Chapter 2

Data Type and Console I/O

2.1 Essentials of Program

2.1.1 Identifier

- Like every person has a name for identification
- Variable, array, structure, function, class, and etc. in a program
  ⇨ Easy to identify
  ⇨ Name is required
- Identifier naming rule has to be followed

Identifier Naming Rule

Continue…

1. Initial letter has to be Unicode upper cases & lower cases, _, or Chinese character, the following letters are alphabets, number, _, or Chinese character. Space is not allowed
2. Maximum length is 16,383 letters, not too long, lest it is hard to remember and easy to cause wrong key-in
3. Keywords cannot be identifiers, however they can be identifier if there is @ before them. Ex: if is a keyword, @if is an identifier
4. Upper cases and lower cases are different. Ex: tube and TuBe is different
5. _pagecount, Part9 and Number_Items are legal identifiers
6. Illegal identifier: 101Metro(initial letter is a number)，M&W(& character)

2.1.2 Statement

- The smallest executable unit in high-level language
- Collection of statements forms a program
- Statement is formed by
  ⇨ keyword, operator, variable, constant, expression, and so on
- Write program line by line from top to bottom
- Statements in VC# is ended by ;. It is allowed to combine many statements in one line.
  Not suggested for it is uneasy to read.

- **C# has free writing style like C language**
- **When the statement is more than 1 line, it is able to press <Enter> after words, operators, slave symbol(.) and put it in multiple line**
- **Compiler takes these lines as a statement**

```
MessageBox.Show("Visual C# 2012 is an" + "Object Oriented Language.");
```

Multi line writing:

```
MessageBox.
    ("Visual C# 2012 is an" +
    "Object Oriented Language."
    );
```

2.1.3 Keyword

- Also called reserved word
- Reserved identifiers have special meanings to the compiler
- Keywords cannot be used as the name of variables
- Pre-fix character @ is required if the keyword is taken as an identifier
- The identifiers' color is blue when programming

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

2.2 Constant and Variable

- **Data are divide into 2 kinds if they fit four fundamental operations of arithmetic:**
  - **① numeric data**
  - **② string data**
- **Data are divide into 2 kinds if they can be changed:**
  - **① Constant**
  - **② Variable**

2.2.1 Constant

- Cannot be changed when the program is running
- Divided into:
  ① Literal Constant
  ② Symbolic Constant
- Literal Constant
  saved as specific number or text
- Symbolic constant can be a part of statement, also can be assigned to a symbolic constant or a variable

C# allows these types of Literal constant:

Literal constant

- Boolean ：true、false
- Integer ：25, -30
- Float ：24.5, 7.1E+10
- String ："Hello"，"24.5"，"VC# 2012 從零開始"
- Character："a"，"8"
- Date ：#103/5/20#， 10:36PM#

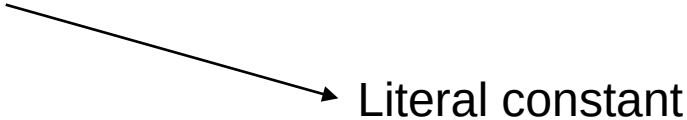
2. Symbolic Constant

- **Use meaningful variable name to replace the unchanged numbers or strings**
- **To store the unchanged value when the application is running**
- **Cannot reassign the value during runtime**
- **Symbolic constant uses const keyword and expression to declare and assign initial value**

Ex:


```
const string a = “Hello”;
Console.Write(a + “ world!”);
```

Result: Hello world!



Symbolic Constant Declaration:

[access modifier] const [data type] [name] = [value/string/expression]

Available:
public
private(default)
protected

https://msdn.microsoft.com/zh-tw/library/ba0a1yw2.aspx

Available:
int
long
short
decimal
float
double
char
string
DateTime
bool
object

Follow identifier naming rule

Cannot be leave out

const string a = “Hello”;

- **Ex:**

```
const int DaysInYear=365;
const int WeeksInYear=52;
const int may=5;
const double PI=3.1416;
const bool pass=true;
const string name="Viusal C# " ;
```

