

W3Schools C#線上編輯編譯執行器

— W3schools之C#語言線上教學手冊

— https://www.w3schools.com/cs/cs_intro.php

Home HTML CSS JAVASCRIPT SQL PYTHON PHP BOOTSTRAP HOW TO W3.CSS

C# Tutorial

- C# HOME
- C# Intro
- C# Get Started**
- C# Syntax
- C# Comments
- C# Variables
- C# Data Types
- C# Type Casting
- C# User Input
- C# Operators
- C# Math
- C# Strings
- C# Booleans
- C# If...Else
- C# Switch
- C# While Loop
- C# For Loop
- C# Break/Continue
- C# Arrays

C# Get Started

< Previous

Next >

C# IDE

The easiest way to get started with C#, is to use an IDE.

An IDE (Integrated Development Environment) is used to edit and compile code.

In our tutorial, we will use Visual Studio Community, which is free to download from <https://visualstudio.microsoft.com/vs/community/>.

Applications written in C# use the .NET Framework, so it makes sense to use Visual Studio, as the program, the framework, and the language, are all created by Microsoft.

C# 線上編輯編譯執行器

- C # 語言線上教學課程 (.NET線上編譯器)
 - <https://docs.microsoft.com/zh-tw/dotnet/csharp/tour-of-csharp/#code-try-0>
- 在 **Visual Studio** 中建立簡單的 c # 主控台應用程式
 - <https://docs.microsoft.com/zh-tw/visualstudio/get-started/csharp/tutorial-console?view=vs-2017>

Visual Studio IDE整合開發工具

- **Visual Studio Professional IDE各種版本下載**
([2022/2019/2017/2015/2013/2012](https://visualstudio.microsoft.com/zh-hant/vs/older-downloads/))
 - <https://visualstudio.microsoft.com/zh-hant/vs/older-downloads/>
- 本課程一律使用**Microsoft Visual Studio 2017 Professional Version**

第五講

陳述式

5.1

流程控制陳述式

陳述式 (Statements)

陳述式是一種程式碼指令 (instruction) 用以描述型別或告訴程式執行動作。簡單的陳述式直接使用分號 (;) 作為結束，至於一系列複雜的陳述式則可用區塊 (block) ({ ... }) 來表示。

陳述式主要可以分為下面三種類型：

- | 宣告陳述式 (Declaration statements)：主要是型別 (type) 宣告或變數 (variables) 使用等 (已在先前單元介紹過)。
- | 嵌入陳述式 (Embedded statements)：主要是利用運算式 (expression) (已在先前單元介紹過) 及流程控制 (flow-of-control) (將在這個單元介紹)，對物件和變數進行運算。
- | 標籤陳述式 (Labeled statements)：讓程式能跳入 (jump) 某一位置的陳述。

陳述式 (Statements)

```
int x = 5;           // 簡單的宣告
int z;               // 簡單的宣告
{                   // 開始區塊
    int y = 15;      // 簡單的宣告
    z = x + y;        // 嵌入陳述式(Embedded statements)
    top: y = 20;      // 標籤陳述式(Labeled statements)
    {                 // 巢狀區塊
    }
}                   // 結束區塊
```

陳述式 (Statements)

空陳述式 (empty statement) 可以使用在語法需要你輸入一個陳述式，但你卻不想要做任何動作的時候。

```
if (x < y)
    ; // 空的陳述式
else
    z = a + b; // 簡單的陳述式
```


流程控制的陳述式

(Flow-of-Control Statements)

C# 提供下列的流程控制 (flow-of-control) 方式：

- 條件執行 (Conditional execution) 藉著條件的判斷決定哪一個部分的程式碼要被執行或略過。以下為條件執行的陳述式：
 - if
 - if ... else
 - switch
- 迴圈陳述 (Looping statements) 重複執行某一部分的程式碼。以下為迴圈的陳述式：
 - while
 - do while
 - for
 - foreach

流程控制的陳述式

(Flow-of-Control Statements)

- 跳躍陳述式 (Jump statements) 改變控制流程，也就是將程式執行的控制權由一區域轉換至另一區域。以下為跳躍的陳述式：
 - break
 - continue
 - return
 - goto
 - throw

