("D:\\CsBook\\joky_1.bmp");
        MessageBox.Show(this, "存檔成功");
    }
    private void rbNormal_Click(object sender, System.
        EventArgs e)
    {
        pictureBox1.SizeMode = PictureBoxSizeMode.Normal;
    }
    private void rbStretch_Click(object sender, System.
        EventArgs e)
    {
        pictureBox1.SizeMode = PictureBoxSizeMode.StretchImage;
    }
    private void rbAuto_Click(object sender, System
        .EventArgs e)
    {
        pictureBox1.SizeMode = PictureBoxSizeMode.AutoSize;
    }
    private void rbCenter_Click(object sender, System.
        EventArgs e)
    {

```

```
        pictureBox1.SizeMode = PictureBoxSizeMode.CenterImage;
    }
    private void btnClose_Click(object sender, System.EventArgs e)
    {
        this.Close();
    }
    // 放大
    private void button1_Click(object sender, EventArgs e)
    {
        pictureBox1.Width = pictureBox1.Width + 30;
        pictureBox1.Height = pictureBox1.Height + 30;
    }
    private void rbroom_Click(object sender, EventArgs e)
    {
        pictureBox1.SizeMode = PictureBoxSizeMode.Zoom ;
    }
    // 縮小
    private void button2_Click(object sender, EventArgs e)
    {
        pictureBox1.Width = pictureBox1.Width - 30;
        pictureBox1.Height = pictureBox1.Height - 30;
    }
    // 取用資源檔
    private void button3_Click(object sender, EventArgs e)
    {
        pictureBox1.Image = Properties.Resources.gwosheng;
    }
}
```

自我練習

1. 請寫一程式，共安排一個 PictureBox、及六個 Button，此六個 Button 可分別放大、縮小圖片，可向左、向右、向上、向下移動圖片。

3-7 Timer

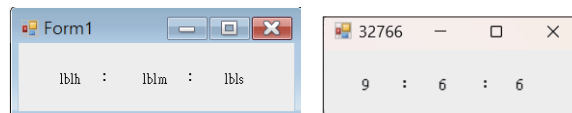
Timer(時間)控制項通常用來作為與時間有關的功能。例如，您希望若干時間自動執行一些事情，此時即可使用 Timer 物件。其最主要的屬性為 Interval，單位是 ms(千分之一秒)。其次，最常見的事件是 Tick。例如，若設定。

```
Timer1.Interval=1000
Timer1.Enabled=True
```

則程式將每隔 1 秒，自動執行 Timer1_Tick() 事件，直到 Timer1.Enable=False。

範例 3-7a

試設計一個數字時鐘或計時器。



運算思維

1. 人類關於時間的運算是，每秒加 1，逢 60 歸 0，分針加 1，分針也是逢 60 歸 0，時針加 1，時針則是逢 24 歸 0，日期加 1，但電腦並不是這樣，電腦是用 1 個長整數代表 1 個時間點，此時間點就包含年、月、日、時、分、秒。
2. 本例將先簡化一天的時間，定義為 0 到 $(60 \times 60 \times 24 - 1)$ 。則秒針、分針、時針分別如下：

```
t = (t + 1) % (24 * 60 * 60)
s = t % 60
m = (t / 60) % 60
h = t / 60 / 60
```

程式列印

```
public partial class Form1 : Form
{
    int t = 0 ;
    private void Form1_Load(object sender, EventArgs e)
    {
        t = (DateTime.Now.Hour * 60 * 60 + DateTime.Now.
            Minute * 60 + DateTime.Now.Second);
```

```
        this.Text = t.ToString() ;
        timer1.Interval = 1000 ;
        timer1.Enabled = true ;
    }
    private void timer1_Tick(object sender, EventArgs e)
    {
        int h, m, s ;
        t = ( t + 1 ) % (24 * 60 * 60) ;
        this.Text = t.ToString();
        s = t % 60;
        m = (t / 60)% 60;
        h = t / 60 / 60;
        lbls.Text = s.ToString();
        lblm.Text = m.ToString();
        lblh.Text = h.ToString();
    }
}
```

自我練習

1. 請設計一程式，於表單載入時讓 PictureBox 自動由小逐漸變大。
2. 請設計一程式，於表單載入時讓 PictureBox 逐漸向右移動，當跑到最右邊時，亦可回到最左邊，
3. 計時器。請設計一個程式，當按鈕被按時，開始計時，再按一下時，傳回計時時間。

3-8 Random 類別

日常生活的猜拳、擲骰子及電腦開獎等活動，這些遊戲都必須藉由亂數的產生。Random 類別提供一些關於亂數的產生方法。Random 的定義如下：

```
public class Random
```

是一個實例類別，實例類別要先樣例，才可使用。

✧ 建構子

Random 類別的建構子共有兩種多載，分別是 `public Random()` 與 `public Random(int)`，前者是使用系統時間當亂數的起點，後者是自己給一個特殊的整數當亂數起點。例如：前面 3-2 到 3-7 節，都是使用控制項的方式，使用者於工具箱按兩下，就可以產生物件，但以下各節，則要自己撰寫程式，產生物件。

```
Int a;  
Random r = new Random ();  
           //使用系統時間當亂數起點，系統時間一直在改變，所以每次起點都不同  
Random r = new Random (34); //自己規定起點，所以每次起點都相同，請自行觀察
```

✧ 實例方法

本單元將介紹以下 3 個常用的實例方法，分別是 `Next()`、`NextBytes()` 及 `NextDouble()`。

Next

`Next()` 共有 3 種多載，分別說明如下：

► `public virtual int Next();`

傳回大於等於 0 且小於 `MaxValue`（註：`MaxValue=2147483647`）的整數。例如，以下敘述可建立一個 `r` 物件，並產生一個 0 至 `MaxValue` 之間的亂數。

```
Random r=new Random();//樣例一個物件實體r  
Console.WriteLine ( r.Next( ) ) ;//透過物件實體r，取用其方法
```

請留意前面的 `Console` 是靜態類別，靜態類別是使用

類別.方法

取用其方法。`Random` 則是實例類別，實例類別要先使用物件樣例其類別，然後再使用

物件.方法

取用其方法。

► `public virtual int Next(int);`

此方法的語法如下：

```
public virtual int Next(int maxValue);
```

其中 `maxValue` 必須小於等於 `MaxValue`。其次，本方法可傳回一個整數亂數，此數大於等於 0 且小於所輸入的 `maxValue`。例如，以下敘述可傳回 0，1，2，3 之任一數。

```
Console.WriteLine(r.Next(4)); // r 物件同上
```

► `public virtual int Next (int,int);`

此方法的語法如下：

```
public virtual int Next(  
    int minValue,  
    int maxValue  
);
```

可傳回一個亂數，此亂數大於等於 `minValue` 且小於 `maxValue`。例如，以下敘述可傳回 1 至 42 之間的亂數。

```
console.WriteLine (r.Next(1, 43)); // r 物件同上
```

NextBytes()

`NextBytes()` 的語法如下：

```
public virtual void NextBytes(byte[]buffer);
```

其中，`buffer` 的型別是 `byte[]`。此方法可將 `buffer` 填入 0 至 255 之間的整數。例如：

```
byte[] b=new byte[4]  
r.NextBytes(b);  
foreach (byte i in b)  
    Console.WriteLine(i);
```

將可得 4 個 0 至 255 之間的亂數。

NextDouble()

此方法的語法如下：

```
public virtual double NextDouble();
```

可傳回一個浮點亂數，此亂數大於等於 0.0 且小於 1.0 。例如：

```
Console.WriteLine(r.NextDouble());
```

將可得一個亂數，此亂數大於等於 0.0 且小於 1.0 。

範例 3-8a

請寫一程式，可以擲兩顆骰子，輸出結果。

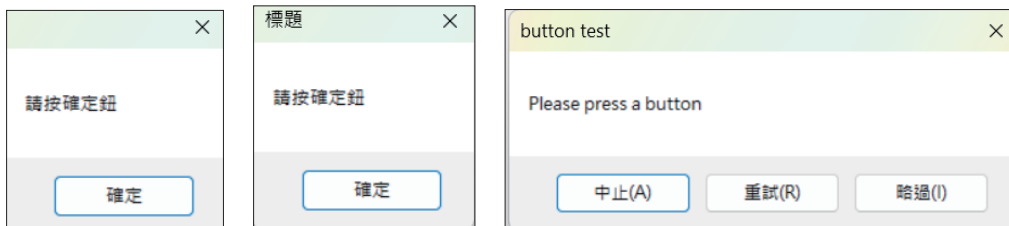
程式列印

```
private void button1_Click(object sender, EventArgs e)
{
    int a, b;
    Random r = new Random (); // 自己寫程式，產生物件
    a = r.Next(1, 7);
    b = r.Next(1, 7);
    label1.Text = a.ToString ();
    label2.Text = b.ToString ();
}
```

3-9 輸出入對話框

✎ MessageBox

MessageBox 類別的主要功能是顯示一個對話框，並等待使用者確認，如以下各圖。



前面的 `Random` 類別是實例類別，要自己樣例產生物件，再以物件.方法，存取其方法，本節的 `Message` 類別則是靜態類別，靜態類別直接以類別名稱.方法，存取物件方法，不用先樣例，請看以下說明。

`MessageBox` 類別的常用方法為 `Show`，此方法請自行線上查詢。

本書僅介紹以下三種多載，分別是 `MessageBox.Show(String)`、`MessageBox.Show(String,String)`、`MessageBox.Show(String,String,MessageBoxButtons)`。

`MessageBox.Show (String)`

顯示 `String` 的內容，並等待使用者按一下「確定」按鈕。例如，上頁圖左的程式如下敘述：

```
MessageBox.Show("請按確定鈕");
```

`MessageBox.Show (String1 , String2)`

以 `String2` 當標題，且顯示 `String1` 的內容，並等待使用者按一下 " 確定 " 按鈕。例如，上頁圖中的程式如下：

```
MessageBox.Show("請按確定鈕","標題");
```

`MessageBox.Show (String , String,MessageBoxButtons)`

`MessageBoxButtons` 是一個列舉成員，成員名稱如下：

成員名稱	說明
AbortRetryIgnore	訊息方塊包含 中止(A) 、 重試(R) 和 略過(I) 按鈕
OK	訊息方塊包含 確定 按鈕
OKCancel	訊息方塊包含 確定 和 取消 按鈕
RetryCancel	訊息方塊包含 重試(R) 和 取消 按鈕
YesNo	訊息方塊包含 是(Y) 和 否(N) 按鈕
YesNoCancel	訊息方塊包含 是(Y) 、 否(N) 和 取消 按鈕

範例 3-9a

示範以上 MessageBox。

程式列印

```
private void button1_Click(object sender, EventArgs e){
    MessageBox.Show("請按確定鈕");
}
private void button2_Click(object sender, EventArgs e){
    MessageBox.Show("請按確定鈕", "標題");
}
private void button3_Click(object sender, EventArgs e){
    DialogResult dr = new DialogResult();
    dr=MessageBox .Show ("Please press a button ","button test",
        MessageBoxButtons .AbortRetryIgnore );
    button3 .Text ="You presses " +dr.ToString ()+" button";
}
```

Interaction

Interaction 類別包含用來與物件、應用程式和系統互動的程序。其成員請自行線上查詢。此類別隸屬 Microsoft.VisualBasic 命名空間，所以使用此類別前，請先載入其命名空間如下：

```
using Microsoft.VisualBasic;
```

本單元僅介紹 Beep()、Shell()、InputBox() 等方法。

► Beep()：讓電腦發出嗶一聲。例如，

```
Interaction.Beep();
```

Interaction 也是靜態類別，也是以「類別名稱.方法」存取物件的方法。

► Shell()：執行某一應用程式。例如，以下程式可以啟動記事本程式。

```
Interaction.Shell("C:\\Windows\\notepad.exe");
```

► InputBox()：可出現一個輸入對話框，InputBox() 語法如下：

```
public static string InputBox (  
    string Prompt,  
    [OptionalAttribute] string Title,  
    [OptionalAttribute] string DefaultResponse,  
    [OptionalAttribute] int XPos,  
    [OptionalAttribute] int YPos  
)
```

例如，請鍵入以下程式，並觀察執行結果：

```
Interaction.InputBox("Prompt", "Title", "DefaultResponse", 100, 50);
```

範例 3-9b

示範以上 Interaction 類別。

程式列印