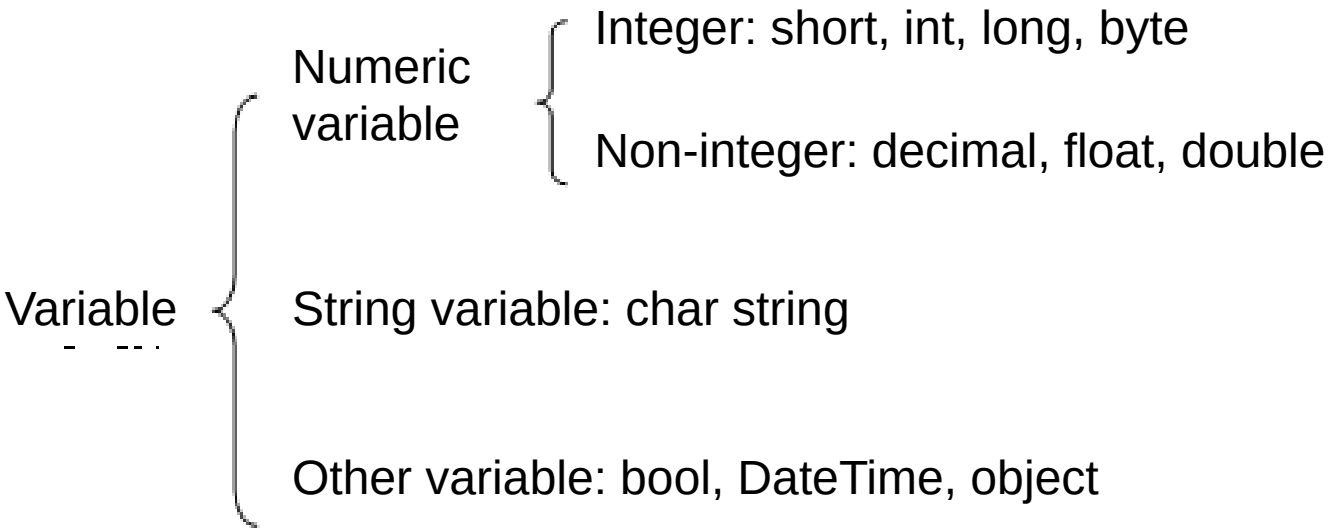
- **Many symbolic constant declarations in one statement**

```
const double x = 1.0, y = 2.0, z = 3.0;
const int four = 4, five=5;
```

2.2.2 Variable

- A variable represents a specific data item or value
- A memory space is reserved to store data Program progress:
  constant is unchanged, variable can be reassigned as different values
- Declare the variable before use it. When declaring variable:
  - give variable name
  - follow naming rule
  - give data type for deploy memory space

Data Types of Variable



X = Y + 10 ;

① **10 is a constant**

memory stores 10, unchangeable

② **X and Y are variable**

X and Y are changeable

2.3 Declare Data Type of Variable

- **Variables used in the program have to be declared in advance**
- **Declared variables' data type are known.**
- **Suitable memory spaces is reserved for the data when the program is under compiling。**
- **Grammar:**

  [data type] [variable name];

Allowed data type of variables in VC#

Data type	Scope	Memory size
bool (Boolean value)	true or false	1 Byte
sbyte (8-bit signed int)	-128 ~ 127	1 Byte
byte (8-bit unsigned int)	0 ~ 255	1 Byte
short (short int)	-32,768 ~ 32,767	2 Bytes
ushort (unsinged short)	0 ~ 65,535	2 Bytes
int (integer)	-2,147,483,648 ~ 2,147,483,647	4 Bytes
uint (unsigned int)	0~4,294,967,295	4 Bytes
long (long int)	-9,223,372,036,854,775,808 ~ 9,223,372,036,854,775,807	8 Bytes
ulong (unsigned long)	0 ~ 18,446,744,073,709,551,615	8 Bytes

Data Type	Scope	Memory Size
float (floating decimal)	-3.402823e38 ~ 3.402823e38	4 Bytes
double	-1.79769313486232e308~ +1.79769313486232e308	8 Bytes
decimal	-79228162514264337593543950335~ +79228162514264337593543950335	16 Bytes
char (character)	Unicode character (‘)	2 Bytes
string	Unicode character string (“)	Variable length
object	Object data type, can store any type of data	

Variable Declaration

Continue…

- **Many variables can be declared in one statement**

  int a , b ;

- **You can allow to assign initial values to variables when declaration**

  Grammar：

  [data type] [variable 1]=[value 1], [variable 2]=[value 2], … ;