5.2

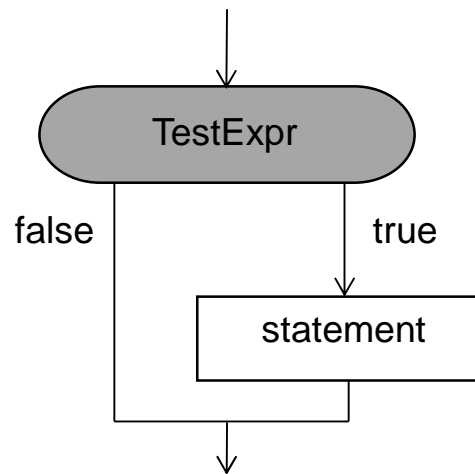
if 陳述式

if 陳述式

if 陳述式的語法：

```
if( TestExpr )  
    statement
```

- TestExpr 的評估結果必須為 **bool** 型別的值。
- 如果 TestExpr 的評估結果是正確的 (true)，陳述式 (statement) 就被執行。反之，如果評估結果是錯的 (false)，就跳過此一陳述式。



if 陳述式

```
if (x1 <= 10)
    z1 = x1 - 1;    //簡單的陳述式不需要大括號
```

```
if (x2 >= 20)
{
    //多個陳述式必須使用區塊
    x2 = x2 - 5;
    y2 = x2 + z2;
}
```

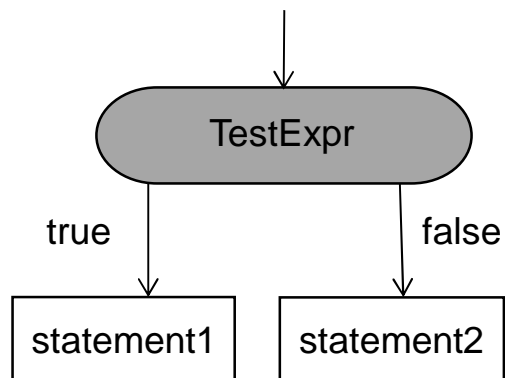
```
int x3 = 5;
if (x3)
{
    //錯誤:TestExpr必須是bool，不能是int
}
}
```

if ... else 陳述式

if...else陳述式的語法：

```
if( TestExpr )
    statement1
else
    statement2
```

- 如果TestExpr的評估結果是正確的(true)，陳述式1(statement1)就被執行；反之如果評估結果是錯的(false)，陳述式2(statement2)就被執行。



if ... else 陳述式

```
if (x1 <= 10)
    z1 = x1 - 1;    //簡單的陳述式不需要大括號
else
{
    //多個陳述式必須使用區塊
    x1 = x1 - 5;
    y1 = x1 + z1;
}
```

if ... else 陳述式

Statement1 及 Statement2 都可以再以 if 或 if ... else 陳述式取代。

```
if ( TestExpr1 )  
    if ( TestExpr2 )  
        statement11  
    else  
        statement12  
else  
    statement2
```

```
if ( TestExpr1 )  
    statement1  
else  
    if ( TestExpr2 )  
        statement21  
    else  
        statement22
```



```
if ( TestExpr1 )  
    statement1  
else if ( TestExpr2 )  
    statement21  
else  
    statement22
```


if ... else 陳述式

哪一個正確？

```
if ( TestExpr1 )  
    if ( TestExpr2 )  
        statement1  
else  
    statement2
```

```
if ( TestExpr1 )  
    if ( TestExpr2 )  
        statement1  
    else  
        statement2
```

規則：else會與在它之前最近的且沒有用到else的if為一對

if ... else 陳述式

哪裡有問題？

```
int x = 1;

if (x == 1)
{
    int y = 2;
    y++;
}
else
{
    int z = 3;
    --z;
}

Console.WriteLine("{0}, {1}, {2}", x, y, z);
```