```
private void button1_Click(object sender, EventArgs e){  
    Interaction.Beep();  
    Interaction.Shell("C:\\Windows\\notepad.exe");  
    Interaction.InputBox("Prompt", "Title", "DefaultResponse");  
}
```

自我練習

1. 請寫程式，可連續用 InputBox 輸入兩數，並用 MessageBox 輸出相加結果。

3-10 實例探討

✧ 複數

複數的表示法有直角座標的標準式與極座標的極式，兩者的關係如下：

$$\begin{aligned}\vec{A} &= A \angle \theta = a + bi \\ \Rightarrow A &= \sqrt{a^2 + b^2} \\ \theta &= \tan^{-1} \frac{b}{a}\end{aligned}$$

範例 3-10a

示範複數的標準式轉為極式。

✧ 執行結果

✧ 程式列印

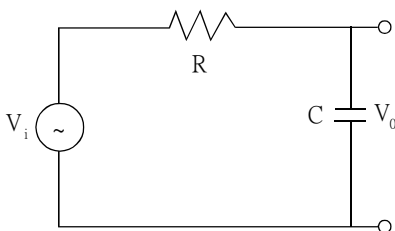
```
private void button1_Click(object sender, EventArgs e)
{
    double a, b;
    double r, s;
    a=double.Parse (txta.Text);
    b=double.Parse (txtb.Text);
    r=Math.Sqrt(a * a + b * b);
    s = Math.Atan(b / a); // 徑度量
    s = Math.Round(s * 180 / Math.PI, 1); // 度度量，四捨五入到
    小數點第一位
    txtr.Text=r.ToString ();
    txts.Text=s.ToString ();
}
```

自我練習...

1. 請寫一程式，可以將極座標轉為直角座標。

※ 範例 3-10b (非資訊本科系可先跳過)

示範以下電壓增益的計算。(技藝競賽考古題)



上圖是一個低通濾波器電路，這個電路包括一個電阻 R 和一個電容器 C ，其電壓增益為輸出電壓 V_0 和輸入電壓 V_i 的比值，可以表示為 $V_0 / V_i = 1 / (1 + j2\pi fRC)$ ，其中 f 表示輸入正弦波的頻率，單位為赫芝 (Hz)，電阻 R 的單位為歐姆 (Ω)，電容 C 的單位為法拉 (F)， j 表示 $\sqrt{-1}$ 。請您寫一個程式，求出低通濾波器的增益大小 $z = 20\log_{10}|V_0 / V_i|$ 。(電子學實例)

例如：輸入電阻值 $R = 1600$ (單位為歐姆)，輸入電容值 $C = 0.000001$ (單位為法拉)，輸入頻率值 $f = 1000$ (單位為赫芝)，則輸出濾波器的大小 $Z = -20.089\text{dB}$ ，濾波器的相角 $\theta = -84.319^\circ$ 。

演算法則...

$$\begin{aligned} \frac{V_0}{V_i} &= \frac{1}{1 + j2\pi fRC} \\ &= \frac{1 - j2\pi fRC}{1 + (j2\pi fRC)^2} \\ &= \frac{1}{1 + (2\pi fRC)^2} - j \frac{2\pi fRC}{1 + (2\pi fRC)^2} \end{aligned}$$

指派 $a = 2\pi fRC$ ，則上式簡化如下：

$$= \frac{1}{1+a^2} - j \frac{a}{1+a^2}$$

指派 $b = \frac{1}{1+a^2}$ ， $d = \frac{a}{1+a^2}$ ，則上式簡化如下：

$$= b - jd$$

上式的大小如下：

$$g = \sqrt{b^2 + d^2}$$

$$\text{電壓增益的大小} = 20 \log_{10} \left| \frac{V_o}{V_i} \right|$$

$$= 20 \log_{10} g$$

$$\text{電壓增益的相角} = \tan^{-1} \frac{-d}{b}$$

程式列印

```
private void Button1_Click(object sender, EventArgs e)
{
    int r = 1600 ;
    double c = 0.000001 ;
    int f = 1000;
    double a = 2 * Math.PI * f * r * c ;
    double b = 1 / (1 + a * a) ;
    double d = a / (1 + a * a);
    double g = Math.Sqrt(b * b + d * d);
    double h = 20 * Math.Log10(g);
    double i = Math.Atan2(-d, b) * 180 / Math.PI ;
    Label1.Text = h.ToString();
    Label2.Text = i.ToString();
}
```

* 範例 3-10c (非資訊本科系可先跳過)

有一 RLC 串連電路， $R=300$ ， $L=10\text{mH}$ ， $C=0.1\mu\text{F}$ ， $V_s=2.8\sin(2*\pi*f*t)$ ， $\pi=3.14$ ， $f=9140\text{Hz}$ ，請寫一程式，計算 $\vec{Z}$ ， $\vec{I}$ 與 $\vec{V}_L$ 。以上的 $\vec{Z}$ 表示向量，答案既然是向量，請含大小與角度。(技藝競賽考古題，基本電學實例)

執行結果

演算法則

$$V_p = 2.8$$

$$R = 300$$

$$L = 10\text{m}$$

$$c = 0.1\text{u}$$

$$\text{PI} = \text{Math.PI}$$

$$X_L = 2\pi fL \quad \& \quad \bar{X}_L = X_L \angle 90^\circ$$

$$X_C = \frac{1}{2\pi fC} \quad \& \quad \bar{X}_C = X_C \angle -90^\circ$$

$$\bar{Z} = \bar{R} + \bar{X}_L + \bar{X}_C$$

$$= R \angle 0^\circ + X_L \angle 90^\circ + X_C \angle -90^\circ$$

$$= R \angle 0^\circ + (X_L - X_C) \angle 90^\circ$$

$$\Rightarrow Z = \sqrt{R^2 + (X_L - X_C)^2} \quad Z\angle = \tan^{-1} \frac{(X_L - X_C)}{R}$$

$$\bar{V} = \frac{V_p}{\sqrt{2}} \angle 0^\circ \quad \Rightarrow \quad V = \frac{V_p}{\sqrt{2}} \quad \text{VA} = 0 \quad (\text{VA 代表電壓的角度})$$

$$\bar{I} = \frac{\bar{V}}{\bar{Z}} \quad \Rightarrow \quad I = \frac{V}{Z} \quad \text{IA} = \text{VA} - \text{ZA} \quad (\text{IA 代表電流的角度})$$

$$\bar{V}_L = \bar{I} \cdot \bar{X}_L \quad \Rightarrow X_L = I \cdot X_L \quad X_L\angle = \text{IA} + 90^\circ$$

程式列印

```
private void cmdstart_Click(object sender, EventArgs e)
{
    double vp, r, l, c, m, u, pi, f ;
    double v, va, xl, xc, z, za, i, ia, vl, vla ;
```

```

m = Math.Pow(10 , (-3));
u = Math.Pow(10 , (-6)) ;
vp = 2.8 ;// 峰值
v = vp / 1.4 ;// 有效值
va = 0 ; // 角度
f = 9140; // 頻率, Hz
r = 300 ;// 電阻
l = 10 * m ;// 電感
c = 0.1 * u; // 電容
pi = Math.PI;
xl = 2 * pi * f * l; // 感抗
xc = 1 / (2 * pi * f * c); // 容抗
z = Math.Sqrt(r * r + Math.Pow( (xl - xc) , 2)); // 阻抗大小
za = Math.Atan((xl - xc) / r); // 阻抗大小
lblz.Text = Math.Round(z).ToString();
lblza.Text = Math.Round(za * 180 / pi).ToString();
// 將徑換成度
i = v / z;
ia = va - za;
lbli.Text = Math.Round(i, 4).ToString();
lbliia.Text = Math.Round(ia * 180 / pi).ToString();
// 將徑換成度
vl = i * xl;
vla = pi / 2 + ia;
lblvl.Text = Math.Round(vl, 1).ToString();
lblvla.Text = Math.Round(vla * 180 / pi).ToString();
// 將徑換成度
}

```

✕ Complex 類別

C# 有複數類別，可以直接進行複數運算，請鍵入以下程式，觀察執行結果。

```

using System;
using System.Numerics;
Complex a = new Complex(3, 4); //表示3+j4
Complex b = new Complex(3, 4); //表示3+j4
Complex c = a + b; //請自行修改為-、*、/

```

```
double d = Complex.Abs(c);  
double f = c.Phase * 180 / Math.PI;  
Console.WriteLine(d); //大小  
Console.WriteLine(f); //角度
```

自我練習...

1. 請將以上範例 3-10a、3-10b、3-10c，以 Complex 類別重做。
2. 技藝競賽模擬題。泉勝基本電學下冊範例 9-2 與 9-3a，原題目使用 Python 的 complex 計算交流電流與繪圖，請自行使用 C# 完成以上計算與繪圖，C# 繪圖，請看第十六章。

習題



- 請設計一程式，可以計算兩個以標準式 $(3+4i)$ 表示的複數的加、減、乘、除等運算。假設 $\bar{P} = a+bi$ ， $\bar{Q} = c+di$ ，則以上運算方法如下：
 - $\bar{P} + \bar{Q} = (a+c) + (b+d)i$
 - $\bar{P} - \bar{Q} = (a-c) + (b-d)i$
 - $\bar{P} * \bar{Q} = (ac-bd) + (ad+bc)i$
 - $\bar{P} / \bar{Q} = ((ac+bd) + (bc-ad)i) / (a^2 + d^2)$
 (本題也請練習使用 Complex 類別)
- 請設計一程式，可以計算兩個以極式 $(5 \angle 30)$ 表示的複數的加、減、乘、除等運算。假設 $\bar{P} = r_1 \angle \theta_1$ ， $\bar{Q} = r_2 \angle \theta_2$ ，則以上運算方法如下：
 - $\bar{P} + \bar{Q}$ 與 $\bar{P} - \bar{Q}$ (請善用 Complex 才能省事，測試資料 $20 \angle -60 - 20 \angle 60 = 34.6 \angle -90^\circ$)
 - $\bar{P} * \bar{Q} = (r_1 * r_2) \angle (\theta_1 + \theta_2)$
 - $\bar{P} / \bar{Q} = (r_1 / r_2) \angle (\theta_1 - \theta_2)$
- 請分四次輸入 1 個 0 到 9 的整數，並將它合併為 1 個整數。例如，輸入 1，輸入 2，輸入 3，輸入 4，則輸出 1234。(請先不用迴圈)
- 同上習題，請先輸入 3 個 0 到 9 的整數，再輸入兩個 0 到 9 的整數，並將其合併為 1 個浮點數。例如，輸入 1，輸入 2，輸入 3，輸入 4，輸入 5，則輸出 123.45。(請先不用迴圈)
- 請輸入 1 個 4 位數，並將其分解輸出如 $a_1 * 10^3 + a_2 * 10^2 + a_3 * 10^1 + a_4$ 。例如，輸入 1234，則輸出 $1 * 10^3 + 2 * 10^2 + 3 * 10^1 + 4$ 。(請先不用迴圈)
- 請輸入一個 0 到 15 的整數，並將其轉為 2 進位輸出。例如，輸入 12，則輸出 $(12)_{10} = (1100)_2$ 。(請先不用迴圈)
 10 進位轉 2 進位的方法如下，一直除以 2，直到商為 0，例如 12 轉 2 進位的過程如下：

$$12/2=6..0$$

$$6/2=3..0$$

$$3/2=1..1$$

$$1/2=0..1$$
 商為 0，計算結束，其 2 進位為 1100，先出來的餘數放在最右邊。以上是 12 的轉換，待學完迴圈，可用迴圈完成任意數值的轉換。

MEMO

決策敘述

人類的生活必須不斷面對決策問題，連我家一個不到三歲的小孩，也常要思考他手裡的十元是要坐電動車還是買棒棒糖。程式語言是協助解決人類問題的工具，當然也有決策敘述，C# 依決策點的多寡，分為以下兩種決策敘述，第一是兩雙向分歧決策 `if else`，例如肚子餓了就吃飯，否則繼續前進；第二是多向分歧決策的 `switch`，例如你身上有 5000 元，走進一家五星級的大飯店用餐，你的分歧點就很多，有自助餐、中式套餐、日本料理、泰國餐點等等分歧點。本章的重點即是探討 C# 的決策敘述。

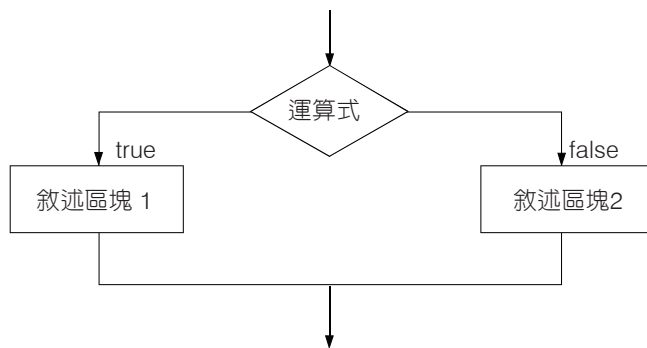
4-1 if...else

日常生活裏，常出現「假如～則～，否則～」，此種決策模式有兩種解決問題的方案，故稱為雙向分歧決策，此時可使用 `if...else` 敘述，`if...else` 敘述的語法如下：

```
if (運算式)
{
    敘述區塊1 ;
}
else
{
    敘述區塊2 ;
}
```

以上語法說明如下：

1. 運算式的值若為 `true`，則執行敘述區塊 1; 運算式的值若為 `false`，則執行敘述區塊 2, 其流程圖如下：



2. 敘述區塊內可以放置任何合法敘述，當然也可以再放置 `if`; `if` 中有 `if`, 稱為巢狀 `if`, 請看範例 4-1c。
3. 敘述區塊上下一一定要用大刮號 `{}` 包圍。例如，以下敘述可依 `a` 的大小評量其及格與否，且統計及格和不及格人數。

```
if (a >= 60) {  
    b = "及格";  
    c = c+1;  
}  
else {  
    b = "不及格";  
    d = d+1;  
}
```

4. 敘述區塊內的敘述若只有一個，則敘述區塊上下兩個大括號可予省略。例如，以上程式可簡化如下：

```
if (a>=60)  
    b = "及格";  
else  
    b = "不及格";
```

5. 有時爲了簡化程式的撰寫，可將否則的部分寫在 if 前面，並省略 else。例如，以下程式同義於上面程式。

