# 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...

- 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

- Collection of statements forms a program
- Statement is formed by
   keyword, operator, variable, constant, expression, and so on
- 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." ("Objec
```

#### 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:

#### 2. Symbolic Constant

- Use meaningful variable name to replace the unchanged numbers or strings
- Cannot reassign the value during runtime
- Symbolic constant uses const keyword and expression to declare and assign initial value

```
Ex: Symbolic constant 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.c

om/zh-

tw/library/ba0a1yw2.asp

X

Available:

Follow

identifier

long naming short rule

decimal

float

int

double

char

string

**DateTime** 

bool

object

Cannot be leave out

const string a = "Hello";

#### Ex:

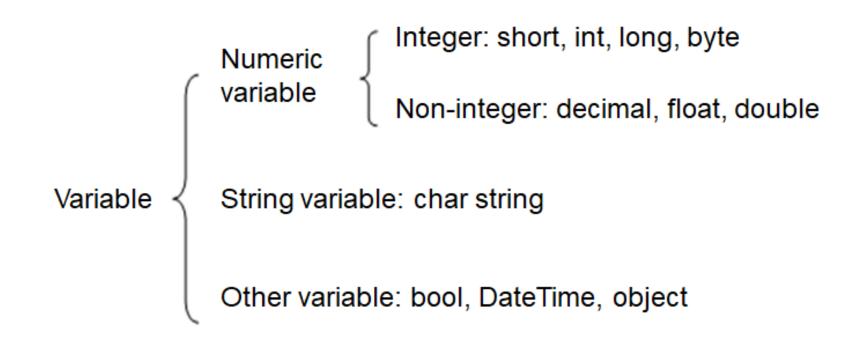
```
const int DaysInYear=365;
const int WeeksInYear=52;
const int may=5;
const double PI=3.1416;
const bool pass=true;
const string name="Viusal C# ";
```

 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
- Declare the variable before use it. When declaring variable:
  - give variable name
  - follow identifier naming rule
  - give data type for deploy memory space







$$X = Y + 10$$
;

- 10 is a constant memory stores 10, unchangeable
- ② X and Y are variableX and Y are changeable

### 7

#### 2.3 Declare Data Type of Variable

- Variables used in the program have to be declared in advance
- Declared variables' data type are known.
- 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	-792281625142643375935439 50335~ +792281625142643375935439 50335	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

- 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], ...;
```

M

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;

### M

#### 2.4 Operator and Expression

#### 2.4.1 Operator and operand

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

- ×
  - Unary Operatoronly 1 operand is requiredprefix notation, ex: -5
  - ② Binary Operator2 operands are required infix notation), ex: 5+8
  - 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) )

#### 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
	*	Multiply	area = height * width;
1	/	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

### 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# provides logical operators:
- 1. AND

А	В	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	В	$A \parallel B$
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	В	$\mathbf{A} \wedge \mathbf{B}$
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
/=	$x \neq y (x = x \neq y)$	x=5/2=2.5
%=	x %= y (x = x % y)	x=x%2=1
^=	XOR logic operation x^=y	$x=x^y \rightarrow 7$ Process: $0101_2=5_{10}$ $\underline{Xor\ 0010_2=2_{10}}$ $0111_2 \rightarrow 7_{10}$



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

### 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

### м.

#### 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);

// 10_{10} = 1010_2 \Rightarrow \text{ shift right 1 bit } \Rightarrow 0101_2 = 5_{10}

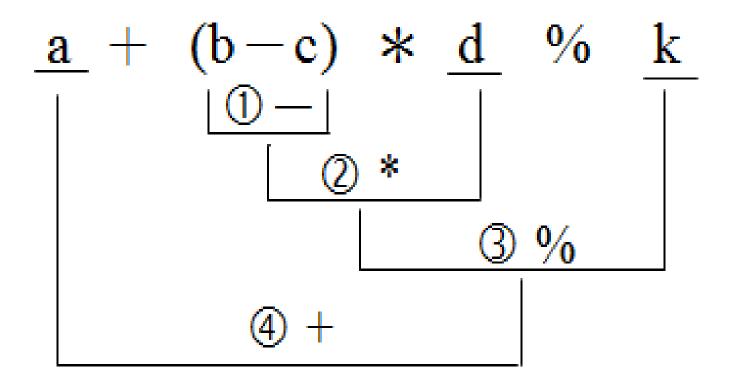
Console.WriteLine(a<<2);

// 10_{10} = 1010_2 \Rightarrow \text{ shift left 2 bit } \Rightarrow 101000_2 = 40_{10}
```

# 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	II	OR logical operator
9	?:	Conditional operator
10	=, +=, -=, *=, /=, %=	Assign operator

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



### 2.5 Datatype Conversion

- 1. Implicit Conversion is also called Automatic Conversion.
- When two types of data are compatible with each, or the range of the purpose of the datatype is wider than the original datatype, automatic conversion is allowed.