Note：在區塊內宣告的變數，只能在其區塊內使用

5.3

switch 陳述式

switch 陳述式

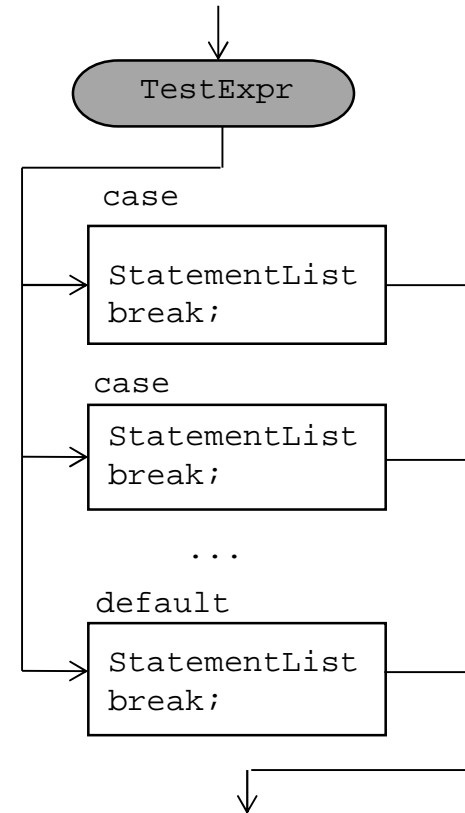
switch陳述式的語法：

```
switch( TestExpr )
{
    case ConstExpr1 :
        StatementList
        break;

    ...

    case ConstExprN :
        StatementList
        break;

    default:
        StatementList
        break;
}
```



- TestExpr在一開始就先被評估。
- 如果 TestExpr 的值跟 ConstExpr 的值一樣，在此一switch label內的陳述式們(StatementList)就會被執行，一直到出現跳躍陳述式為止。
- default 部分是可有可無的選項。若有此一陳述，一樣需要跳躍陳述式來做結尾。

switch 陳述式

```
int i = 4;

switch (i)                                     //評估 i 變數.
{
    case 1:                                   //如果i = 1
        Console.WriteLine("i is {0} -- In Case 1", i);
        break;                               //Go to end of switch.
    case 4:                                   //如果i = 4
        Console.WriteLine("i is {0} -- In Case 4", i);
        break;                               // Go to end of switch.
    default:                                 //如果i不是1和4
        Console.WriteLine("i is {0} -- In Default case", i);
        break;                               // 結束
}
```

switch 陳述式

哪裡有問題？

```
string s = "2"; //switch value不可用string

switch (s)
{
    case '1':
        Console.WriteLine("You are \"{0}\"", s);
        break;
    case '2':
        Console.WriteLine("You are \"{0}\"", s);
        break;
}
```

5.4

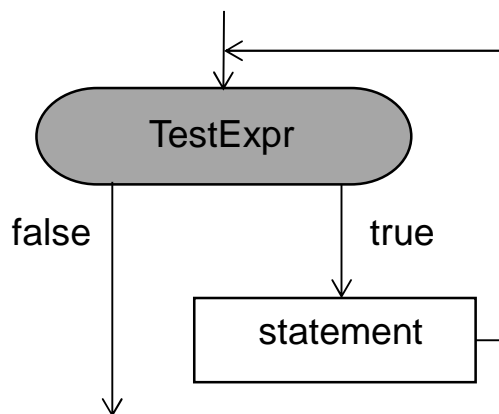
`while` 與 `do while` 迴圈

while 迴圈

while迴圈(loop)的語法：

```
while( TestExpr )  
    statement
```

- 首先評估TestExpr
- 如果TestExpr是false，則跳過陳述式的執行
- 如果TestExpr是true，陳述式會被執行，接著TestExpr會再被評估一次。
。每次只要是TestExpr評估結果是true，則陳述式會不斷被執行。迴圈會一直執行直到TestExpr的結果是false。



while 迴圈

```
int var1 = 3;
while (var1 > 0)
{
    Console.WriteLine("var1:  {0} ", var1);
    var1--;
}
Console.WriteLine("迴圈結束");
```



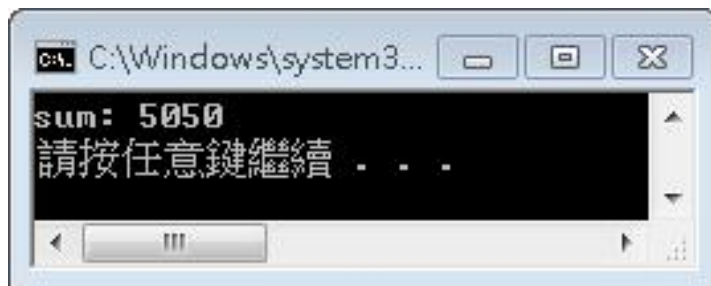
while 迴圈

如何利用while迴圈實作 $\text{sum} = 1 + 2 + 3 + \dots + 100$?

```
int sum = 0, i = 1;

while (i <= 100)
{
    sum += i;
    i++;
}

Console.WriteLine("sum: {0}", sum);
```



while 迴圈

哪裡有問題？

```
int var1 = 3;
while (var1 > 0)
{
    int var2 = 4;
    Console.WriteLine("var1: {0}, var2: {1} ", var1, var2);
    var1--;
    var2++;
}
Console.WriteLine("迴圈結束");
Console.WriteLine("var1: {0}, var2: {1} ", var1, var2);
```