```
b = "不及格";  
if(a>=60) {  
    b = "及格";  
}
```

6. 若敘述區塊上下的大括號遺漏，則條件成立時，僅執行敘述區塊的第一個敘述，但更嚴重的問題是，敘述區塊的第二個敘述，不論條件成立與否，均會自動執行。例如，以下敘述執行之後，b=0,c=3。

```
a=2;b=0;c=0;  
if(a==0)  
    b=3;  
    c=3;  
Console.WriteLine(b); // b=0  
Console.WriteLine(c); // c=3
```

7. if() 後面不要緊接分號，例如，

```
a=2;b=0;c=0;  
if(a==0);  
    b=3;  
    c=3;  
Console.WriteLine(b); // b=3  
Console.WriteLine(c); // c=3
```

這是初學者常犯的錯誤，因爲 if 遇到分號 (;)，表示此指令已經結束。

8. C# 爲了強調語法的簡潔性，若決策的結果，只爲求得簡單的運算式，則可用下式的三元運算子 (?:)，其中運算式 2 與運算式 3 的括號只是增加程式的可讀性，此兩括號的有無並不影響程式的執行結果。以下式子，當運算式 1 的值爲 true 時，Z= 運算式 2，當運算式 1 的值爲 false 時，則 Z= 運算式 3，請看範例 4-1b。

```
Z = (運算式 1) ? (運算式 2) : (運算式 3)
```

範例 4-1a

請寫一程式，可評判成績是否及格。

題目分析

本例是典型的雙向分歧的決策，此時可用 if...else 實現程式的要求。

程式列印

```
int a; string b;
Console.Write("請輸入成績：");
a = int.Parse(Console.ReadLine());
if (a >= 60)
    b = "及格";
else
    b = "不及格";
Console.WriteLine(b);
```

範例 4-1b

請使用三元運算子，重寫以上範例。

程式列印

```
string b;
Console.Write("請輸入成績：");
int a = int.Parse(Console.ReadLine());
b = (a >= 60) ? ("及格") : ("不及格");
Console.WriteLine(b);
```

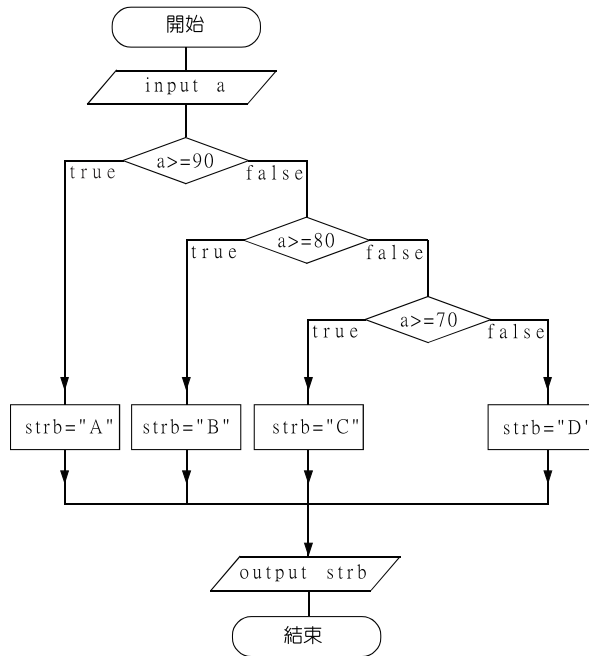
範例 4-1c

請寫一個程式，完成以下要求：

1. 輸入一個 0 ~ 100 的分數。
2. 當分數大於 90 時，輸出 A。
3. 當分數介於 80 ~ 89 時，輸出 B。
4. 當分數介於 70 ~ 79 時，輸出 C。
5. 當分數介於 0 ~ 69 時，輸出 D。

題目分析

1. 使用流程圖分析如下：



2. 以上每一個決策點，都有兩個分歧點，所以適用 if ~ else，每一個 else 後面均需再放置 if 作進一步決策。

程式列印

```
string strb = "";
Console.Write("請輸入成績：");
string s1 = Console.ReadLine();
int a = int.Parse(s1);
if (a >= 90)
    strb = "A";
else
    if (a >= 80)
        strb = "B";
    else
        if (a >= 70)
            strb = "C";
        else
            strb = "D";
```

```
        strb = "D";  
        //Console.WriteLine("您的成績等級爲：{0},strb");  
        Console.WriteLine($"您的成績等級爲：{ strb }");
```

補充說明

1. 本程式亦有同學寫成如下，這樣雖然效果相同，但是不管分數是多少，都要執行四次，此爲不好的方法。

```
string strb = "";  
Console.Write("請輸入成績：" );  
string s1 = Console.ReadLine();  
int a = int.Parse(s1);  
if (a >= 90)  
    strb = "A";  
if (a<90 & a >= 80)  
    strb = "B";  
if (a <80 & a >= 70)  
    strb = "C";  
if (a<60)  
    strb = "D";  
Console.WriteLine(" 您的成績等級爲： {0}", strb);
```

2. 另一種寫法如下，這樣可以減少比較次數，提高效率。

```
string strb = "";  
Console.Write("請輸入成績：");  
string s1 = Console.ReadLine();  
int a = int.Parse(s1);  
if (a >= 80)  
    if (a >= 90)  
        strb = "A";  
    else  
        strb = "B";  
else  
    if (a>=70)  
        strb = "C";  
    else  
        strb = "D";  
Console.WriteLine("您的成績等級爲：{0}", strb);
```

自我練習

1. 假設自來水費率如下：
 - A. 100 以下，每度 3 元。
 - B. 100～300 度，超過 100 度的部分，每度 5 元。例如，用水 200 度，則費用等於 $100 \times 3 + 100 \times 5 = 800$ 元。
 - C. 300 度以上，超過 300 度的部分，每度 6 元。例如，用水 400 度，則費用等於 $100 \times 3 + 200 \times 5 + 100 \times 6 = 1900$ 元。請寫一個程式，可以計算自來水費。
2. 請寫一個程式，可以判斷所輸入座標的所在象限。(應先判斷是否落在原點或座標軸)
3. 請寫一個主控台或 Windows Form 應用程式，滿足以下條件。
 - A. 輸入密碼。(若是 Windows Form 程式，則出現輸入密碼對話框)
 - B. 檢查密碼是否等於「123」，若「是」則輸出「密碼正確」並繼續執行自我練習 2。
 - C. 密碼錯誤，則告知「密碼錯誤」，若錯誤三次，則程式結束。
4. 直線。直線標準式為 $ax+by+c=0$ ，請寫一程式，可以輸入一直線係數 a, b, c 。其次，可再輸入任一點座標，並判斷所輸入點是否在直線上。

範例 4-1d

請寫一個程式，可以使用亂數輸出任意字元，由使用者輸入此字元，電腦評判是否正確。

程式列印

```
Random r = new Random();
int a = r.Next(97, 122);
Console.WriteLine((char)a);
int b=Console.Read();
string result;
if (b == a)
    result = "Right";
else
    result = "Wrong";
Console.WriteLine(result);
```

 自我練習

1. 線上心算練習。請設計一個程式，可以自動產生二個一位數，讓使用者練習心算加法練習，電腦並評定是否正確。
2. 請寫一程式，可以判斷所指派的數值是奇數或偶數。
3. 請寫一程式，滿足以下條件：

- (1) 指派兩個數。
- (2) 求指派兩數極大值。
- (3) 輸出極大值。

提示：寫出運算思維

- (1) 指派第一數，本例以變數 a 儲存。
- (2) 指派第二數，本例以變數 b 儲存。
- (3) 設定極大值 (max) 為第一數。

```
max=a;
```

- (4) 當第二數 (b) 大於極大值時，極大值即以 b 取代。

```
if (b>max)  
    max=b;
```

- (5) 輸出極大值 (max) 即為所求。
 - (6) 根據以上運算思維完成程式。
4. 請寫一程式，滿足以下條件：
 - A. 輸入三個數。
 - B. 求三個數極小值。
 - C. 輸出極小值。

提示：寫出運算思維如下：

- (1) 輸入第一個數，本例以變數 a 儲存。
- (2) 輸入第二個數，本例以變數 b 儲存。
- (3) 輸入第三個數，本例以變數 c 儲存。
- (4) 設定極小值 (min) 為第一數。

(5) 當第二數 (b) 小於極小值時，極小值即以 b 取代。

```
if (b<min)
    min = b;
```

(6) 當第三數 (c) 小於極小值時，極小值即以 c 取代。

```
if (c<min)
    min = c;
```

(7) 輸出極小值 min。

(8) 根據以上運算思維完成程式。

(9) 以上是 3 筆資料，資料數量若增加，則應改用陣列與迴圈。

5. 請寫一程式，可以輸入 3 個人名與分數，並找出最高分的人名與分數。

6. 請寫一程式，滿足以下條件。

(1) 指派三個數。

(2) 將此三個數由小而大輸出。

提示：寫出運算思維如下：

(1) 分別以 a、b 及 c 表示欲排序的資料。

(2) 假如 a 大於 b，則 a 與 b 交換，如下圖的 (1)。

(3) 假如 b 大於 c，則 b 與 c 交換，如下圖的 (2)。

(4) 假如 a 大於 b，則 a 與 b 交換，如下圖的 (3)，此時排序完成，共需進行 3 次的比較與交換，如下圖所示，此即為氣泡排序法。

a	b	c
(1)	(2)	
(3)		

(5) 請根據以上運算思維，完成程式。

7. 請寫一程式，可以指派 5 個人名與分數，並可將此人名與分數，由小而大排列輸出。

補充說明

1. 若有 4 筆資料需要排序，則共需進行 6 次比較與交換，如下圖所示。

a	b	c	d
(1)	(2)	(3)	
(4)	(5)		
(6)			

2. 若有 5 筆資料需排序，則共需進行 10 次比較與交換，如下圖所示。

a	b	c	d	e
(1)	(2)	(3)	(4)	
(5)	(6)	(7)		
(8)	(9)			
(10)				

3. 以上為 3、4 或 5 筆資料的比較與排序，其比較與交換的次數尚可克服，但若欲排序的資料超過 5 個，例如 20 筆資料欲排序，則應待迴圈與陣列敘述介紹以後，才有較快速的解法。

4-2 switch...case

一個決策點若同時擁有三個或三個以上的解決方案，則稱此為多向分歧決策。多向分歧決策雖也可使用範例 4-1c 的巢狀 if..else 解決，但卻增加程式的複雜度及降低程式可讀性，若此一決策點能找到適當的運算式，能使問題同時找到分歧點，則可使用 switch case 敘述。switch case 語法如下：

```
switch (運算式)
{
    case 常數1:
        敘述區塊1;
        break ;
    case 常數2:
```

```
        敘述區塊2;  
        break ;  
    case...  
    [default:  
        敘述區塊n;]  
}
```

以上語法說明如下：

1. switch 的運算式值僅能整數或字串。
2. case 的常數僅能整數或字串。
3. 電腦將會依 switch 的運算式值，逐一至常數 1、常數 2 尋找合乎條件的 case，並執行相對應的敘述區塊，直到遇到 break 敘述，才能離開 switch。
4. default 可放置特殊情況，其兩旁加中括號表示此敘述可省略；若省略 default，且若沒有任何 case 滿足 switch 運算式，則程式會默默離開 switch 敘述。
5. 敘述區塊可放置任何合法的敘述，當然也可放置 switch 或 if。
6. 例如，以下敘述，可將 0,1,2 轉為對應的剪刀、石頭、布。

```
int a = 1;string b="";  
switch (a){  
    case 0:  
        b = "剪刀";  
        break;  
    case 1:  
        b = "石頭";  
        break;  
    case 2:  
        b = "布";  
        break;  
    default:  
        b = "輸入錯誤";  
        break;  
}  
Console.WriteLine(b);
```

7. 以下程式 case 0：漏掉 break，電腦會出現錯誤訊息。

```
switch (a){  
    case 0:  
        b = "剪刀";  
    case 1:  
        b = "石頭";  
        break;  
}
```

範例 4-2a

試以 switch case 重作範例 4-1c。

程式列印

本例若將分數先除以 10，則僅剩 0..10 的整數，所以可以根據結果評判等級，程式如下：

```
int a = 88; string strb;  
switch ((int)(a / 10)){  
    case 10:  
    case 9:  
        strb = "A";  
        break;  
    case 8:  
        strb = "B";  
        break;  
    case 7:  
        strb = "C";  
        break;  
    default://分數在70分以下，則由default處理  
        strb = "D";  
        break;  
}  
Console.WriteLine("您的成績等級為： {0}", strb );
```

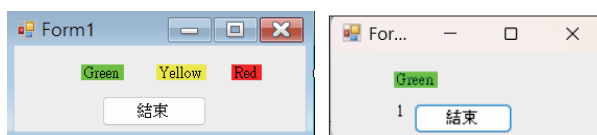
程式說明

1. 有些語言可用逗號將兩種 case 放在一起，但 C# 中每一 case 僅能放置一個常數，所以若兩個或兩個以上 case，有相同的處理方法，則應將兩個 case 分成兩個敘述。

範例 4- 2b

請寫一個紅綠燈控制程式。

表單配置與執行結果



操作步驟

1. 爲了簡化程式的撰寫，本例先假設一個流程共 15 個時序。各時序所要顯示的燈號如下：
 - A. 時序 0、1、2、3、4，顯示「綠燈」。
 - B. 時序 5、7，顯示「綠滅」。
 - C. 時序 6、8，顯示「綠燈」。
 - D. 時序 9、10，顯示「黃燈」。
 - E. 時序 11、12、13、14，顯示「紅燈」。
2. 本例共 15 個時序，所以採用求餘數 (%) 運算子。

```
i = (i + 1) % 15;
```

如此才能控制 i 必在 0 至 14 之間轉動。

3. 每個時序 0.5 秒，所以採用 Timer 控制項的 Tick 事件。
4. 紅綠燈的輸出採用 3 個 Label 控制項，物件名稱分別爲 lblGreen、lblYellow、lblRed，並逐一將每一控制項的屬性 Visible 設爲 true，即可改變燈號。

5. 爲了使變數 i 的有效範圍，能持續至下一次 timer1_Tick，所以必須將其宣告放在 timer1_Tick 的外面。

```
int i=0;
private void timer1_Tick(object sender, EventArgs e)
{ i = (i + 1) % 15; }
```

🔗 程式列印.....

