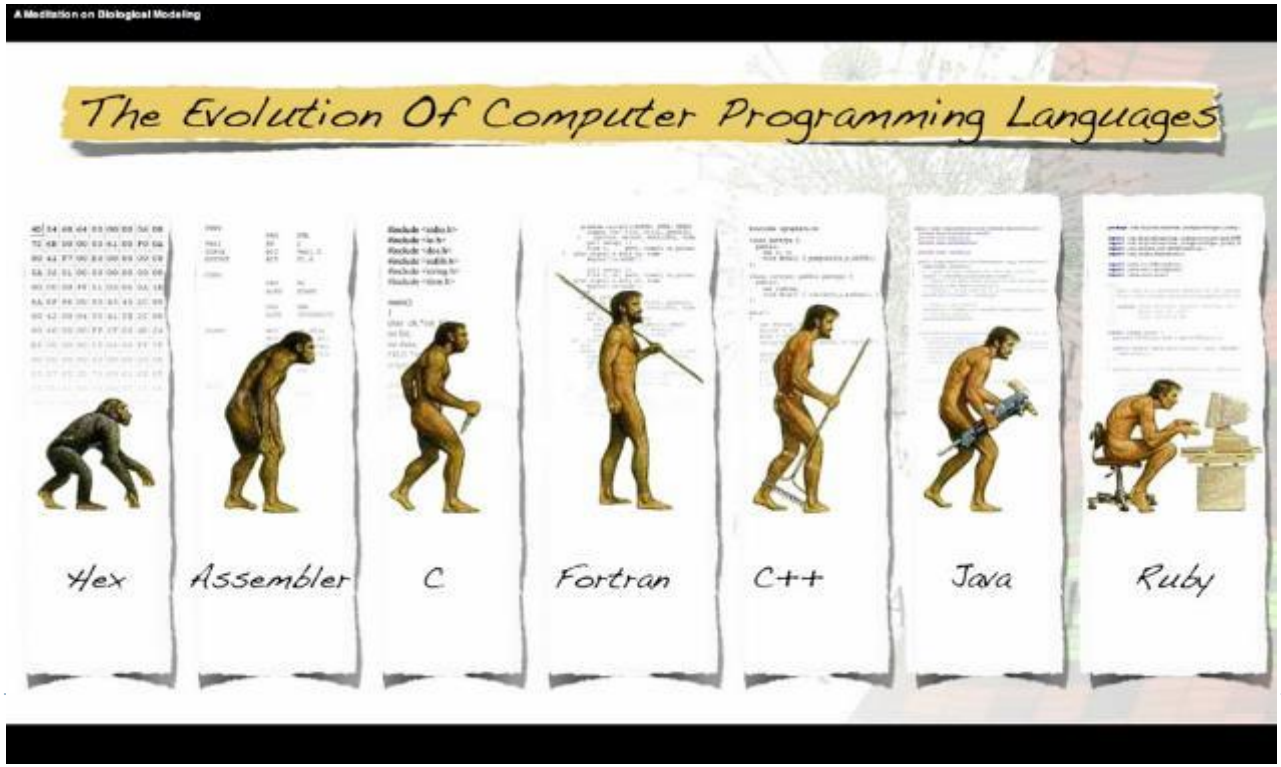
程式語言(programming language)

使用Visual Basic　劉和師

Part 1

何謂程式語言

- **程式語言**（programming language），是用來定義電腦**程式**的形式**語言**。 它是一種被標準化的交流技巧，用來向電腦發出指令。
  - 維基百科，自由的百科全書 - Wikipedia