Note：在區塊內宣告的變數，只能在其區塊內使用

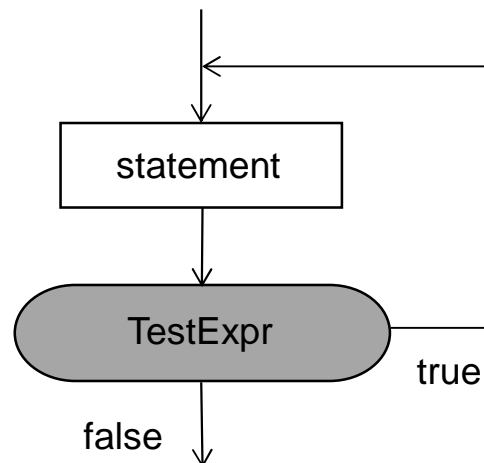
do while 迴圈

do迴圈(loop)就像其它迴圈一樣，只是它的測試陳述式在最下方。

```
do  
    statement  
while( TestExpr );
```

- 首先陳述式被執行
- 然後評估TestExpr
- 如果是true，陳述式會再被執行。
- 只要TestExpr評估結果一直為true，陳述式就會一直被執行，直到TestExpr 評估為false止。

要用分號來結束TestExpr的括號



do while 迴圈

```
int var1 = 3;  
do  
{  
    Console.WriteLine("var1:  {0} ", var1);  
    var1--;  
} while (var1 > 0);  
  
Console.WriteLine("迴圈結束");
```



5.5

for 迴圈

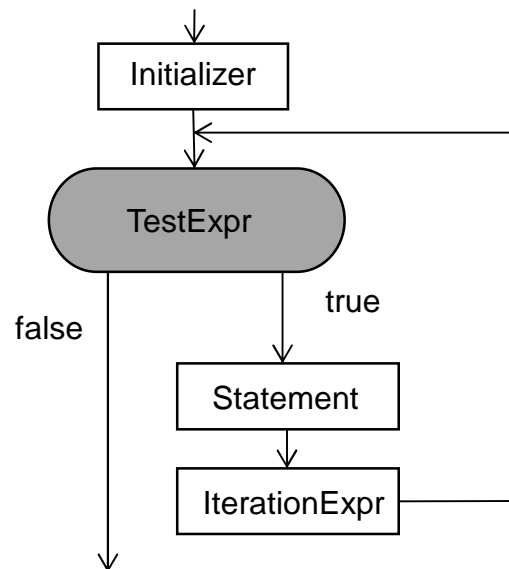
for 迴圈

for迴圈(loop)的語法：

```
for( Initializer ; TestExpr ; IterationExpr )  
    statement
```

- for迴圈一開始，Initializer會被執行一次(only one time)。通常是指定初始變數的值。
- TestExpr被評估。
- 如果 TestExpr 評估結果為true，陳述式(statement)將被執行，之後再執行IterationExpr。
- 接下來控制會回到迴圈的最前面，並評估 TestExpr 。同樣地，如果 TestExpr 評估結果為true，陳述式將被執行，之後再執行IterationExpr。
- 只有當TestExpr 評估為false，迴圈將結束執行。