```
namespace e4_3b{
    public partial class Form1 : Form {
        int i=0;
        private void timer1_Tick(object sender, EventArgs e) {
            lblOut.Text=i.ToString ();
            switch (i)
            {
                case 0:
                case 1:
                case 2: //以上3個case 共用以下程式
                    //綠燈
                    lblGreen.Visible = true;
                    lblYellow.Visible = false;
                    lblRed.Visible = false;
                    break;
                case 3:
                case 5:
                    //綠滅
                    lblGreen.Visible = false;
                    lblYellow.Visible = false;
                    lblRed.Visible = false;
                    break;
                case 4:
                case 6:
                    //綠燈
                    lblGreen.Visible = true;
                    lblYellow.Visible = false;
                    lblRed.Visible = false;
                    break;
                case 7:
                case 9:
                    //黃燈
                    lblGreen.Visible = false;
```

```

        lblYellow.Visible = true;
        lblRed.Visible = false;
        break;
    case 8:
    case 10:
        //黃燈
        lblGreen.Visible = false;
        lblYellow.Visible = false;
        lblRed.Visible = false;
        break;
    case 11:
    case 12:
    case 13:
    case 14:
        //紅燈
        lblGreen.Visible = false;
        lblYellow.Visible = false;
        lblRed.Visible = true;
        break;
    }
    i = (i + 1) % 15;
}
}
}

```

自我練習

- 同上自我練習，且顯示每種燈號的剩餘秒數。
- 同上範例，但是可以用一個按鈕，手動控制一個雙向紅綠燈。也就是程式一開始先顯示綠燈，當按一下按鈕變為黃燈，再按一下按鈕變為紅燈，再按一下按鈕變為綠燈，依此類推。提示：規劃時序如下：

時序	綠	黃	紅	綠	黃	紅
0	1	0	0	0	0	1
1	0	1	0	0	0	1
2	0	0	1	0	0	1
3	0	0	1	1	0	0
4	0	0	1	0	1	0
5	0	0	1	0	0	1

3. 同上範例，但改為雙向紅綠燈。

範例 4-2c

猜拳遊戲。請設計一個程式，可以讓使用者與電腦玩猜拳遊戲。（提示：使用者輸入 0，1，2 代表剪刀、石頭、布；電腦產生亂數 0，1，2 亦代表剪刀、石頭、布。）

程式列印

```
int a, b;
string a1 = " ", b1 = " ", r = "";
Console.Write("input 0(剪刀) 1(石頭) 2(布) :");
a = int.Parse(Console.ReadLine());
Random rd = new Random();
b = rd.Next(0, 3); //產生0, 1, 2的亂數
switch (a){
    case 0:
        a1 = "剪刀";
        switch (b) {
            case 0:
                b1 = "剪刀";
                r = "平手";
                break;
            case 1:
                b1 = "石頭";
                r = "電腦贏";
                break;
            case 2:
                b1 = "布";
                r = "你贏";
                break;
        }
        break;
    case 1:
        a1 = "石頭";
        switch (b) {
            case 0:
                b1 = "剪刀";
                r = "你贏";
```

```
        break;
    case 1:
        b1 = "石頭";
        r = "平手";
        break;
    case 2:
        b1 = "布";
        r = "電腦贏";
        break;
    }
    break;
case 2:
    a1 = "布";
    switch (b)
    {
        case 0:
            b1 = "剪刀";
            r = "電腦贏";
            break;
        case 1:
            b1 = "石頭";
            r = "你贏";
            break;
        case 2:
            b1 = "布";
            r = "平手";
            break;
    }
    break;
}
Console.WriteLine($"你出: {a},{a1} ");
Console.WriteLine($"電腦出: {b},{b1}");
Console.WriteLine($"結果: {r}");
```

自我練習

1. 同上範例，但使用三個按鍵，分別代表剪刀、石頭、布。
2. 同上自我練習，但增加顯示雙方贏拳的次數。
3. 同上自我練習，但增加判斷誰先贏 3 次，並輸出結果。
4. 同上範例，但可以三個人同時猜拳並評定勝負。

提示：以上兩個人，兩層 case，三個人可以用三層。

4-3 實例探討

✧ 一元二次方程式

解一元二次方程式的演算法如下：

1. 設有一元二次方程式如右： $ax^2 + bx + c = 0$
2. 若 $a = 0$ 則應輸出 " 輸入錯誤 "。
3. 令 $d = b^2 - 4ac$ 。
4. 若 $d = 0$ ，則方程式有唯一解 $x = \frac{-b}{2a}$ ，若 $d > 0$ ，則方程式有二解
 $x_1 = \frac{-b + \sqrt{d}}{2a}$ ， $x_2 = \frac{-b - \sqrt{d}}{2a}$ ，否則無實數解。請寫程式完成。

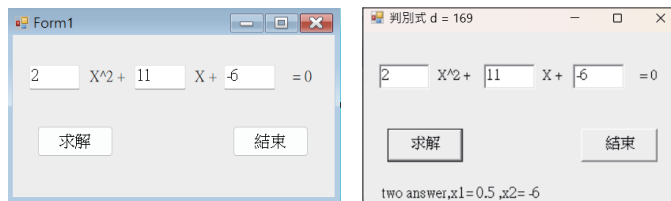
範例 4-3a

請設計一個程式，可以解一元二次方程式 $ax^2 + bx + c = 0$ 。

測試資料：輸入 0,1,2，結果是輸入錯誤；輸入 1,-2,1，結果是唯一解 $x=1$ ；輸入 1,5,-6，結果是 $x_1=1, x_2=-6$ ；輸入 1,2,5，結果是無實數解。

✧ 表單配置與執行結果

1. 使用 txta、txtb、txtc 分別輸入方程式的三個係數 a、b 及 c。
2. 使用 lblout 輸出結果。



✧ 程式列印

```
private void btnClose_Click(object sender, System.EventArgs e){
    Close();
}
private void btnStart_Click(object sender, System.EventArgs e){
```

```

double a = double.Parse(txta.Text);
double b = double.Parse(txtb.Text);
double c = double.Parse(txtc.Text);
double ans1, ans2;
string strb = "";
if (a == 0){
    lblOut.Text = "輸入錯誤";
    return;
}
//判別式
double d = Math.Pow(b, 2)-4*a*c;
this.Text = "判別式d = " + d;
if (d == 0){
    // 判別式=0, 唯一解
    ans1 = (-b/(2*a));
    strb = string.Format("only one answer, x1, x2= {0}", ans1);
}
else if (d > 0){
    // 判別式>0, 二實數根
    ans1 = ((-b+Math.Sqrt(d))/(2*a));
    ans2 = ((-b-Math.Sqrt(d))/(2*a));
    strb = string.Format("two answer, x1= {0} , x2= {1}", ans1,
                        ans2);
}
else
    // 判別式<0, 無實數解
    strb = "no real answer";
lblOut.Text = strb;
}

```

✧ 二元一次方程式

解二元一次方程式的演算法如下：

1. 設二元一次方程式如右： $a_1x + b_1y = c_1$
 $a_2x + b_2y = c_2$
2. 令 $d = \begin{vmatrix} a_1 & b_1 \\ a_2 & b_2 \end{vmatrix}$ (表示 $d = a_1b_2 - a_2b_1$)
3. 假如 $\frac{a_1}{a_2} = \frac{b_1}{b_2} = \frac{c_1}{c_2}$ ，則方程式無限多解，且程式結束。

4. 假如 $d = 0$ ，則程式無解，且程式結束。

5. 唯一解如下：

$$x = \frac{\begin{vmatrix} c_1 & b_1 \\ c_2 & b_2 \end{vmatrix}}{d} = (c_1 b_2 - c_2 b_1) / d$$

$$y = \frac{\begin{vmatrix} a_1 & c_1 \\ a_2 & c_2 \end{vmatrix}}{d} = (a_1 c_2 - a_2 c_1) / d$$

6. 請寫程式完成。

7. 測試資料：輸入 1,2,3,2,4,6，結果是無限多解；輸入 1,2,3,2,4,7，結果是無解；輸入 1,5,-6，結果是 $x_1=1, x_2=-6$ ；輸入 2,1,4,1,-1,-1，結果是 $x=1, y=2$ 。

✕ 三角形面積

若已知三角形三邊長，計算三角形三邊長的演算法如下：

1. 輸入三角形三邊長 a 、 b 、 c 。
2. 將三邊長由小而大重新排列，最小的放入 a ，其次放入 b ，最大的放入 c 。
3. 最小的兩邊之和若小於等於第三邊，則此三邊未能構成三角形，程式提早離開。
4. 假如 $a^2 + b^2 > c^2$ 則為銳角三角形，否則，假如 $a^2 + b^2 = c^2$ 則為直角三角形，否則此三角形為鈍角三角形。
5. 令 $d = \frac{1}{2}(a+b+c)$ 。
6. 三角形面積 $= \sqrt{d(d-a)(d-b)(d-c)}$ 。
7. 請根據以上演算法，寫出程式。

測試資料：3,4,5 直角三角形，面積 6；3,4,6 頓角三角形，面積大約 5.3；3,4,4 銳角三角形，面積大約 5.6；3,4,9 無法構成三角形。

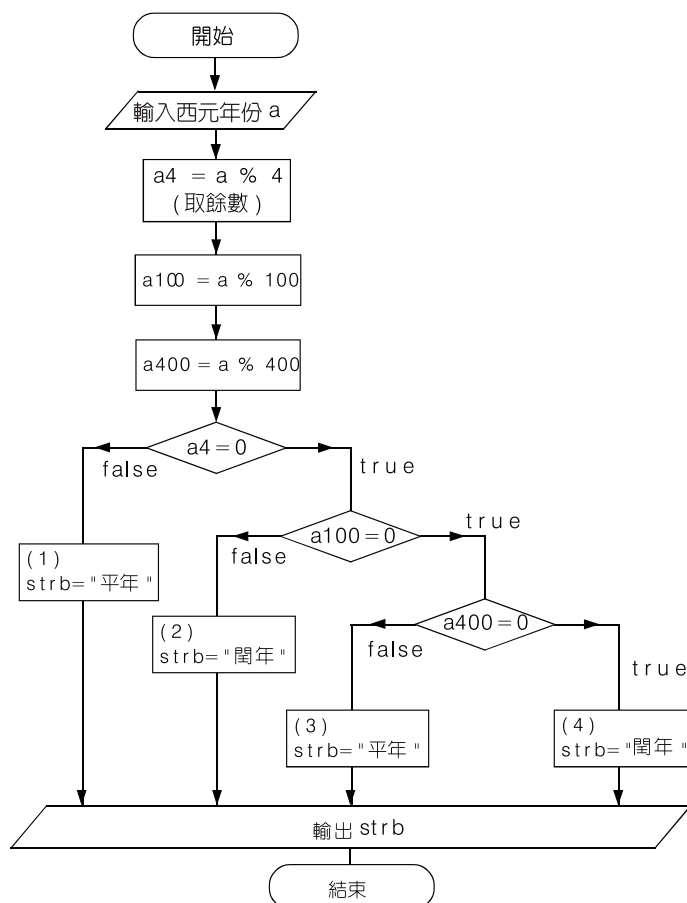
自我練習

1. 隨意撿拾 4 根棒子 (其長度可輸入)，請寫一程式，判斷此 4 根棒子是否可構成四邊形。

提示：圍成三角形的擴充。

閏年

1. 西元的閏年為每 400 年必須有 97 次閏年，其規劃方式如下：
 - (1) 4 的倍數。依此條件共有 100 次。
 - (2) 於 (1) 的條件，扣掉 100 的倍數。依此條件，共有 96 次。
 - (3) 於 (2) 的條件，再加回 400 的倍數。所以共有 97 次。
2. 以上演算分析，流程圖分析如下：



3. 測試資料如下：（流程路線，請對照上面的流程圖）

西元年份	性質	流程路線
3	平年	(1)
4	閏年	(2)
100	平年	(3)
200	平年	(3)
300	平年	(3)
400	閏年	(4)
600	平年	(3)
1200	閏年	(4)
2000	閏年	(4)

4. 請寫程式完成。

※ 範例 4-3b (技藝競賽試題，非資訊本科系可先跳過)

在電腦中，顯示彩色照片或圖片，大都是用 RGB 色彩空間，其中 R 表示紅色 (Red)，其範圍為 0 ～ 255。G 表示綠色 (Green)，其範圍為 0 ～ 255。B 表示藍色 (Blue)，其範圍為 0 ～ 255。上述 0 表示最暗，255 表示最亮。但是要對彩色照片或圖片處理時，RGB 色彩空間無法使用。HSI 色彩座標系統是色彩處理中，常被使用來處理彩色照片或圖片的色彩空間，這是因為 HSI 的轉換有效率，而且，其色彩表示，比較接近人的視覺感知，其中 H 表示色調 (Hue)，其範圍為 0 到 360 度，不同顏色不同度數。S 表示色彩飽和度 (Saturation)，其範圍為 0 到 255，0 表示最不飽和，255 表示最飽和。I 表示強度 (Intensity)，其範圍為 0 到 255，0 表示黑色，255 表示白色。

從 RGB 轉換到 HSI 的公式如下：

$$r = \frac{R}{R+G+B} \cdot g = \frac{G}{R+G+B} \cdot b = \frac{B}{R+G+B}$$

$$h = \cos^{-1} \left\{ \frac{0.5 \cdot [r-g] + (r-b)}{[(r-g)^2 + (r-b)(g-b)]^{1/2}} \right\} \quad h \in [0, \pi] \text{ for } b \leq g$$

$$h = 2\pi - \cos^{-1} \left\{ \frac{0.5 \cdot [(r-g) + (r-b)]}{[(r-g)^2 + (r-b)(g-b)]^{1/2}} \right\} \quad h \in [\pi, 2\pi] \text{ for } b > g$$

$$s = 1 - 3 \cdot \min(r \cdot g \cdot b); \quad s \in [0,1]$$

$$i = (R + G + B) / (3 \cdot 255); \quad i \in [0,1]$$

$$H = h \times 180 / \pi; \quad S = s \times 255 \quad \text{and} \quad I = i \times 255$$

從 HSI 轉換到 RGB 的公式如下：

$$h = H \cdot \pi / 180; \quad s = S / 255 \quad \text{and} \quad i = I \times 255$$

$$x = i \cdot (1 - s)$$

$$y = i \cdot \left[1 + \frac{s \cdot \cos(h)}{\cos(\pi/3 - h)} \right]$$

$$z = 3i - (x + y)$$

其中

$$\text{when } h < 2\pi/3, \quad b = x; \quad r = y \quad \text{and} \quad g = z$$

$$\text{when } 2\pi/3 \leq h < 4\pi/3, \quad h = h - 2\pi/3 \quad \text{and} \quad r = x; \quad g = y \quad \text{and} \quad b = z$$

$$\text{when } 4\pi/3 \leq h < 2\pi, \quad h = h - 4\pi/3 \quad \text{and} \quad g = x; \quad b = y \quad \text{and} \quad r = z$$

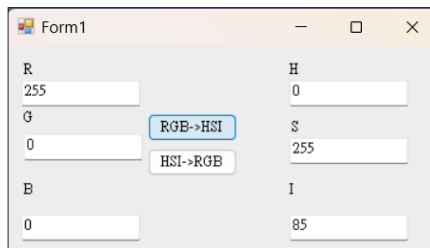
由 RGB 轉換 HSI 的測試資料如下，反之亦然。

	輸入			輸出		
顏色	R	G	B	H	S	I
紅色	255	0	0	0	255	255
綠色	0	255	0	120	255	85
藍色	0	0	255	240	255	85

由 HSI 轉換 RGB 上述輸入和輸出資料相反。

(以上公式，我們重新打字，很難發現是否打錯，若程式不符理想，請參考 <https://codingnote.cc/zh-tw/p/35908/>)

 **執行結果** (本題很容易錯，想走這一行，就要有耐心除錯)



The screenshot shows a Windows application window titled "Form1". It contains a grid of input and output fields for color conversion. On the left, under the "RGB" section, there are three input boxes: "R" with the value "255", "G" with "0", and "B" with "0". On the right, under the "HSI" section, there are three output boxes: "H" with "0", "S" with "255", and "I" with "85". In the center, there are two buttons: "RGB->HSI" (highlighted with a blue border) and "HSI->RGB".

程式列印