Datatype	Automatically-converted Types	
sbyte	short、int、long、float、double 或 decimal	
byte	short、ushort、int、uint、long、ulong、float、double 或 decimal	
short	int、long、float、double 或 decimal	
ushort	int、uint、long、ulong、float、double 或 decimal	
int	long、float、double 或 decimal	
uint	long、ulong、float、double 或 decimal	
long	float、double 或 decimal	
ulong	float、double 或 decimal	
float	double	
char	ushort、int、uint、long、ulong、float、double 或 decimal	

# 9

### 2. Explicit Conversion

Explicit Conversion uses the method of Type Cast to get a datatype converted into a new one.

- Double datatype variables are converted into int. integer variables a.
- $\bullet$  a = (int)x;

Syntax

variables=(cast) variables or formulas;

### 3.Parse method

- Datatypes read by keyboard from the method of ReadLine () or Read () are string values.
- If someone's height is keyboard-input as 175, it would be seen as a string value and can not be directly processed under mathematical operation.
- Method of Parse() must be used to convert string value into numerical ones.
- int.Parse(string)

Syntax

Numerical variable= datatype. Parse(string);

# 2.6 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

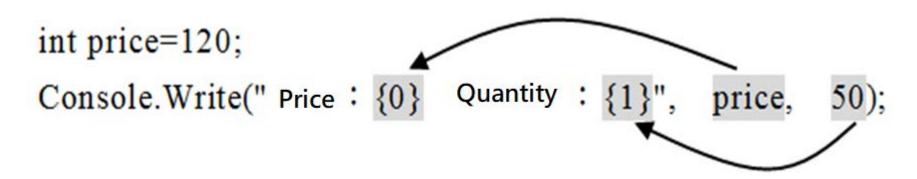
### м

### 2.6.1 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 "NCKU" show after the current cursor:

Console.Write("NCKU");

- H
  - Assigned variable or 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 "Price: 120 Quantity: 50" at the current of cursor
  - Writing:



#### Example: write1.sln



```
FileName : write1.sln
01 using System;
02 using System.Collections.Generic;
03 using System.Ling;
04 using System. Text;
05
06 namespace write1 //命名空間
07 {
    class Program //類別名稱
80
09
       static void Main(string[] args) // Main()方法為程式進入點
10
11
12
         int price = 120;
         Console.Write("售價:{0} 數量:{1}", price, 50);
13
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

#### **Example :writeline1.sln**

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
       static void Main(string[] args) //Main 方法為程式的進入點
05
06
07
          //官告整數變數 price 為 120,整數變數 qty 為 50
          int price = 120, qty = 50;
80
09
          //印出資料後游標往下移一行
10
          Console.WriteLine("售價:{0} 數量:{1}", price, qty);
          Console.WriteLine(); //游標往下移一行
11
          Console.WriteLine("打八折後 總金額: {0}", price * qty * 0.8);
12
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 }
```

# м

### 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()) )

#### Example :EX4.sln

```
■ file:///D:/ncku_school/開課課程/C#視窗程式設計參考簡報/...
                                                                                                         Х
 請輸入第 1 位學生的資料:
學號:1
生名:stu1
}數:90
請輸入第 2位學生的資料:
■
!##:2
   名:stu2
數:100
輸入資料如下...
第 1 位學生資料-->學號:1, 姓名:stu1, 成績90
第 2 位學生資料-->學號:2, 姓名:stu2, 成績100
```

#### Example: EX4.sln

```
Program.cs a → X
C# ex4

▼ % ex4.Program

          ∃using System;
           using System.Collections.Generic;
           using System.Linq;
          using System. Text;
          ⊟namespace ex4
               0 個參考
               class Program
     9
                   1 個參考
    10
                   struct Student
    11
    12
                       public String Id, Name;
    13
                       public int Score:
    14
                   0 個參考
    15
                   static void Main(string[] args)
    16
    17
                       Student Peter, Mary;
    18
                       Console.WriteLine("請輸入第 1 位學生的資料:");
    19
                       Console.Write("學號:");
    20
                       Peter.Id = Console.ReadLine();
    21
                       Console.Write("姓名:");
    22
                       Peter.Name = Console.ReadLine();
    23
                       Console.Write("分數:");
    24
                       Peter.Score = int.Parse(Console.ReadLine());
    25
                       Console.WriteLine("請輸入第 2位學生的資料:");
                       Console.Write("學號:");
    26
    27
                       Mary.Id = Console.ReadLine();
    28
                       Console.Write("姓名:");
    29
                       Mary.Name = Console.ReadLine();
    30
                       Console.Write("分數:");
    31
                       Mary.Score = int.Parse(Console.ReadLine());
    32
    33
                       Console.WriteLine();
    34
                       Console.WriteLine ("輸入資料如下...");
    35
    36
                       Console.WriteLine("第 1 位學生資料-->學號: {0}, 姓名: {1}, 成績{2}", Peter.Id, Peter.Name, Peter.Score.ToString());
                       Console.WriteLine("第 2 位學生資料-->學號: {0}, 姓名: {1}, 成績{2}", Mary.Id, Mary.Name, Mary.Score.ToString());
    37
    38
    39
                       Console.ReadLine();
    40
    41
    42
    43
```

## 2.7 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("\" Wang \""); // Print "Wang "-

Console. WriteLine("Jack\'s Wang"); // Print "Jack's Wang"

Console. WriteLine("Why 1\\2"); // Print "Why 1\\2"
```

# The End

Take a Break ....