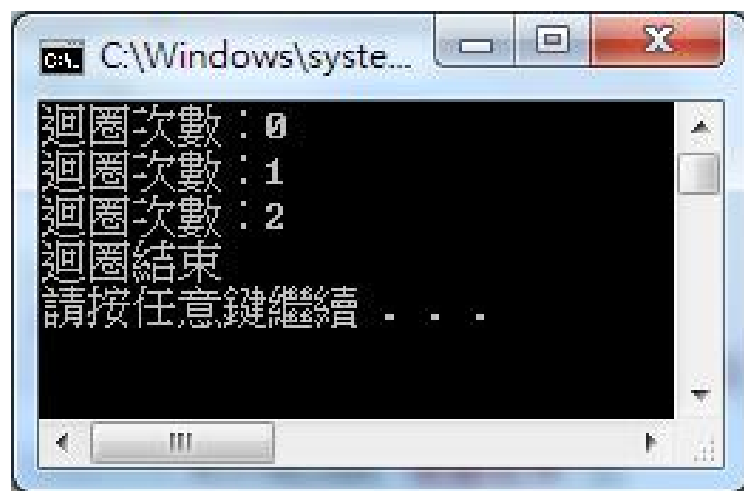
for 迴圈



`Initializer`、`TestExpr`和`IterationExpr`都可以選擇性的被空白，但分號(;)皆不能省略。如果`TestExpr`是空白的，則將會回傳`true`，因此將會有無窮迴圈的問題。

for 迴圈

```
//這個for迴圈會完整的被執行三次  
for (int i = 0; i < 3; i++)  
    Console.WriteLine("迴圈次數：{0} ", i);  
  
Console.WriteLine("迴圈結束");
```



for 迴圈

在Initializer宣告的變數，稱為迴圈變數(loop variables)，只有在陳述式(statement)內可被使用

```
int x = 0;
for (int i = 0; i < 10; i++)
{
    x = x + i;        // x += i;
    Console.WriteLine("i is: {0}", i);
}
Console.WriteLine("x will be: {0}", x);
```

```
i is: 0
i is: 1
i is: 2
i is: 3
i is: 4
i is: 5
i is: 6
i is: 7
i is: 8
i is: 9
x will be: 45
請按任意鍵繼續 . . .
```

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



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



for 迴圈

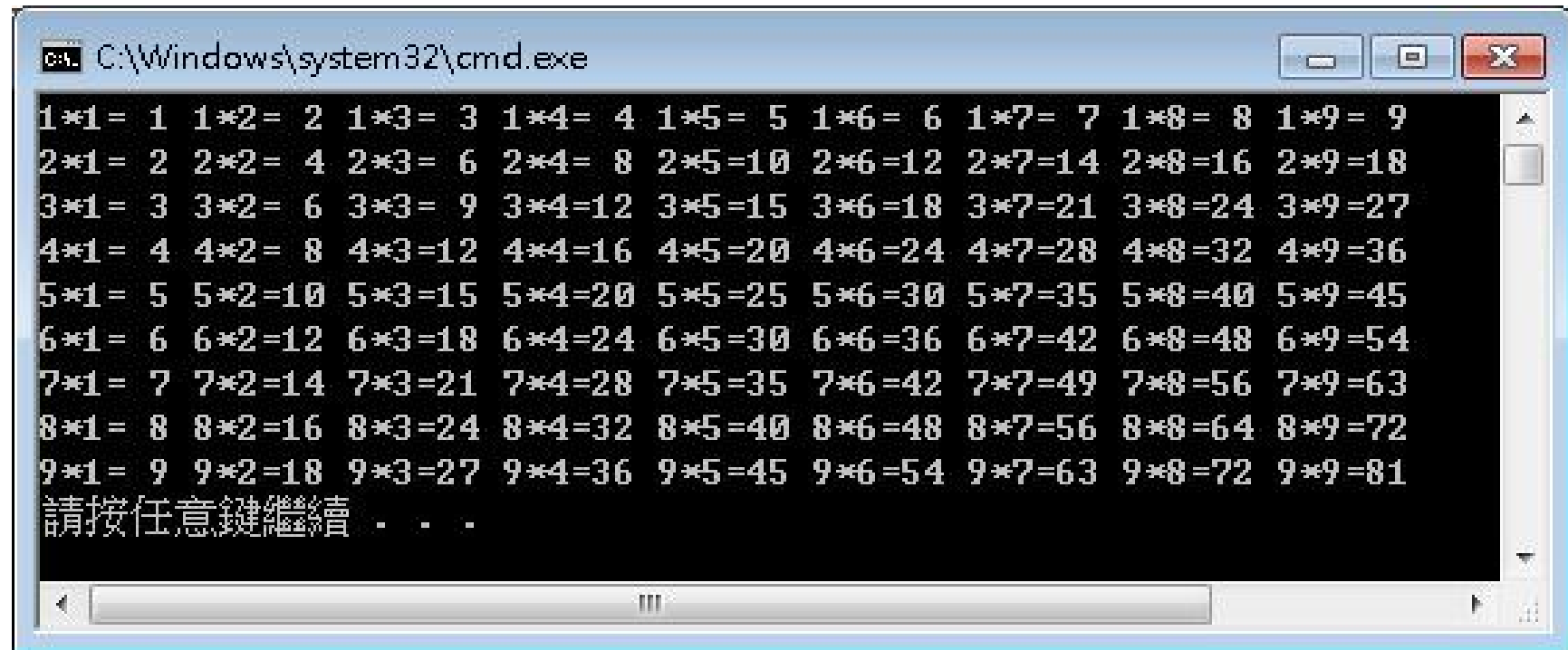
Initializer及IterationExpru皆可以包含多個運算式 (expressions)，但必須以逗號(,)分隔開。

```
for (int i = 10, j=10; i >0; i--, j+=10)  
    Console.WriteLine("i is: {0}, j is: {1}", i, j);
```

```
i is: 10, j is: 10  
i is: 9, j is: 20  
i is: 8, j is: 30  
i is: 7, j is: 40  
i is: 6, j is: 50  
i is: 5, j is: 60  
i is: 4, j is: 70  
i is: 3, j is: 80  
i is: 2, j is: 90  
i is: 1, j is: 100  
請按任意鍵繼續 . . .
```