Ex1: declare a is an integer variable and initial value is 10, usage: int a=10;

Ex2: declare a is an integer variable and initial value is 10; declare b is a Boolean variable and initial value is false

Usage: int a =10 ; bool b =false;

Ex3: declare a, b, c is a double variable, initial values are 1.1, 2.2, 3.3

Usage: double a=1.1, b=2.2, c=3.3;

2.4 Operator and Expression

2.4.1 Operator and operand

- **The symbol of operator**
  Ex: ＋, －, ×, ÷
- **Use operators to combine operands, variables, constants and functions into an expression**
- **Expression is made up of operators and operands**
- **Ex: price*0.05**

① **Unary Operator**

only 1 operand is required

prefix notation, ex: -5

② **Binary Operator**

2 operands are required

infix notation), ex: 5+8

③ **Tenary Operator**

3 operands are required, like ? ... : …

Grammar: [variable] = [condition] ? [value 1] : [value 2];

```
string str = (score >= 60 ? ”PASS” : ”DOWN”);
```

Unary Operator

Operator	Usage	Example
+	Positive number	+100
-	Minus number	-money
!	Not	!x (if x is true, then !x is false)

Ex: check the score variable is passed or not?

if passed, assign “PASS” to string variable str,

if failed, assign “DOWN”

```
int score = 59;
string str = (score >= 60 ? ”PASS” : ”DOWN”);
```

Seven types of operators

1. Arithmetic Operator
2. Assignment Operator
3. Relational Operator
4. Logical Operator
5. Additive Operator
6. Increment/Decrement Operator
7. Shift Operator

2.4.2 Arithmetic Operator

Priority	Operator	Usage	Example
1	*	Multiply	area = height * width;
	/	Divide	salary = income / 12;
	%	Remainder	x = y % 6;
2	+	Plus	x = x + 50;
	-	Minus	y = x - z;

2.4.3 Relational Operator

- Also called comparison operator
- Used when two numbers or strings have to be compared
- Two operands are required, data types of these operands have to be identical. The result is true or false
- Change the program's process by structural statement and the result of relational operator

Relational operator

Relational operator	Usage	Example
= =	Equal	‘A’ == ‘a’ is false
>	Greater	7 > 4 is true
<	Smaller	‘b’ < ‘a’ is false
> =	Greater or equal	-124 >= -123 is false
< =	Smaller or equal	12.3 <= 12.4 is true
! =	Not equal	“ABC” != “abc” is true

2.4.4 Logical Operator

- One relational statement is one condition
- Use logical operator to link many relational statements
- VC# 2013 provides logical operators:

1. AND

A	B	A && B
true	true	true
true	false	false
false	true	false
false	false	false

- Ex: 70 < score ≤ 79 statement: (score) > 70 && (score<=79);

2. OR

A	B	A || B
true	true	true
true	false	true
false	true	true
false	false	false

Ex: score < 0 or score ≥ 100 written in conditional way:

(score<0) || (score>=100)

3. XOR (Exclusive OR)

A	B	A ^ B
true	true	false
true	false	true
false	true	true
false	false	false

4. NOT

A	!A
true	false
false	true

2.4.5 Assignment Operator

Operand	Statement	Assume x=5, y=2. after calculation
=	x=y	x=5
+=	x += y (x = x + y)	x=5+2=7
-=	x -= y (x = x - y)	x=5-2=3
*=	x *= y (x = x * y)	x=5x2=10
/=	x /= y (x = x / y)	x=5/2=2.5
%=	x %= y (x = x % y)	x=x%2=1
^=	XOR logic operation x^=y	x=x^y → 7 Process: $0101_2=5_{10}$ Xor $0010_2=2_{10}$ $0111_2 \rightarrow 7_{10}$

Operand	Statement	Assume x=5, y=2. after calculation
&=	AND logic operation x^=y	x=x&y → 0 Process: $0101_2=5_{10}$ and $0010_2=2_{10}$ $0000_2 \rightarrow 0_{10}$
|=	OR logic operation x^=y	x=x|y → 7 Process: $0101_2=5_{10}$ Xor $0010_2=2_{10}$ $0111_2 \rightarrow 7_{10}$
<<=	x <<= 1 (x = x << 1)	x = 5 << 2 (5 * 4 = 20)
>>=	x >>= 1 (x = x >> 1)	x = 5 >> 2 (5 / 4 = 1)