```

namespace WindowsFormsApplication1 {
    public partial class Form1 : Form {
        // RGB->HSI
        private void button1_Click(object sender, EventArgs e) {
            double R, G, B; // 本例若宣告為byte, 將會導致錯誤
            double r, g, b, min;
            double h, s, i;
            byte H, S, I;
            R = double.Parse(txtR.Text);
            G = double.Parse(txtG.Text);
            B = double.Parse(txtB.Text);
            r = R / (R + G + B);
            g = G / (R + G + B);
            b = B / (R + G + B);
            if (b <= g)
                h = Math.Acos(0.5 * ((r - g) + (r - b)) / Math.Sqrt((r - g) *
                    (r - g) + (r - b) * (g - b)));
            else
                h = 2 * Math.PI - Math.Acos(0.5 * ((r - g) +
                    (r - b)) / Math.Sqrt((r - g) * (r - g) + (r - b) *
                    (g - b)));
            min = Math.Min(r, g);
            min = Math.Min(min, b);
            s = 1 - 3 * min;
            i = (R + G + B) / (3 * 255);
            H = (byte)((h * 180) / Math.PI);
            S = (byte)(s * 255);
            I = (byte)(i * 255);
            txtH.Text = H.ToString();
            txtS.Text = S.ToString();
            txtI.Text = I.ToString();
        }
        // HSI->RGB
        private void button2_Click(object sender, EventArgs e) {
            double r, g, b;
            double h, s, i;
            double x, y, z;
            double H, S, I;
            const double pi = Math.PI;

```

```
H = double.Parse(txtH.Text);
S = double.Parse(txtS.Text);
I = double.Parse(txtI.Text);
h = H * pi / 180;
s = S / 255;
i = I / 255;
x = i * (1 - s);
if (h < 2 * pi / 3) {
    y = i * (1 + s * Math.Cos(h) / Math.Cos(Math.
        PI / 3 - h));
    z = 3 * i - (x + y);
    b = x; r = y; g = z;
}
else
    if (h < 4 * pi / 3) {
        h = h - 2 * pi / 3;
        y = i * (1 + s * Math.Cos(h) / Math.Cos(Math.
            PI / 3 - h));
        z = 3 * i - (x + y);
        r = x; g = y; b = z;
    }
    else {
        h = h - 4 * pi / 3;
        y = i * (1 + s * Math.Cos(h) / Math.Cos(Math.
            PI / 3 - h));
        z = 3 * i - (x + y);
        g = x; b = y; r = z;
    }
txtR.Text = Math.Round (r*255).ToString();
txtG.Text = Math.Round (g*255).ToString();
txtB.Text = Math.Round (b*255).ToString();
}
}
```

習題



1. 假設所得稅稅率法則如下：
 - (1) 淨所得 30 萬以下 6%。
 - (2) 淨所得 30 ~ 80 萬之間 13%。(前面的 30 萬仍扣 6%，超過 30 萬的部分稅率為 13%，不是全部都扣 13%)
 - (3) 淨所得 80 ~ 200 萬之間扣 21%。
 - (4) 淨所得超過 200 萬的部分扣 30%。試寫一程式可以輸入淨所得，並計算應繳稅額。例如，淨所得若為 100 萬，則其納稅金額計算如下：
 $30 \text{ 萬} \times 6\% + 50 \text{ 萬} \times 13\% + 20 \text{ 萬} \times 21\% = 125000 \text{ 元}$
2. 寫一程式輸入 x 值，並印出其所對應的值，其函數對應如下：
$$y = f(x) = \begin{cases} x+3 & x > 12 \\ x^2 & 5 \leq x \leq 12 \\ \sqrt{x} & 0 < x < 5 \\ 0 & x \leq 0 \end{cases}$$
3. 某一貨品定價 100 元，若購買 500 件（含）以上打 7 折，若購買 499 ~ 300 件則打 8 折，若購買 299 ~ 100 件則打 9 折，購買 100 件以下則不打折，試寫一程式可以輸入購買件數而得總價。
4. 使用者猜數字遊戲。
 - (1) 電腦產生 0 ~ 100 的亂數，然後由使用者猜。
 - (2) 使用者假設下限 $x_1=0$ ，上限 $x_2=100$ ，每次猜其中間值 $x=(x_1+x_2)/2$ 。
 - (3) 電腦回應太大、太小、或猜中。若電腦回應太大，則調整上限 x_2 值為 x ，若回應太小，則調整下限 x_1 為 x ，使用者繼續猜其中間值 $x=(x_1+x_2)/2$ ，直到猜中為止。(提示：本題還沒介紹迴圈，所以先用視窗介面，讓使用者輸入猜測值，本題主要目的是讓使用者瞭解二分猜值法)
5. 同上習題，統計且輸出使用者猜測次數，並依猜測次數給予獎金。假設使用者籌碼初值是 100 元，每次遊戲賭金 10 元，獎勵金依猜中次數如下表。

猜中次數	獎勵金
1	500
2 ~ 3	300
4 ~ 6	10
7 次以後	0

6. 電腦猜數字遊戲。請寫一程式，滿足以下條件。
- (1) 使用者於紙上寫下 0 到 100 的任一個數字。
 - (2) 電腦設定下限 $x_1=0$ ，上限 $x_2=100$ 。
 - (3) 電腦以二分猜值法，猜測使用者數字。 $x=(x_1+x_2)/2$
 - (4) 使用者回應猜中、太大、與太小。
 - (5) 若猜中，則回應臆測次數並結束，若太大則調整上限 x_2 為 x ，太小則調整下限 x_1 為 x 。
- (提示：本題還沒介紹迴圈，所以先用視窗介面，使用三個按鈕，讓使用者回答「太大」「太小」或「猜中」，本題主要目的是讓使用者瞭解二分猜值法)
7. 假設電阻串聯的公式為 $R=R_1+R_2+R_3+R_4$ ，並聯公式為 $R=1/((1/R_1)+(1/R_2)+(1/R_3)+(1/R_4))$ ，請設計一程式，可以同時處理 2、3、4 個電阻的串聯或並聯。(也就是您的程式，僅輸入 2 個、3 個或 4 個都可以計算串聯或並聯的電阻值。)
- * 8. 請設計一程式，可以計算所輸入複數的平方根。例如 $(3-i)*(3-i)=8-6i$ ，所以輸入 $8-6i$ ，可得其平方根為 $3-i$ ， $-(3-i)$ 。(非資訊本科系可先跳過)

方法一：使用標準式

- (1) 本例假設要求 $a+bi$ 的平方根，先設其解為 $c+di$ ，所以以數學式表示如下：

$$\sqrt{a+bi} = c+di \quad a, b, c, d \text{ 屬於 } \mathbb{R}$$

- (2) 兩邊平方。

$$a+bi = (c^2 - d^2) + 2cdi$$

(3) 複數的相等表示實部與虛部對應相等。

$$c^2 - d^2 = a \quad (1)$$

$$2cd = b \quad (2)$$

(4) 解方程式：

由得 $d=b/2c$ 代入①

$$c^2 - \left(\frac{b}{2c}\right)^2 = a$$

$$4c^4 - 4ac^2 - b^2 = 0$$

$$4(c^2)^2 - 4ac^2 - b^2 = 0$$

$$c^2 = \frac{4a \pm \sqrt{(4a)^2 + 16b^2}}{8} = \frac{a \pm \sqrt{a^2 + b^2}}{2} \quad (c^2 > 0)$$

(5) 上式 c^2 只能取正數，負不合。

$$c^2 = \frac{a + \sqrt{a^2 + b^2}}{2}$$

$$c = \pm \sqrt{\frac{a + \sqrt{a^2 + b^2}}{2}}$$

(6) 再將 c 代入 $d=b/2c$ 求得 d ，所以共有兩組解。

方法二：使用極式

(1) 將標準式化為極式：

$$p = a + bi = r \angle \theta$$

(2) 使用隸美弗定理求平方根：

$$\sqrt{p} = \pm \sqrt{r} \angle \frac{\theta}{2} \quad (\text{隸美弗定理})$$

(3) 再將以上兩個極式轉為標準式即為所求，所以其平方根分別是：

$$\sqrt{r} \cos \frac{\theta}{2} + \sqrt{r} \sin \frac{\theta}{2} i$$

$$-\sqrt{r} \cos \frac{\theta}{2} - \sqrt{r} \sin \frac{\theta}{2} i$$

9. 三點共線。請寫一個程式，任意指派 3 點座標，檢查是否共線，若未共線，輸出其面積。例如：(1,1),(2,2),(3,3) 共線；(1,1),(2,2),(5,2) 不共線。
提示：任意 3 點面積若為 0，則共線。
10. 判斷任意點 D 是否在三角形 ABC 內或外。已知 ABC 三點座標，請寫一程式，可以輸入 D 點座標，並判斷任一點 D 是否在三角形 ABC 內或外。例如：三角形三點分別是 (0,0),(3,0),(0,4)，則 (1,1) 在三角形內，(5,0) 則在三角形外。
(提示：D 若在三角形內，則三角形 ABC 面積 = DAB + DBC + DAC 的面積) 例如：三角形三點分別是 (0,0),(3,0),(0,4)，則 (1,1) 在三角形內，(5,0) 則在三角形外。
11. 台北市計程車費率計算如下：
- (1) 計程運價：起程 1.25 公里 70 元，續程每 200 公尺 5 元（日間）。
 - (2) 延滯計時運價：車速每小時 5 公里以下，累計每 1 分鐘 20 秒 5 元（日間）。
 - (3) 夜間加成運價：自夜間 11 時起至翌晨 6 時止（遇跨夜間加成時段之情形，統一以「上車時間」為準），每趟次依日間運價加收 20 元（即起程運價 1.25 公里 90 元，續程及延滯計時運價與日間相同）。
- 請問如何規劃與設計此程式？

測試資料編號	搭乘時間 (時:分)	行駛距離 (公里)	延滯計時累計 時間(分:秒)	費率
1	07:20	1	1:20	$70+5=75$
2	08:30	1.3	5:00	$70+5*1+5*3=90$
3	23:10	1.9	6:00	$70+20+5*4+5*4=130$
4	03:40	2	7:00	$70+20+5*4+5*5=135$

12. 中華郵政普通信件郵資如下表，請寫一程式，可以讓櫃臺辦事員計算郵資。

重量 (公克)	普通	限時	掛號	限時掛號	掛號 附 回執	限時掛號 附 回執
不逾20	8	15	28	35	43	50
21-50	16	23	36	43	51	58
51-100	24	31	44	51	59	66
101-250	40	47	60	67	75	82
251-500	72	79	92	99	107	114
501-1000	112	119	132	139	147	154
1001-2000	160	167	180	187	195	202

備註：每件限重不逾20公斤，逾2公斤，每續重1公斤加收48元。

13. 勞動基準法第 38 條內文如下：

第 38 條：勞工在同一雇主或事業單位，繼續工作滿一定期間者，應依下列規定給予特別休假：

- 一、六個月以上一年未滿者，三日。
- 二、一年以上二年未滿者，七日。
- 三、二年以上三年未滿者，十日。
- 四、三年以上五年未滿者，每年十四日。
- 五、五年以上十年未滿者，每年十五日。
- 六、十年以上者，每一年加給一日，加至三十日為止。

請寫一個程式，可以依照勞工任職年資，計算其特休天數。

14. 任意給定四點座標，請完成以下要求。

- (1) 按順序組合一個四邊形。
- (2) 判斷是否為凸多邊形。
- (3) 若為凸四邊形，請計算其面積。

提示：求其中心點，以中心點為中心，求 4 個點的極座標角度，依照角度排序。

迴圈敘述

前面範例 4-1d，我們已經介紹如何產生一個亂數字元，再由使用者輸入是否正確，現在若要重複 10 次，或直到錯了才停止，此時就需要迴圈指令，才能省事。C # 迴圈敘述如下：

1. for
2. while
3. goto
4. foreach

for 的使用時機為設計階段已知執行次數，for 的用法請看 5-1 節。若未知執行次數，則應使用 while，while 的用法請看 5-2 節。goto 則為非結構化語言的遺留產物，請儘量避免使用，goto 的用法請看 5-3 節。foreach 則對於陣列與集合特別有用，關於陣列的使用請看 6-1 節；關於集合的使用請看 8-4 節。

5-1 for

若於程式設計階段已知執行次數，則可使用 for 敘述，for 敘述的語法如下：