何謂程式語言

- 程式就是指揮電腦動作的一連串指令
- 只要用到電腦的地方，就需要程式

何謂程式語言

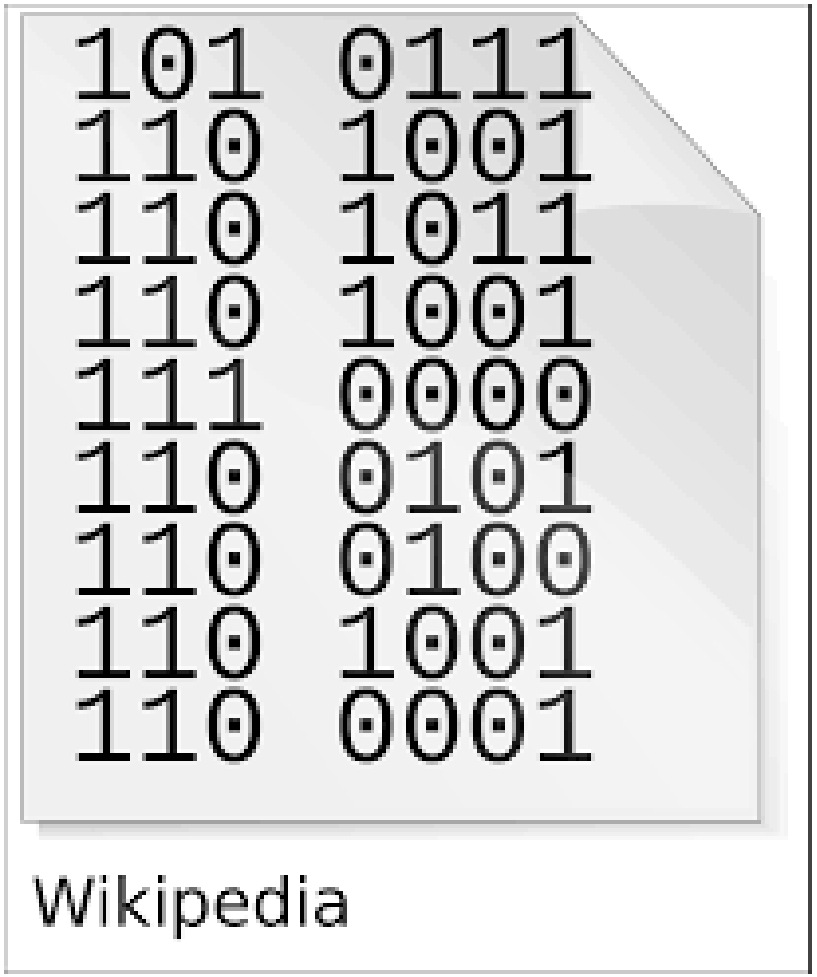
- 人的語言有中文、英文、日文等，說法不同，但都表達同一個意思

何謂程式語言

- 程式也有多種語言，寫法也許不同，但都可以表達同一個意思

最底層的機械語言

- 其實電腦只認得這個...
- 電腦內部是數位的世界，只認得0與1



- 這叫機械語言，但這對人類來說太困難了...