2.4.6 Additive Operator

- + can be an additive operator, or be used to combine strings
- If operands at two side of +
  ① numeric data, additive operator, the result is number
  ② string data, combination operator, the result is string
- string myStr ;
  int a ;
  myStr= "To be " + "Or Not to be";
  a = 20 + 30 ;

Increment/Decrement Operator

Operator	Expression	Description
++	x=a++ (post-increment)	Process x=a, then process a+1
	x=++a (pre-increment)	Process a+1, then process x=a
--	x=a-- (post-decrement)	Process x=a, then process a-1
	x=--a (pre-decrement)	Process a-1, then process x=a

2.4.7 Shift Operator

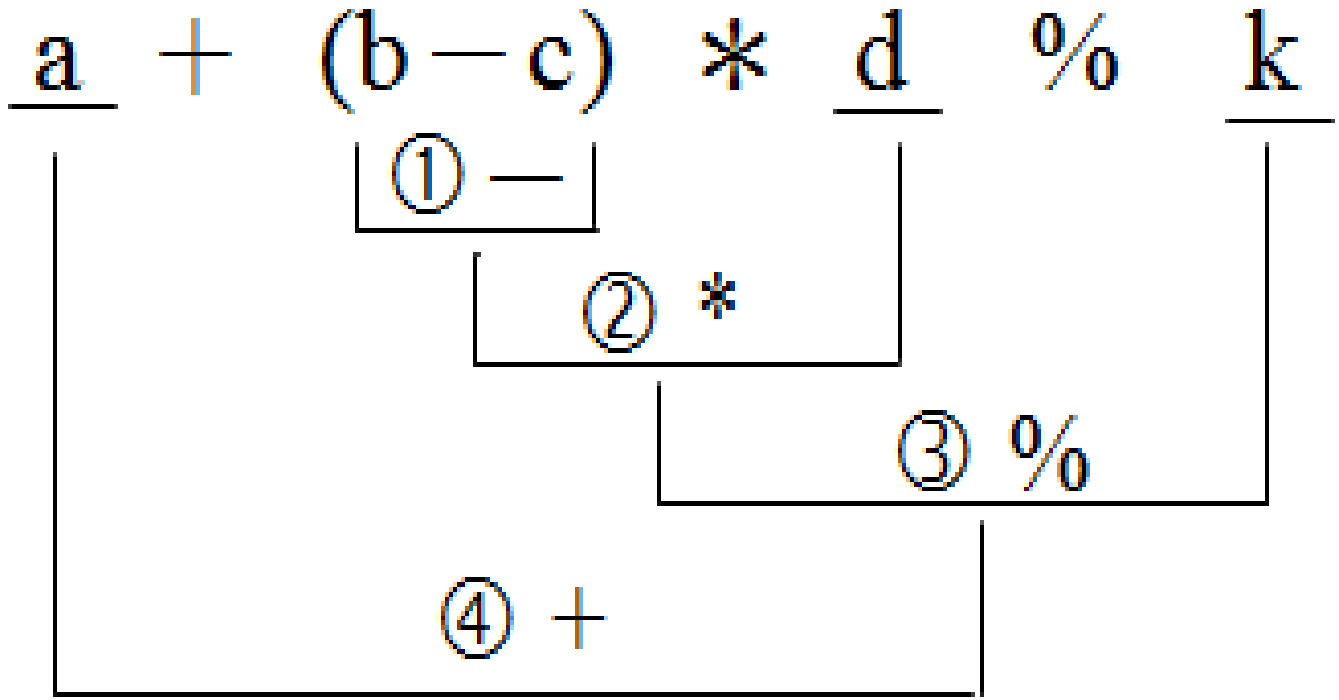
- Used for numeric data, binary unsigned integer or float
  ① Shift left 1 bit, multiplied by 2
  ② Shift right 1 bit, divided by 2
- Shift operators:
  ① << : left-shift operator
  ② >> : right-shift operator
- Ex:

```
int a=10;
Console.WriteLine(a>>1);
// 10₁₀ = 1010₂ ⇨ shift right 1 bit ⇨ 0101₂ = 5₁₀
Console.WriteLine(a<<2);
// 10₁₀ = 1010₂ ⇨ shift left 2 bit ⇨ 101000₂ = 40₁₀
```