```
for ( [ 計數變數= 起始值 ] ; [ 迴圈運算式 ] ; [ 計數變量 ] )  
{  
    [ 敘述區塊1 ; ]
```

```
[ break ; ]  
[ continue ; ]  
[ 敘述區塊2 ; ]  
}
```

以上語法說明如下：

1. 只要“迴圈運算式”結果為 true，則繼續執行迴圈內的指令區塊。
2. for 運算僅預設執行 1 個敘述，若要執行兩個以上敘述，請加上 {}。請鍵入以下程式，寫出執行結果。

<pre>for (int i = 1; i <= 10; i = i + 1){ Console.WriteLine(i); Console.WriteLine(i); }</pre>	<pre>for (int i = 1; i <= 10; i = i + 1) Console.WriteLine(i); Console.WriteLine(i);</pre>
--	---

以上變數 i 的有效範圍僅在迴圈內，所以以上右邊程式，第 2 個 Console 的 i 無效。若一定要印出 i，那就要在 for 外面宣告 i。

```
int i;  
for (i = 1; i <= 10; i = i + 1){  
    Console.WriteLine(i);  
}  
Console.WriteLine(i);
```

3. 計數變數的改變量若為遞增，請留意終值較大，且迴圈運算式是「<=」；若為遞減，則終值較小，且迴圈運算式是「>=」。請鍵入以下程式，並寫出執行結果。

<pre>for (int i = 1; i <= 10; i = i + 1){ Console.WriteLine(i); } for (int i = 1; i <= 10; i = i + 4){ Console.WriteLine(i); }</pre>	<pre>for (int i = 10; i >= 1; i = i - 1){ Console.WriteLine(i); } for (int i = 10; i >= 1; i = i - 4){ Console.WriteLine(i); }</pre>
--	--

4. 迴圈運算式，若一開始就為 false，則直接離開迴圈。

<pre>for (int i = 1; i >=10; i = i + 1){ Console.WriteLine(i); }</pre>	<pre>for (int i = 10; i <=1; i = i - 1){ Console.WriteLine(i); }</pre>
---	---

5. 遞增或遞減錯誤，則形成無限迴圈。

<pre>for (int i = 1; i <= 10; i = i - 1){ Console.WriteLine(i); }</pre>	<pre>for (int i = 10; i >=1; i = i + 1){ Console.WriteLine(i); }</pre>
--	---

6. 以下程式，左邊每次遞增 2，右邊每次遞減 3。

<pre>for (int i = 1; i <= 10; i = i + 2){ Console.WriteLine(i); }</pre>	<pre>for (int i = 10; i >=1; i = i - 3){ Console.WriteLine(i); }</pre>
--	---

7. 迴圈運算式的關係運算子通常使用『<,<=,>,>=』，若使用『==』，則沒有錯誤訊息，但執行結果不同。請鍵入以下程式，寫出執行結果。

```
for (int i = 1; i ==5; i = i + 1){  
    Console.WriteLine(i);  
}
```

8. 迴圈運算式也不能寫成：

```
int main() {  
    for ( int i=1;i=10;i++)  
        printf("%d",i);  
}
```

因為 C# 關係運算子沒有「=」，這樣是語法錯誤。

9. 計數變量也可在迴圈內再改變。例如：

```
for (int i = 1; i <=10; i = i + 1){  
    Console.WriteLine(i);  
    i = i + 1;  
}
```

程式沒人這樣寫，但是考試卻會出現。

10. 程式若執行到 break，則會提早離開 for 迴圈。

```
int i;
for (i = 1; i <=10; i = i + 1){
    Console.WriteLine(i);//1 2 3 4
    if (i == 4)
        break;
}
Console.WriteLine(i);//4
```

11. 程式若執行到 continue，則會略過 continue 下面的指令區塊 2，繼續執行下一個計數變量。

```
int i;
for (i = 1; i <=5; i = i + 1){
    if (i == 4)
        continue;
    Console.WriteLine(i);//1 2 3 5
}
Console.WriteLine(i);//6
```

12. 指令區塊內可以放置任何合法的指令，當然也可含 for。for 內有 for，稱為巢狀迴圈。

```
for (int i = 1; i <= 5; i = i + 1) {
    for (int j = 1; j <= i; j++) {
        Console.Write(j);
    }
    Console.WriteLine();
}
```

13. 運算式只要合理就好，不一定要用計數變數。例如：

```
int i, s = 0;
for (i = 1; s <= 10; i = i + 1) {
    s=s+i;
}
Console.WriteLine(i);
Console.WriteLine(s);
```

```
int i, s = 0;
for (i = 1; i<=10 && s <= 10; i = i + 1) {
    s=s+i;
}
Console.WriteLine(i);
Console.WriteLine(s);
```

14. 計數變量若用浮點數，因為浮點數會有誤差，請鍵入以下程式，並寫出執行結果，請問有沒有 2.0 呢？

```
for (double i = 0; i<=2; i = i + 0.1) {  
    Console.WriteLine(i);  
}
```

範例 5_1a

認識鍵盤遊戲。請寫一個程式，電腦可以連續出現 10 個字元，每出現一個字元，都可由使用者輸入，並判斷是否正確，統計正確與錯誤題數。

程式列印

前面我們已經完成產生一個字元，使用者鍵入字元，電腦判斷是否正確，現在要連續出現 10 個字元，只要加上 for 迴圈就可以。

```
Random r = new Random();  
int a, b; string result;  
for (int i = 1; i <= 10; i++){  
    Console.Write(i); Console.Write(":");  
    a = r.Next(97, 122);  
    Console.Write((char)a);  
    b = Console.Read();  
    if (b == a)  
        result = "Right";  
    else  
        result = "Wrong";  
    Console.WriteLine(result);  
}
```

1. 請產生 6 個 -3~3 的亂數，統計正數、0、負數的個數。
2. 請產生 10 個 1~6 的亂數，統計偶數與奇數個數。
3. 請產生 10 組 (x,y) 座標亂數，(x,y) 都介於 -2 與 2 的整數，並統計落在原點、x 軸、y 軸、四個象限的個數。

4. 請產生 10 個 1~6 的亂數，統計數字「3」出現的次數。
5. 請產生 10 個 1~6 的亂數，統計所有數字出現的次數。
6. 請產生 10 個 1~6 的亂數，統計哪一個數字出現的次數最多。
7. 心算練習。請連續產生 8 題兩個 1 位數，由使用者輸入相加結果，電腦回應對或錯，並統計正確與錯誤的題數。
8. 費式數列定義如下： $F1=1, F2=1, F_n=F_{n-1}+F_{n-2} \ (n \geq 3)$ ，請寫一程式，可以輸入 n ，得其數列，例如，輸入 $n=6$ ，輸出 1,1,2,3,5,8

✕ 循序法

循序法故名思義就是將所有的值一一列出或一一嘗試，又稱為暴力法，此為電腦程式的與人類思考方式的特殊運算思維，程式語言裡的 for 即可實現此演算法。請看以下範例。

範例 5_1b

電腦的乘法運算。

✕ 運算思維

1. 人類計算乘法大致如下：

$$\begin{array}{r}
 82 \\
 \times 36 \\
 \hline
 492 \\
 246 \\
 \hline
 2952
 \end{array}$$

2. 但是電腦可不用如此麻煩，不用任何記憶，不用任何學習，電腦的強項是計算能力強，很耐操，所以就用 for 迴圈實現一個一個累加如下：
(電腦內部只有累加器與比較器)

```
int a = 82, b = 36, s = 0;
for (int i = 1; i <= b; i++){
    s=s+a;
}
Console.WriteLine(s);
```

自我練習

1. 求 $1+2+3+\cdots+100$ 的和。提示：本題在高中數學稱為等差級數，您要學習公式為（首項 + 末項）* 項數 / 2，但電腦就不用如此麻煩，請直接用一個迴圈，一個一個相加 ($s=s+i$) 就可以，此即為電腦的運算思維。
2. 求 $1+4+7+\cdots+100$ 的和。提示：本題在高中數學稱為等差級數，您要學習公式為（首項 + 末項）* 項數 / 2，但電腦就不用如此麻煩，請直接用一個迴圈，一個一個相加 ($s=s+i$) 就可以，此即為電腦的運算思維。

循序猜值法

所謂循序搜尋法，就是將所有可能的解一一循序代入，又稱為暴力猜值法。例如：您要求任一數的開平方，因為開平方的結果一定在 0 與此數之間，所以我們就從 0 開始，每次遞增 1、0.1、0.01 或 0.001...，至於是要遞增多少，那就依您要的精密度了。例如：要求整數解，那就遞增 1，要求到小數 1 為，那就遞增 0.1，要求到小數兩位，就遞增 0.01 等等等，請看以下範例說明。

範例 5_1c

請用循序猜值法求任意數的平方根。

運算思維

1. 人類求開根號的方法，大致如下：

		3 7 2
1	法則 3	1 3 8 3 8 4 9
2	$3 \times 20 + 7 = 67$ $67 \times 7 = 469$	3 8 4 4 6 9
3	$37 \times 20 + 2 = 742$ $742 \times 2 = 1484$	1 4 8 4 1 4 8 4
4		0

以上演算法則可以減少計算，此演算法解開根號，請看第 8 章習題。但是電腦計算能力強，所以可使用循序猜值法或二分猜值法求解，此即為電腦的運算思維。（二分猜值法請看範例 8_4e）

2. 電腦因為運算能力強，可以不用學習，直接一個一個猜，所以程式如下：

```
int a = 9;
for (int i = 1; i <= a; i++){
    if (i * i >= a)    {
        Console.WriteLine(i);
        break;
    }
}
```

3. 若要實數解呢？也沒問題，以下程式，可以每次遞增 0.1，可以求到小數點 1 位。同理，若要求小數兩位，則每次遞增 0.01。

```
int a = 10;
for (double i = 1; i <= a; i=i+0.1){
    if (i * i >= a)    {
        Console.WriteLine(i);
        break;
    }
}
```

自我練習

1. 請用循序法求解某一正數的立方根。例如：輸入 27 可得到 3.0。本例小數點取 1 位。
2. 假設有一函式 $y=f(x)=x^2-4x-5$ 請分別印出 x 從 -10 到 10 的值。
3. 同上題，請用循序猜值找出其整數解。提示： x 從 -10 到 10 一一代入，找出使函數為零的值，此即為暴力猜值法解題。
4. 同上題，請找出極小值。
5. 函數極值。請寫一程式，可以輸入一個一元二次函式，並求其極大或極小值。例如：輸入 $y=f(x)=x^2-2x+2$ 有極小值 1，輸入 $y=f(x)=-x^2-2x+2$ 有極大值 -1。
6. 假設一個一元多次方程式含有實數解，請寫一程式，可求其解。例如： $y=f(x)=x^2+x-0.75=0$ 的解是 0.5 和 -1.5。提示：浮點運算時，若無法得到 0，此時要使用接近 0 的判斷。例如： $abs(y)<0.0001$ ，即可視為

成立，0.0001 即是其精密度，請換成自己想要的精密度 0.1、0.01 或 0.001。

7. 同上題，求出其實數解在哪兩個整數之間。請問所求範圍有沒有符合上一題所求？提示：若 $f(x) \cdot f(x+1) < 0$ ，則於 x 與 $x+1$ 之間有一實數解。
8. 請用循序法求解任意整數的所有因數。
9. 請寫一個程式，找出 1 至 200 的整數中，找出含有 2 的數字，且統計共含有多少個 2。例如：122 有兩個 2。
10. 請寫一個程式，判斷所輸入數值，是否為質數。
11. 寫一程式，求出西元元年到西元 2050 之間的所有潤年。

巢狀迴圈

迴圈內有迴圈，稱為巢狀迴圈。例如，以下程式可判斷 i 是否質數：