For 迴圈

如何利用兩個for迴圈，印出九九乘法表？



```
cmd C:\Windows\system32\cmd.exe
1*1= 1 1*2= 2 1*3= 3 1*4= 4 1*5= 5 1*6= 6 1*7= 7 1*8= 8 1*9= 9
2*1= 2 2*2= 4 2*3= 6 2*4= 8 2*5=10 2*6=12 2*7=14 2*8=16 2*9=18
3*1= 3 3*2= 6 3*3= 9 3*4=12 3*5=15 3*6=18 3*7=21 3*8=24 3*9=27
4*1= 4 4*2= 8 4*3=12 4*4=16 4*5=20 4*6=24 4*7=28 4*8=32 4*9=36
5*1= 5 5*2=10 5*3=15 5*4=20 5*5=25 5*6=30 5*7=35 5*8=40 5*9=45
6*1= 6 6*2=12 6*3=18 6*4=24 6*5=30 6*6=36 6*7=42 6*8=48 6*9=54
7*1= 7 7*2=14 7*3=21 7*4=28 7*5=35 7*6=42 7*7=49 7*8=56 7*9=63
8*1= 8 8*2=16 8*3=24 8*4=32 8*5=40 8*6=48 8*7=56 8*8=64 8*9=72
9*1= 9 9*2=18 9*3=27 9*4=36 9*5=45 9*6=54 9*7=63 9*8=72 9*9=81
請按任意鍵繼續 . . .
```

5.6

跳躍陳述式

跳躍陳述式

當流程控制遇到跳躍 (Jump) 陳述時，程式的執行將會無條件的轉移到另一個地方。以下為跳躍 (Jump) 陳述式：

- break
- continue
- return
- goto
- throw

break 陳述式

break陳述式除了用在switch陳述式之外，也用在下列的迴圈陳述內，用以跳離最內層的迴圈(innermost enclosing loop)：

- for
- foreach
- while
- do while

```
int var1 = 0;
while (true)
{
    var1++;
    if (var1 >= 5)
        break;
    Console.WriteLine("var1: {0}", var1);
}
```

```
var1: 1
var1: 2
var1: 3
var1: 4
請按任意鍵繼續 . . .
```

continue 陳述式

continue陳述式用在下列的迴圈陳述內，讓程式執行回到最內層迴圈(innermost enclosing loop)的頂端(top)：

- for
- foreach
- while
- do while

```
for (int i = 1; i < 6; i++)      //執行五次迴圈
{
    if (i < 3)                  //前三次迴圈，回到迴圈頂層
        continue;
    Console.WriteLine("Value of i is {0}", i);
}
```

```
Value of i is 3
Value of i is 4
Value of i is 5
請按任意鍵繼續 . . .
```


Lab 4

1. 用while迴圈計算 $1+2+3+\dots+100$ 的結果
(Ans: 5050)
2. 用while迴圈計算 $4+8+12+\dots+100$ 的結果
(Ans: 1300)
3. 要求使用者輸入一個整數n，並用while迴圈計算 $1+2+3+\dots+n$ 的結果
5. 利用for迴圈計算 $1+2+3+\dots+100$ ，但若遇到7的倍數則不加 (Ans: 4315)
6. 利用巢狀的for迴圈列印九九乘法表的內容