2.4.8 Priority of operators

Priority	Operator	Description
1	()	Raise the priority of expression
1	[]	Used in array, index, subscript and attribute
2	+, -, !	Unary operator
2	++, --	Increment/decrement
3	*, /, %	Arithmetic operator
4	+, -	Arithmetic operator
5	>, >=, <, <=	Relative operator
6	==, !=	Relative operator
7	&&	AND logical operator
8	||	OR logical operator
9	?:	Conditional operator
10	=, +=, -=, *=, /=, %=	Assign operator

- **a + (b – c) * d % k statement's priority:**



2.5 Input and Output in Console Application

- Console is a defined class by system namespace
- Console provides basic functions to input and output from console
- Ex:

  Read or ReadLine methods get data from the keyboard

  Write or WriteLine methods show data on the screen

2.5.1 Write/WriteLine Method

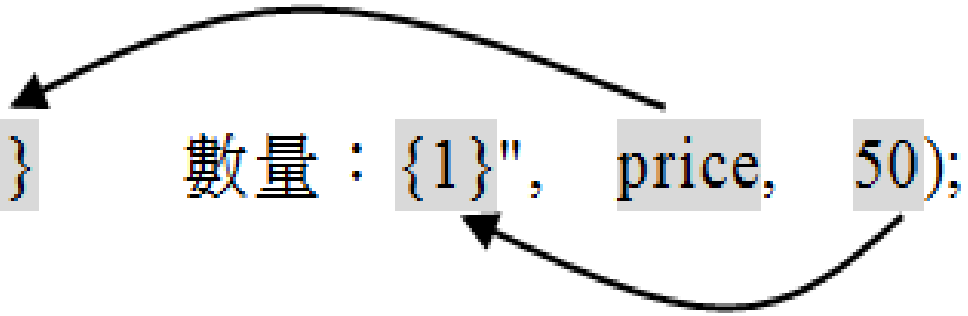
1. Console.Write() method

- Used for writing standard output data stream
- Make the string bracket in double quotation marks (“ ”)after Console.Write
  ⇨ show on the screen after the cursor
  the cursor stops at the end of shown string
- Ex:
  Let“光陰的故事”show after the current cursor:

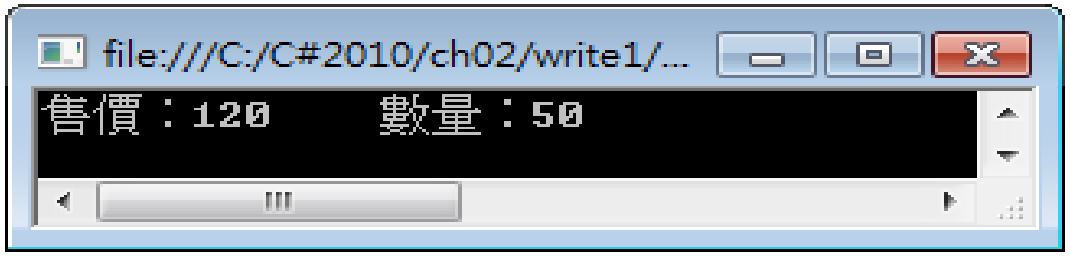
  Console.Write(“光陰的故事”);

- **Assigned variable and expression have to be placed in the middle of output string**
- **Assume the price is a variable 120, quantity is a constant 50. Then show “單價：120　數量：50” at the current of cursor**
- **Writing:**

```
int price=120;
Console.Write("售價：{0}　　數量：{1}",　price,　50);
```