```
bool prime = true;
int i = 11;
for (int j = 2; j <= i - 1; j++){
    if (i % j == 0) {
        prime=false;
        break;
    }
}
if (prime){
    Console.Write(i);
    Console.Write(" is 質數 ");
}
```

現在要找出 2 到 100 的所有質數，繼續使用循序法，就是再加一個迴圈，程式如下：

```
bool prime = true;
for (int i = 2; i <= 100; i++){
    prime = true; //初學者常忘掉
    for (int j = 2; j <= i - 1; j++) {
        if (i % j == 0) {
            prime = false;
            break;
        }
    }
}
```

```
    }  
    }  
    if (prime)    {  
        Console.Write(i);  
        Console.Write(" ");  
    }  
}
```

範例 5-1d

試寫一程式，找出三位數 " 阿姆斯壯數 " 。所謂三位數阿姆斯壯數是指一數等於各個位數的立方和，例如 $153 = 1^3 + 5^3 + 3^3$ 。

程式列印

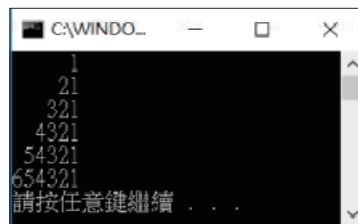
```
int sum1;  
double sum2;  
for (int i = 1; i <= 9; i++){  
    for (int j = 0; j <= 9; j++)    {  
        for (int k = 0; k <= 9; k++)    {  
            sum1 = 100 * i + 10 * j + k;  
            //計算立方和  
            sum2 = Math.Pow(i, 3) + Math.Pow(j, 3) + Math.Pow(k, 3);  
            if (sum1 == sum2)  
                Console.WriteLine(sum1);  
        } //end of for loop k  
    } //end of for loop j  
} //end of for loop i
```

自我練習

1. 請找出 4 位數的阿姆斯壯數。四位數的阿姆斯壯數是各個位數的 4 次方和。例如， $abcd = a^4 + b^4 + c^4 + d^4$ 。

範例 5-1e

請寫一程式，輸出結果如下：（備註：以下題目用來練習找 i 、 j 與 N 的關係，下一章用得上。目的希望整個程式僅與 N 有關，不管 N 如何，都可以執行，讓不同的 N ，都有相同的程式）

**操作步驟**

1. 本例的列數、每一列的空白數、數字個數，列表解析如下：

列編號	空白數	數字
1	5	1
2	4	21
3	3	321
4	2	4321
5	1	54321
6	0	654321

2. 由上表可知，共 6 列，所以 i 從 1 到 6。

3. 每列的空白數 j 與列編號 i 之和都是 6，也就是 $i+j=6$ ，可得 $j=6-i$

4. 每列的數字與列編號均相同，所以 j 從 i 到 1。

程式列印

```
int N=6;
for (int i = 1; i <= N6; i++){
    //印空白
    for (int j = 1; j <= N6 - i; j++)
        Console.Write(" ");
    //印數字
    for (int j = i; j >= 1; j--)
        Console.Write(j);
    Console.WriteLine();
}
```

```
1
21
321
4321
54321
654321
```

自我練習

1. 請寫一程式，輸出結果如下：

***** **** *** ** *	5 4 5 3 4 5 2 3 4 5 1 2 3 4 5
1 2 3 4 5 6 7 8 9 10 11 12 13 14 15	E E E E E D D D D C C C B B A

5-2 while 與 do

上一節的 for 是用於程式設計階段已知迴圈次數，但有些情況，我們於程式設計階段並不知迴圈的執行次數，此時即可使用 while 敘述，且有些迴圈可能一次都不執行，所以 while 敘述又分為前測試迴圈與後測試迴圈。while 的前測試迴圈語法如下圖左，後測試迴圈如下圖右。

while (運算式) { 敘述區塊; }	do { 敘述區塊; } while (運算式);
----------------------------------	---

語法說明

1. 不論是前測試或後測試迴圈，均是運算式值為「真」時，繼續執行迴圈，運算式為「偽」時，離開迴圈。
2. 前測試與後測試迴圈的差別為，前測試迴圈有可能一次均不執行迴圈，但後測試迴圈至少執行一次。
3. 後測試迴圈的 while(運算式) 後面要加分號 (;)，而前測試迴圈的 while 不用加分號。

4. while 敘述區域內亦適用 break 與 continue，前者為強迫提早離開迴圈，後者可略過部份敘述，提早進入條件運算式。
5. 以下敘述，使用前測試迴圈統計 1 至 10 的和。

```
i=1 ; sum=0;
while (i<=10)
{
    sum=sum+i;
    i++;
}
```

6. 以下敘述，使用後測試迴圈統計 1 至 10 的和。

```
i=1 ; sum=0;
do
{
    sum=sum+i;
    i++;
}
while (i<=10);
```

7. 以下敘述，可輸入一連串的整數，待使用者輸入 -1 時，計算所輸入數字的個數、總和及平均。

```
int b, i=0, sum=0, avg=0;
b=int.Parse (Console.ReadLine ());
while (b != -1)
{
    sum=sum+b ;
    i++;
    b=int.Parse (Console.ReadLine ());
}
avg=sum/i;
Console.WriteLine(i);           //個數
Console.WriteLine(sum);         //總和
Console.WriteLine(avg);         //平均
```

8. 若有一級數 $s=1+2+3+\cdots$ ，請問加到第 n 項，其和剛好超過 100。本例於設計階段，就不知要累加幾項，所以使用 while 迴圈如下：

```
int i=0, sum = 0;
while (sum < 100) {
    i++;
    sum +=i;
    Console.WriteLine("{0} , {1}",i,sum);
    //i 對應 0 的位置, sum 對應1 的位置
}
Console.WriteLine(i);
```

✕ 除法

假如沒有除法運算子，該如何完成除法運算呢？可以將被除數連續減去除數，並計算可減去幾個除數，則可減去的除數個數即為商，剩下的被減數即為餘數，其演算法如下：

1. a = 被除數。
2. b = 除數。
3. 商數 $q=0$ 。
4. 所謂的商就是被除數 a 共可減去幾個除數 b ，也就是只要 a 大於等於 b ，就要執行以下敘述：

```
a=a-b ;
q=q+1 ;
```

以 $8/3$ 為例，計算過程如下：

```
8>5 成立 8-3=5 q=1
5>3 成立 5-3=2 q=2
2>3 不成立 離開迴圈 q=2, 餘數=2
```

範例 5-2a

示範以上除法運算。

 程式列印

```
int a = 8, b = 3, q = 0;
while (a >= b){
    a = a - b;
    q++;
}
Console.WriteLine(a); //商
Console.WriteLine(q); //餘數
```

 補充說明

1. 人類通常用直式除法，直式除法的演算，請看第八章習題。以上也稱為循序法，此即為電腦的運算思維。電腦計算能力強，所以可以不用學習任何方法，直接使用循序法解題。
2. 學習程式設計之初，每遇到問題，不要先想有什麼好方法，而是把自己變笨，都先用循序法解，若要提高效率，再好好研究演算法這門課。
3. 以上是整數解，那實數解呢，例如 $8/3$ ，小數 1 位是 2.6，兩位是 2.66，請寫程式完成。

 進位數的轉換

一個十進位數 a 要轉為 n 進位數，其方法是將 a 連續除以 n ，直到整數商為 0，其餘數的字串累加即為 n 進位數的值。以上說明其演算法如下：

1. a 為待轉換的十進位數。
2. n 為某進位數。
3. r 為餘數。
4. $strn$ 為 n 進位數的值。
5. 將 a 連續除以 n ，直到整數商為 0，其餘數的字串累加，即為 n 進位數的值。

以上演算法的原理解說如下：

1. 若要將十進位的 a 轉為二進位，則以數學式表示如下：

$$\begin{aligned}(a)_{10} &= a_0 + a_1 2^1 + a_2 2^2 + a_3 2^3 + a_4 2^4 + \dots \dots \\ &= a_0 + 2(a_1 + a_2 2^1 + a_3 2^2 + a_4 2^3 + \dots \dots) \quad // \quad a_1 \text{ 以後，提出 } 2\end{aligned}$$

2. 上式的 a_0 為 a 除以 2 後所得的餘數， $a_1 + a_2 2^1 + a_3 2^2$ 則為 a 除以 2 而得的整數商，重複上式，直至整數商等於零為止。
3. 最先產生的餘數應放在 2 進位的最右邊。

$$(a)_{10} = (a_n \dots a_4 a_3 a_2 a_1 a_0)_2$$

以 13 為例，其過程如下：

```
13/2=6..1
6/2=3..0
3/2=1..1
1/2=0..1
```

結果是 1101，最先出爐的餘數，放在最右邊。

以上演算法，因為設計階段不知執行次數，正適用前測試迴圈的 while，請看以下範例說明。

範例 5-2b

請將十進位數轉為 n 進位數（本例暫討論 n 範圍為 $n \leq 9$ ，其餘 $n \geq 11$ 的情形，則待陣列介紹後再討論）。

程式列印

```
int a=13, n=2, r; string strn = "";
while (a > 0){
    r = a % n;
    strn = r.ToString() + strn; //先出爐的要放右邊
    a /= n; //整數商
}
Console.WriteLine("其結果為： {0}", strn);
```

✕ 輾轉相除法

輾轉相除法可以求兩數的最大公因數，其處理方式是將大數除以小數，若餘數為 0，則小數即為最大公因數；若不為 0，改以原小數為被除數、原餘數為除數，繼續相除，直到餘數為 0，而促使餘數為 0 的除數，即為此兩數的最大公因數。演算法說明如下：

1. 輸入 a、b 兩數。
2. 假如 $b > a$ ，則兩者交換。
3. 宣告 r 為餘數。
4. 將 a 除以 b、得餘數 r，假如餘數不為 0，則以原除數為被除數、原餘數為除數，重複執行 a 除以 b，直到餘數為 0，促使餘數為 0 的除數，即為最大公因數。
5. 本例以 $a=21, b=9$ ，實際演練如下：
 - (1) $a=21; b=9;$

```
r=a%b=3;  
a=b=9;  
b=r=3
```

$r > 0$ 成立，所以繼續執行迴圈。

- (2) $a=9; b=3;$

```
r=a%b=0;  
a=b=3;  
b=r=0;
```

$r > 0$ 不成立，所以離開迴圈，最大公因數為 $a=3$ 。

以上演算法，設計階段不知執行次數，正適用後測試迴圈 do，請看以下範例說明。

範例 5-2c

請寫一程式完成輾轉相除法，求兩數的最大公因數。

程式列印