程式設計理念

目標

1+2+3+4

直覺想法

```
result = 0 + 1 + 2 + 3 + 4
```

程式

```
result = 0;
```

```
result = result + 1;
```

```
result = result + 2;
```

```
result = result + 3;
```

```
result = result + 4;
```

類似的敘述重複

C#程式(1)

```
int i = 1;  
int result = 0;
```

```
while (i <= 4)  
{  
    result = result + i;  
    i++;  
}
```

```
Console.WriteLine("The answer is: {0}", result);
```

Lab 4

C#程式(2)

```
int i = 0;
```

```
int result = 0;
```

```
while (i <= 15)
```

```
{
```

```
    result = result + i;
```

```
    i = i+4;
```

```
}
```

```
Console.WriteLine("The answer is: {0}", result);
```

Lab 4

C#程式(3)

```
int i = 0;
int result = 0;
Console.WriteLine("輸入 n: ");
int n = Convert.ToInt32(Console.ReadLine());
while (i <= n)
{
    result = result + i;
    i++;
}
Console.WriteLine("The answer is: {0}", result);
Console.ReadKey();
```


Lab 4

C#程式設計(4)

```
int result = 0;
```

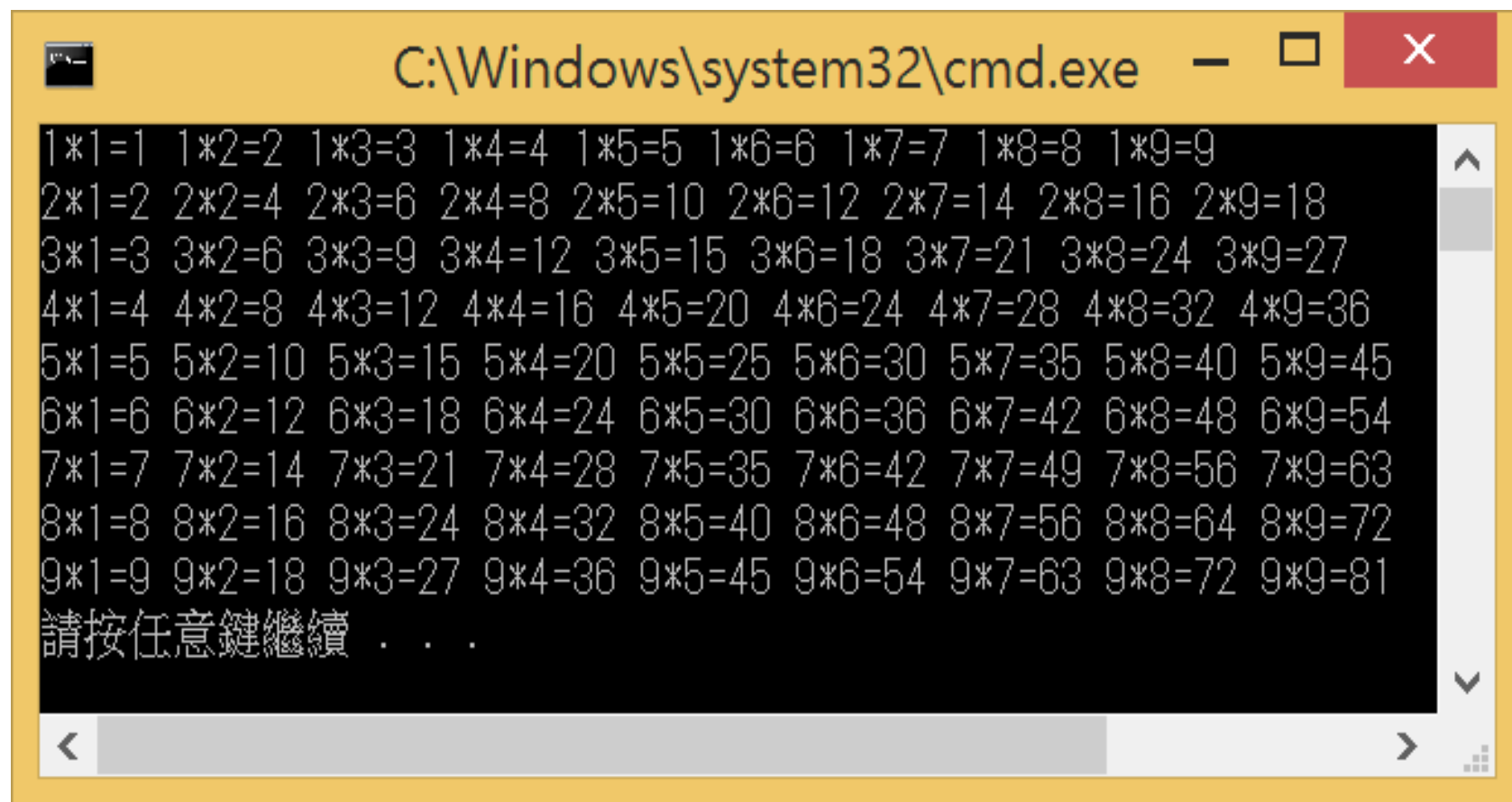
```
for (int i = 1; i <= 100; i++)  
{  
    if (i % 7 != 0)  
        result = result + i;  
}
```

```
Console.WriteLine("The answer is: {0}", result);
```