Practice: write the source code



FileName : write1.sln

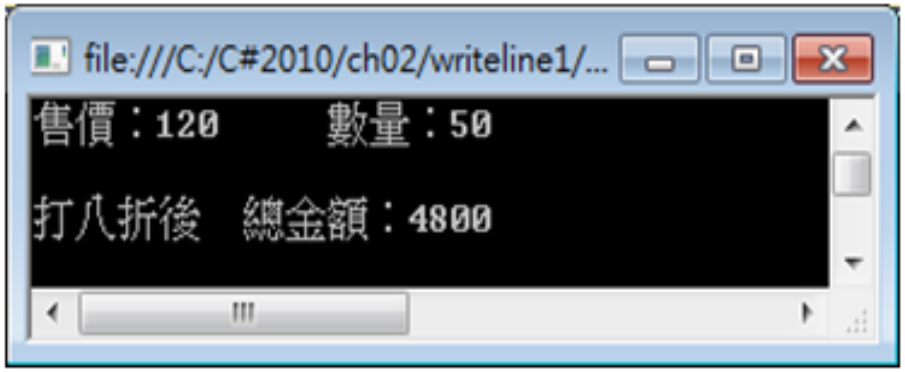
```
01 using System;
02 using System.Collections.Generic;
03 using System.Linq;
04 using System.Text;
05
06 namespace write1        //命名空間
07 {
08   class Program         //類別名稱
09   {
10      static void Main(string[] args)     // Main()方法為程式進入點
11      {
12         int price = 120;
13         Console.Write("售價：{0}   數量：{1}", price, 50);
14         Console.Read();
15      }
16   }
17 }
```

2. Console.WriteLine() Method

- Function is identical to Console.Write()
- Outputs the data stream
- Difference
  Console.WriteLine() will automatically insert the new line character
- Create a space line if Console.WriteLine() has no parameters in parantheses

Practice:

Please create the program which can show the following example, 3 Console.WriteLine() are required



FileName : writeline1.sln

```
01 namespace writeline1        //命名空間
02 {
03    class Program            //類別名稱
04    {
05       static void Main(string[] args)     //Main 方法為程式的進入點
06       {
07          //宣告整數變數 price 為 120，整數變數 qty 為 50
08          int price = 120, qty = 50;
09          //印出資料後游標往下移一行
10          Console.WriteLine("售價：{0}    數量：{1}", price, qty);
11          Console.WriteLine();          //游標往下移一行
12          Console.WriteLine("打八折後  總金額：{0}", price * qty * 0.8);
13          Console.Read();               //暫停主控台畫面，等待使用者按下鍵盤任意鍵
14       }
15    }
16 }
```

2. Read &ReadLine() Method

. Console.Read() Method

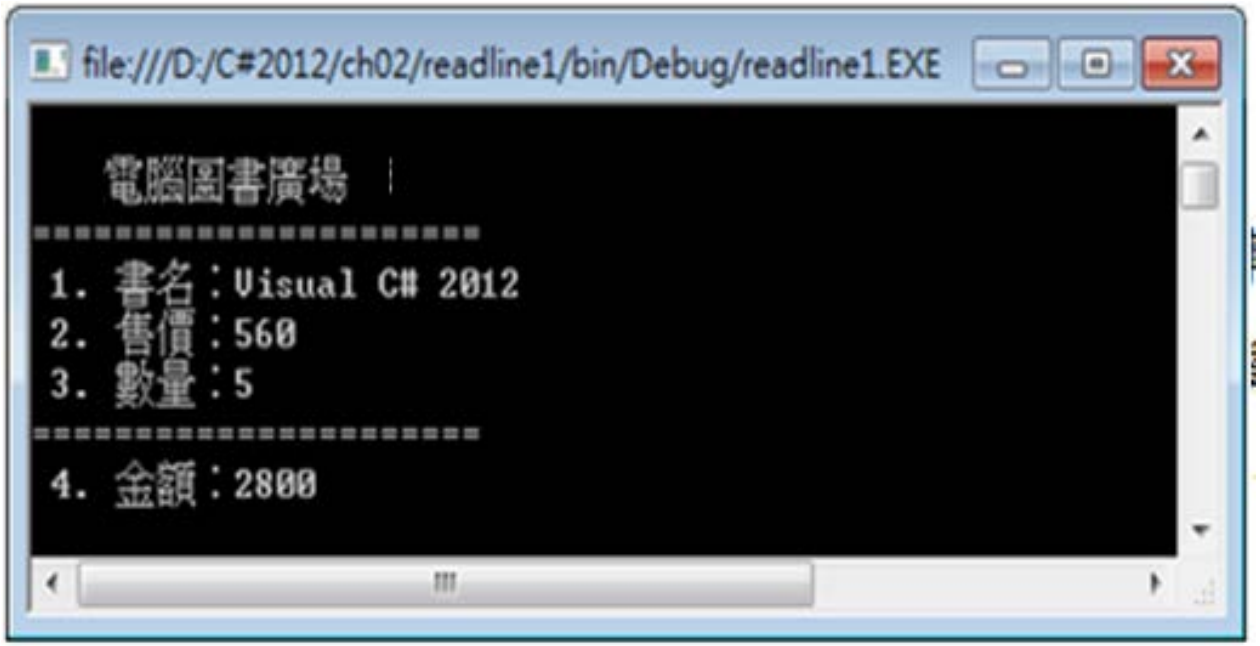
- **Read a character from standard input device (ex: keyboard). Enter key is not required**
- **Usage:**
  press any key to continue or allow only one character input.