```
int a=9, b=21, r,t;  
if (b > a){  
    t = a; a = b; b = t;  
}  
do{  
    r = a % b; //取餘數  
    a = b;      //以原除數為被除數  
    b = r;      //以原餘數為除數  
} while (r > 0);  
Console.WriteLine("其最大公因數為： {0}", a);
```

（備註：以上是輾轉相除法，兩數最大公因數，當然可用循序法，請自行練習）

自我練習

1. 請寫一個程式，滿足以下條件：
 - (1) 可以產生 1 至 6 的亂數。
 - (2) 累加以上亂數。
 - (3) 輸出此亂數與其和。
 - (4) 若亂數不為 1，則重複 (1) ~ (3)，直到亂數為 1，則程式結束。
2. 請寫一個程式，滿足以下條件：
 - (1) 可以產生兩個 2 至 9 的亂數。
 - (2) 累加以上亂數。
 - (3) 輸出此亂數與其和。
 - (4) 若亂數和大於 10，則重複 (1) ~ (4)，直到亂數和小於等於 10，則程式結束。
3. 請寫一程式，可以連續產生兩個 2 ~ 7 的亂數，並輸出此兩個亂數，直到後面的亂數大於前面的亂數。

4. 請寫一程式，可以連續產生 3 個 1 ~ 6 的亂數，並輸出此 3 個亂數，直到有其中兩個亂數相等為止並輸出另一不相同的亂數。
5. 請寫一程式，可以連續產生 4 個 1 ~ 6 的亂數，並輸出此 4 個亂數，直到有其中兩個亂數相等為止，並輸出此不相等的數字與和。例如，產生 6,4,5,1 則繼續產生亂數，若亂數為 6,2,1,6 則其和為 3。

✎ 二分猜值法（高中數學演算法）

求方程式的解時，例如，任意數的平方根，若其解必在兩數之間（本例假設兩數為 x_1 與 x_2 ），則可使用二分猜值法求解。其方法是先假設兩數的一半（本例假設為 x ）為方程式的解，將 x 代入原方程式，若使得原方程式值太大，表示其解假設得太大，而正確解必在 x_1 與 x 之間；反之表示假設得太小，正確解必在 x 與 x_2 之間。重複以上二分猜值法步驟，則每次均可將正確解的範圍縮小一半，直到 x_1 與 x_2 之間的距離非常小時（例如 0.001 或 0.00000001，此為使用者要求的精密度），則此時的 x_1 或 x_2 都可為該方程式的解。那要猜到何時呢？答案是設定一個精密度，例如，您要小數一位，那就是下界與上界之間的距離小於 0.1，若是要小數兩位，那就是下界與上界之間的距離小於 0.01，此時下界與上界的中間值，就是所要求的答案了。

二分猜值法求平方根的演算法如下：

1. 設求解的正數為 x_2 ，則其平方根必在 $x_1=0$ 與 x_2 之間。
2. 首先猜 x_1+x_2 之和的一半 x 。
3. 若所猜 x 的平方小於原數，表示猜得太小，並縮小猜值範圍為 $[x, x_2]$ 。
4. 若所猜的 x 的平方大於原數，表示猜太大，並縮小猜值範圍為 $[x_1, x]$ 。
5. 重覆步驟 2、3、4，直到 $[x_1, x_2]$ 的範圍小於使用者指定的精密度為止。此時的 x_1 或 x_2 即為平方根。

以上演算法以 $x_2=9$ 分析如下：

1. 根必在 0 與 9 之間。
2. 首先猜 4.5，因 4.5 的平方大於 9，表示猜得太大，所以縮小範圍為

$[0, 4.5]$ 。

3. 其次猜 2.25，因 2.25 的平方小於 9，表示猜得太小，所以縮小範圍為 $[2.25, 4.5]$ 。
4. 重複以上動作，直到猜值的範圍很小時，此範圍內的任意數即為解。

範例 5-2d

請以上二分猜值法求任一正數的平方根。

執行結果

```
1:x1,x,x2: 0 , 4.5 ,9
2:x1,x,x2: 0 , 2.25 ,4.5
3:x1,x,x2: 2.25 , 3.375 ,4.5
4:x1,x,x2: 2.25 , 2.8125 ,3.375
5:x1,x,x2: 2.8125 , 3.09375 ,3.375
6:x1,x,x2: 2.8125 , 2.953125 ,3.09375
7:x1,x,x2: 2.953125 , 3.0234375 ,3.09375
其平方根為: 2.953125
```

程式列印

```
double x, x1, x2, a,i=1;
a = 9;
x1 = 0;
x2 = a;
//只要x1與x2之間的距離大於0.1則重覆迴圈
while (Math.Abs(x1 - x2) > 0.1){
    x = (x1 + x2) / 2;
    Console.WriteLine("{0}:x1,x,x2:{1} , {2} ,{3}",i, x1,x,x2);
    if ((x * x - a) <= 0)
        x1 = x; //猜的太小,調整下界為x
    else
        x2 = x; //猜的太大,調整上界為x
    i++;
}
Console.WriteLine("其平方根為: {0}", x1);
```

自我練習

1. 請以二分逼近法，求解兩數相除的結果。

2. 請以二分逼近法，求解一正數的立方根。
3. 請寫一程式，由使用者默想一個 0 到 50 的數字，電腦使用二分逼近法猜此數字，使用者應逐次回答太大、太小、或猜中，且回應幾次猜中。
4. 使用者猜數字遊戲。
 - (1) 電腦產生 0 ~ 100 的亂數，然後由使用者猜。
 - (2) 使用者假設下限 $x_1=0$ ，上限 $x_2=100$ ，每次猜其中間值 $x=(x_1+x_2)/2$ 。
 - (3) 電腦回應太大、太小、或猜中。若電腦回應太大，則調整上限 x_2 值為 x ，若回應太小，則調整下限 x_1 為 x ，使用者繼續猜其中間值 $x=(x_1+x_2)/2$ ，直到猜中為止。

5-3 goto

前面二節已介紹了結構化程式設計所需使用的迴圈敘述 `for`、`while` 與 `do`，原則上使用以上敘述再配合 `break` 及 `continue`，即可解決程式設計所遇到的問題。`goto` 是一種非結構化程式設計所遺留的產物，原則上是一個不該鼓勵使用的敘述，且目前的開發工具 `Java`、`Python` 等，都已將 `goto` 去除，所以強烈建議讀者不要刻意學習 `goto` 的用法，但還是擔心有人不習慣沒有 `goto` 的窘境，因此本節介紹 `goto` 用法。使用 `goto` 之前必須宣告跳躍點的標記，其宣告語法如下，其中標記命名方式同識別字的命名。

標記名稱：

其次，標記的使用語法如下，跳躍點的標記名稱可在 `goto` 的上面或下面，但不可跳離自己的程序、亦不可跳進別的迴圈。

goto 標記名稱：

例如，以下是 `goto` 範例，且可以用下一章的結構化迴圈代替。

```
int i = 1;
aaa: i = i + 1;
if (i <= 10)
    goto aaa;
Console.WriteLine ( i );
```

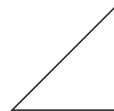
5-4 實例研究

✧ 積分的剖析

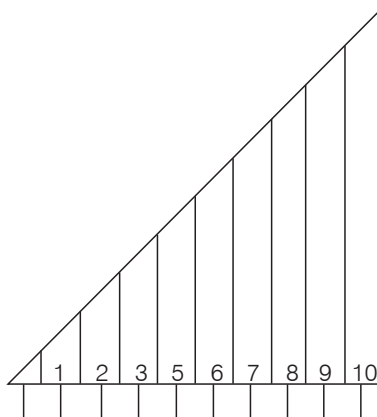
若有積分式如下：

$$\int_0^{10} x dx$$

此積分式就如同求右圖邊長為 10 的等腰三角形，則其面積依照公式是 50。此公式的由來，解析如下：



1. 若 $dx=1$ ，也就是將三角形切為 10 個矩形，如下圖，每個矩形的底是 1，則其面積如下：



```
float sum = 0;
float x, y, dx = 1f;
for (x = 1; x <= 10f; x = x + dx){
    y = x; //高分別是,1,2,3,4,5,6,7,8,9,10
    sum = sum + y * dx;
}
label1.Text = sum.ToString();
```

結果是 55。

2. 若 $dx=0.1$ ，也就是將三角形切為 100 個矩形，則其面積如下：

```
float sum = 0;
float x, y, dx = 0.011f;
for (x = 0; x <= 10f; x = x + dx)
{
    y = x; //高分別是,0.1,0.2,0.3...9.9,10
    sum = sum + y * dx;
}
label1.Text = sum.ToString();
```

結果是 49.5495 。

3. 若 $dx=0.01$, 也就是將三角形切為 1000 個矩形, 則其面積如下

```
float sum = 0;
float x, y, dx = 0.01f;
for (x = 0; x <= 10f; x = x + dx)
{
    y = x;
    sum = sum + y * dx;
}
label1.Text = sum.ToString();
```

結果是 49.99398 。

4. 若 dx 取的非常小, 也就是切為無限多個矩形, 則其結果將會是 50, 這就是積分的道理。

範例 5-4a

示範以上積分的剖析。

自我練習

1. 使用以上積分法求解 $\sin(x)$ 正半週面積 $= \int_0^{\pi} \sin x dx = 2$ 。

✎ Sinx 的剖析

當我們欲求任意角度的 Sinx 值時, 電腦是以其泰勒展開式求其值, 如以下敘述:

$$\sin x = x + \frac{x^3}{3!} + \frac{3^5}{5!} + \frac{3^7}{7!} \dots (\text{x 要取徑度量})$$

例如， $x=30^{\circ}$ 。其徑度量是 $x=0.52359$ ，若取 1 項

$$\sin x \approx x = 0.52359$$

若取 2 項

$$\sin x \approx x + \frac{x^3}{3!} = 0.49967$$

若取 3 項

$$\sin x \approx x + \frac{x^3}{3!} + \frac{x^5}{5!} = 0.500002$$

若取 4 項

$$\sin x \approx x + \frac{x^3}{3!} + \frac{x^5}{5!} + \frac{x^7}{7!} = 0.4999999$$

範例 5-4b

示範以上演算法。

程式列印

```
int i, n, a, j;
double x, y = 0, y1;
x = 30*Math.PI / 180; //30度
n = 3; //項數
for (i = 1; i <= n; i++){
    a = 1;
    for (j = 1; j <= 2 * i - 1; j++){
        a = a * j;
    }
    y1 = Math.Pow(-1, i - 1) * Math.Pow(x, 2 * i - 1) / a;
    y = y + y1;
}
Console.WriteLine(y);
```

取的越多，越後面項數的數字就越小，所累加的精密度就越大，那到底要取幾項，那就是依使用者精密度而定，例如，您要取到小數第二位，那您的取到的項數前面兩位數字就要不為零，也就是您取到的項數的數值要小於 0.01；若是精密度要取到小數點第三位，那就是要取的項數的數值要小於 0.001。

範例 5-4c

寫一程式，可依所需精密度求解 sin 函數。

程式列印

```
int i = 1, a, j;
double x, y = 0, y1, n;
x = 30*Math.PI / 180; //角度
n = 0.01; //精密度
do{
    a = 1;
    for (j = 1; j <= 2 * i - 1; j++)
        a = a * j;
    y1 = Math.Pow(-1, i - 1) * Math.Pow(x, 2 * i - 1) / a;
    y = y + y1;
    i++;
} while (Math.Abs(y1) > n); //只要條件成立，繼續迴圈
Console.WriteLine(y);
Console.WriteLine(i); //次數
```

自我練習

$$1. \cos x = \sum_{n=0}^{\infty} \frac{(-1)^n}{(2n)!} x^{2n} = 1 - \frac{x^2}{2!} + \frac{x^4}{4!} - \frac{x^6}{6!} + \dots$$

請寫一程式，可以輸入角度，並求得 $\cos x$ 之值。

$$2. \text{請寫程式，求證 } \lim_{n \rightarrow \infty} \frac{n!}{n^n} = 0$$

$$3. \text{請寫程式，求證 } \sum_{n=1}^{\infty} \frac{n^2 - n - 1}{n!} = 1$$

範例 5-4d

質因數連乘積。(工科技藝競賽 110)

程式列印

前面找出因數，每個數字僅要除以 1 次，但質因數則不一定，有可能有很多個 2，很多個 3，所以要使用不定迴圈 while，如以下程式。

```
int a = 360;
bool first=true;
for (int i = 2; i <= a; i++){
    int n = 0;
    while (a % i == 0)    {
        n = n + 1;
        a = a / i;
    }
    if (n > 0)    {
        if (first)//第1次不用輸出*號
            first = false;
        else
            Console.Write('*');
        if (n == 1)//僅一個
            Console.Write("{0}", i);
        else
            Console.Write("{0}^{1}", i, n);
    }
}
```

習題



1. 請用雙迴圈印出如下的執行結果：

(1)

```
1
1 2
1 2 3
1 2 3 4
1 2 3 4 5
```

(2)

```
1 2 3 4 5
  2 3 4 5
    3 4 5
      4 5
        5
```

(3)

```
      5
    4 4
  3 3 3
2 2 2 2
1 1 1 1 1
```

(4)

```
      E
    D D D
  C C C C C
B B B B B B B
A A A A A A A A A
```

(5)

```
      *
    * * *
  * * * * *
* * * * * * *
  * * * * *
    * * *
      *
```

(6)

```
      *
    * * *
  * *   * *
* *     * *
  * *   * *
    * * *
      *
```

2. 求 $7+77+777+\dots$ 至 n 項之總和。(n 可任意輸入)
3. 請寫一個程式，輸入一正整數，將其質因數分解後印出其式子，例如：

輸入：3 19

輸出：319 = 11*19

輸入：19

輸出：19= 質數

輸入：521752

輸出：5 2 1 7 52=2^3* 7 ^2 * 1 1 ^3

(技藝競賽試題)

4. 請寫一個程式，可以處理分數的加減乘除運算，運算之後的結果應約化為最簡分數或整數。例如：
- $$\frac{2}{3} + \frac{1}{3} = 1$$
- $$\frac{2}{3} \times \frac{7}{2} = \frac{7}{3}$$
5. 請寫一程式，可以協助使用者將任意小數轉為分數，例如： $0.\bar{3} = \frac{3}{9} = \frac{1}{3}$ ， $0.\overline{165} = \frac{165-1}{990} = \frac{164}{990} = \frac{82}{495}$ （可以約分時要約分）。
6. 請寫一程式，可以執行使用者與電腦猜拳遊戲，使用者使用三個按鍵分別代表剪刀、石頭、布，電腦則使用亂數產生 0,1,2, 並輸出每次猜拳結果，直到有一方先贏 3 次為止。
7. 請用迴圈求解 $4 * \int_0^1 \sqrt{1-x^2} dx$ ，此即為單位圓面積 3.14159。
8. 有一半徑 10 公尺的圓，計算圓面積的方法如下，請寫程式完成。
- (1) 使用寬度 1m 的正方形磁磚填滿，計算可放入幾塊，然後乘以每塊的面積。（可用 if 判斷）
 - (2) 使用 10cm 的正方形磁磚填滿，計算可放入幾塊，然後乘以每塊的面積。
 - (3) 使用 1cm 的正方形磁磚填滿，計算可放入幾塊，然後乘以每塊的面積。