Lab 4

C#程式設計

```
for (int i = 1 ; i <= 9; i++)  
{  
    for (int j = 1; j <= 9; j++)  
    {  
        Console.Write( i + "*" + j + "=" + (i * j).ToString("00") + " " );  
    }  
    Console.Write("\r\n");  
}  
Console.ReadLine();
```



A screenshot of a Windows command prompt window. The title bar is yellow and contains the text "C:\Windows\system32\cmd.exe" along with standard window control buttons (minimize, maximize, close). The main area is black with white text. It displays a 9x9 multiplication table. The first row is "1*1=1 1*2=2 1*3=3 1*4=4 1*5=5 1*6=6 1*7=7 1*8=8 1*9=9". The second row is "2*1=2 2*2=4 2*3=6 2*4=8 2*5=10 2*6=12 2*7=14 2*8=16 2*9=18". The third row is "3*1=3 3*2=6 3*3=9 3*4=12 3*5=15 3*6=18 3*7=21 3*8=24 3*9=27". The fourth row is "4*1=4 4*2=8 4*3=12 4*4=16 4*5=20 4*6=24 4*7=28 4*8=32 4*9=36". The fifth row is "5*1=5 5*2=10 5*3=15 5*4=20 5*5=25 5*6=30 5*7=35 5*8=40 5*9=45". The sixth row is "6*1=6 6*2=12 6*3=18 6*4=24 6*5=30 6*6=36 6*7=42 6*8=48 6*9=54". The seventh row is "7*1=7 7*2=14 7*3=21 7*4=28 7*5=35 7*6=42 7*7=49 7*8=56 7*9=63". The eighth row is "8*1=8 8*2=16 8*3=24 8*4=32 8*5=40 8*6=48 8*7=56 8*8=64 8*9=72". The ninth row is "9*1=9 9*2=18 9*3=27 9*4=36 9*5=45 9*6=54 9*7=63 9*8=72 9*9=81". Below the table, the text "請按任意鍵繼續 . . ." is displayed. The window has a yellow border and a scroll bar on the right side.

```
C:\Windows\system32\cmd.exe
```

1*1=1	1*2=2	1*3=3	1*4=4	1*5=5	1*6=6	1*7=7	1*8=8	1*9=9
2*1=2	2*2=4	2*3=6	2*4=8	2*5=10	2*6=12	2*7=14	2*8=16	2*9=18
3*1=3	3*2=6	3*3=9	3*4=12	3*5=15	3*6=18	3*7=21	3*8=24	3*9=27
4*1=4	4*2=8	4*3=12	4*4=16	4*5=20	4*6=24	4*7=28	4*8=32	4*9=36
5*1=5	5*2=10	5*3=15	5*4=20	5*5=25	5*6=30	5*7=35	5*8=40	5*9=45
6*1=6	6*2=12	6*3=18	6*4=24	6*5=30	6*6=36	6*7=42	6*8=48	6*9=54
7*1=7	7*2=14	7*3=21	7*4=28	7*5=35	7*6=42	7*7=49	7*8=56	7*9=63
8*1=8	8*2=16	8*3=24	8*4=32	8*5=40	8*6=48	8*7=56	8*8=64	8*9=72
9*1=9	9*2=18	9*3=27	9*4=36	9*5=45	9*6=54	9*7=63	9*8=72	9*9=81

請按任意鍵繼續 . . .