. ReadLine() Method

- **Read a line of characters from standard input device (ex: keyboard) until the Enter key is pressed.**
- **Result's data type is string**
  int.Parse() or Convert.ToInt32() is used if the integer result is required (ex: Convert.ToInt32(Console.ReadLine()) , int.Parse(Console.ReadLine()))

Practice:

Try to use Console.ReadLine() to get the book name, price, quantity inputted from keyboard. Also int.Parse() or Convert.ToInt32() is used to convert the data type into integer. Finally use Console.WriteLine() to show the multiplication of price and the quantity



```
01 namespace readline1  // 命名空間
02 {
03    class Program       // 類別名稱
04    {
05       static void Main(string[] args)      // Main 方法為程式一開始的進入點
06       {
07          string str1;
08          int price, qty;
09          Console.WriteLine();
10          Console.WriteLine(" 電腦圖書廣場")
11          Console.WriteLine("======================");
12          Console.Write(" 1. 書名：");
13          str1 = Console.ReadLine();       // 輸入書名並指定給 str1 變數
14          Console.Write(" 2. 售價：");
15          // 輸入售價並使用 int.Parse()方法將輸入的資料轉成整數，再指定給 qty
16          price = int.Parse(Console.ReadLine());
17          Console.Write(" 3. 數量：");
18          // 輸入數量並使用 Convert.ToInt32()方法將輸入的資料轉成整數，再指定給 qt
19          qty = Convert.ToInt32(Console.ReadLine());
20          Console.WriteLine("======================");
21          Console.WriteLine(" 4. 金額：{0}", price * qty);
22          Console.Read();
23       }
24    }
```

2.6 Escape Sequence

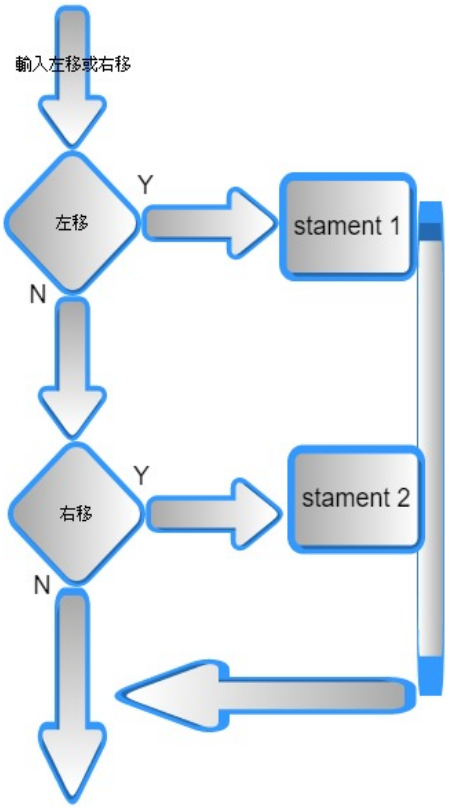
Escape char	Description
\'	Insert a single quote
\"	Insert a double quote
\\	Insert a back slash
\a	Trigger a system alarm sound
\b (Backspace)	Backspace
\f (Form Feed)	Insert a form feed
\n (New line)	Insert a new line
\r (Return)	Move cursor to the beginning of line
\t (Tab)	Insert a tab
\udddd	Insert an Unicode character
\v	Insert a vertical tab
\0 (Null space)	A null space

Example:

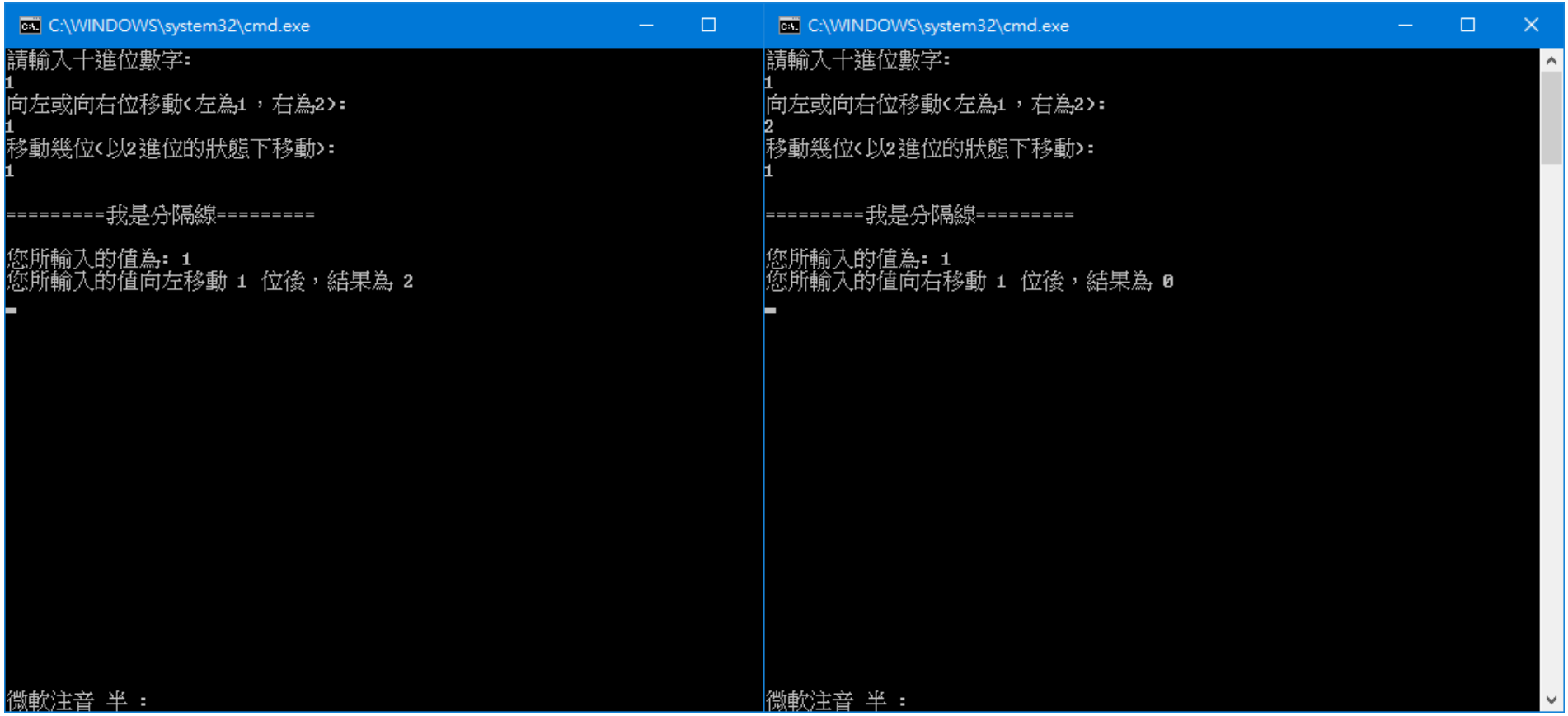
```
Console.WriteLine("\"鳥龍派出所\"");   //印出 "鳥龍派出所"
Console.WriteLine("Jack\'s Wang");     //印出 "Jack's Wang"
Console.WriteLine("Why 1\\2");         //印出 "Why 1\2"
```

Practice 1

- Tips:
  - Use << and >> to finish this program.
  - Use if, else if to decide what you want to do.



Practice 1


C:\WINDOWS\system32\cmd.exe
請輸入十進位數字:
1
向左或向右位移動(左為1，右為2):
1
移動幾位(以2進位的狀態下移動):
1
=========我是分隔線=========
您所輸入的值為: 1
您所輸入的值向左移動 1 位後，結果為 2
微軟注音 半 :
C:\WINDOWS\system32\cmd.exe
請輸入十進位數字:
1
向左或向右位移動(左為1，右為2):
2
移動幾位(以2進位的狀態下移動):
1
=========我是分隔線=========
您所輸入的值為: 1
您所輸入的值向右移動 1 位後，結果為 0
微軟注音 半 :

The End

Take a Break ....