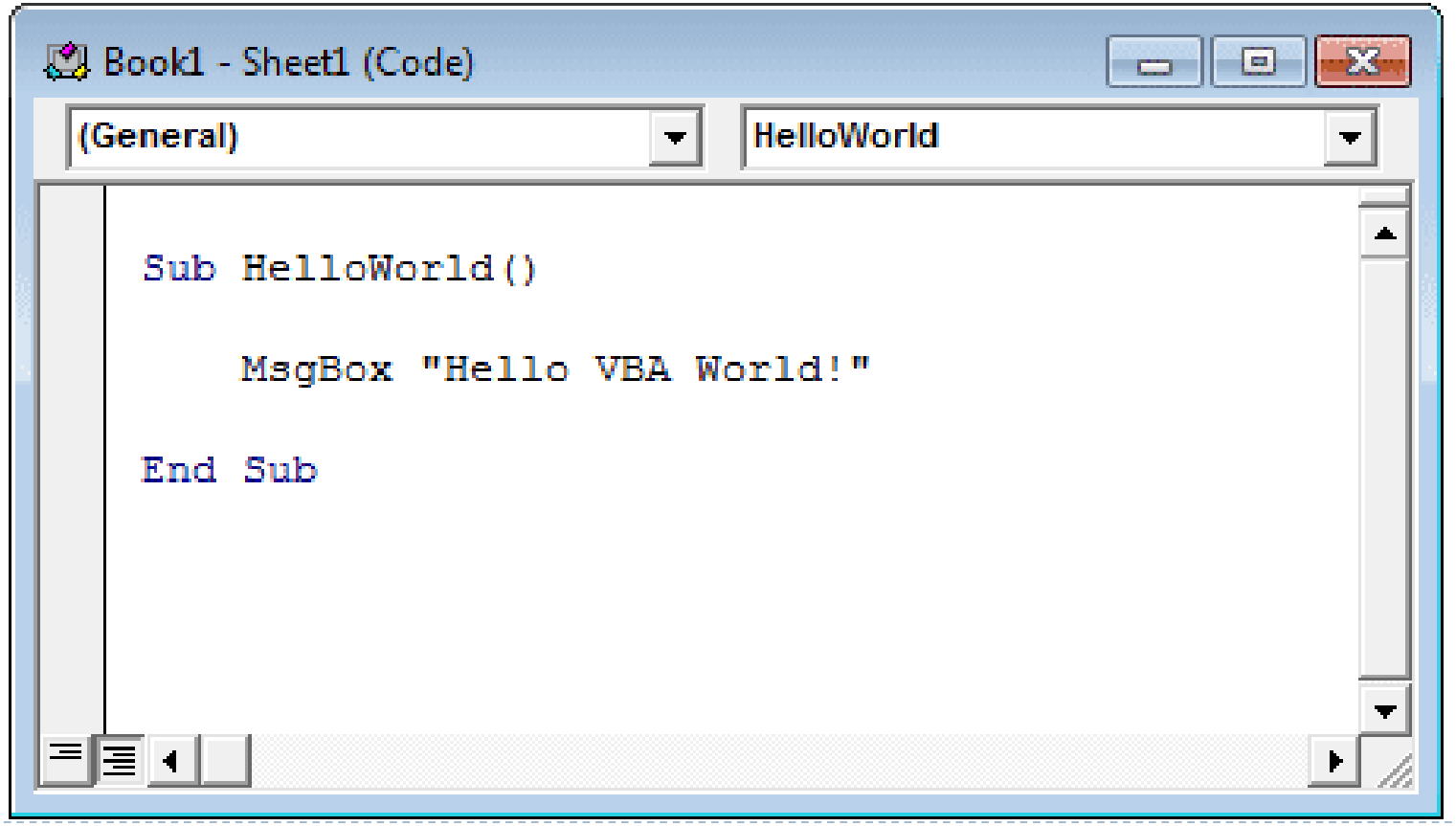
組合語言

- 所以有了組合語言(Assembly)，它使用助憶碼（Mnemonics）來代替和表示特定低階機器語言的操作，硬體工程師一定要會
- 但還是很難...

			記憶體位址	機器碼
	MOV	SI, 1000H	126B：0100	BE 00 10
	MOV	DI, 2000H	126B：0103	BF 00 20
	MOV	CX, 10H	126B：0106	B9 10 00
LOOP:	MOV	AL, [SI]	126B：0109	8A 04
	MOV	[DI], AL	126B：010B	88 05
	INC	SI	126B：010D	46
	INC	DI	126B：010E	47
	DEC	CX	126B：010F	49
	JNZ	LOOP	126B：0110	75 w
	HLT		126B：0112	F4

高階語言

- 為了更方便與電腦溝通，更像人在講話，所以有了高階語言



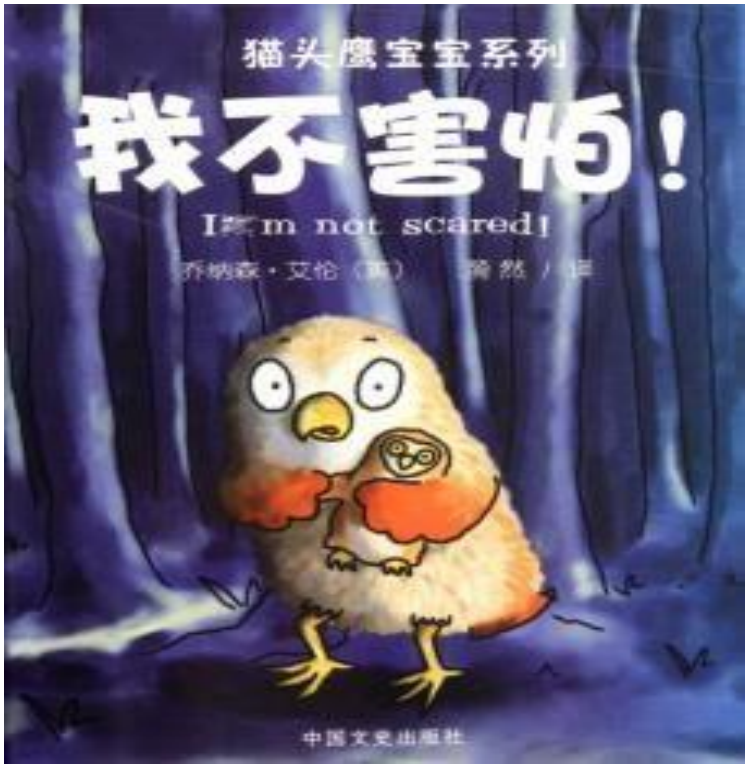
唯一的困難

- 高階語言有很多種，應用的領域也不同
- 唯一相同的是，都是用

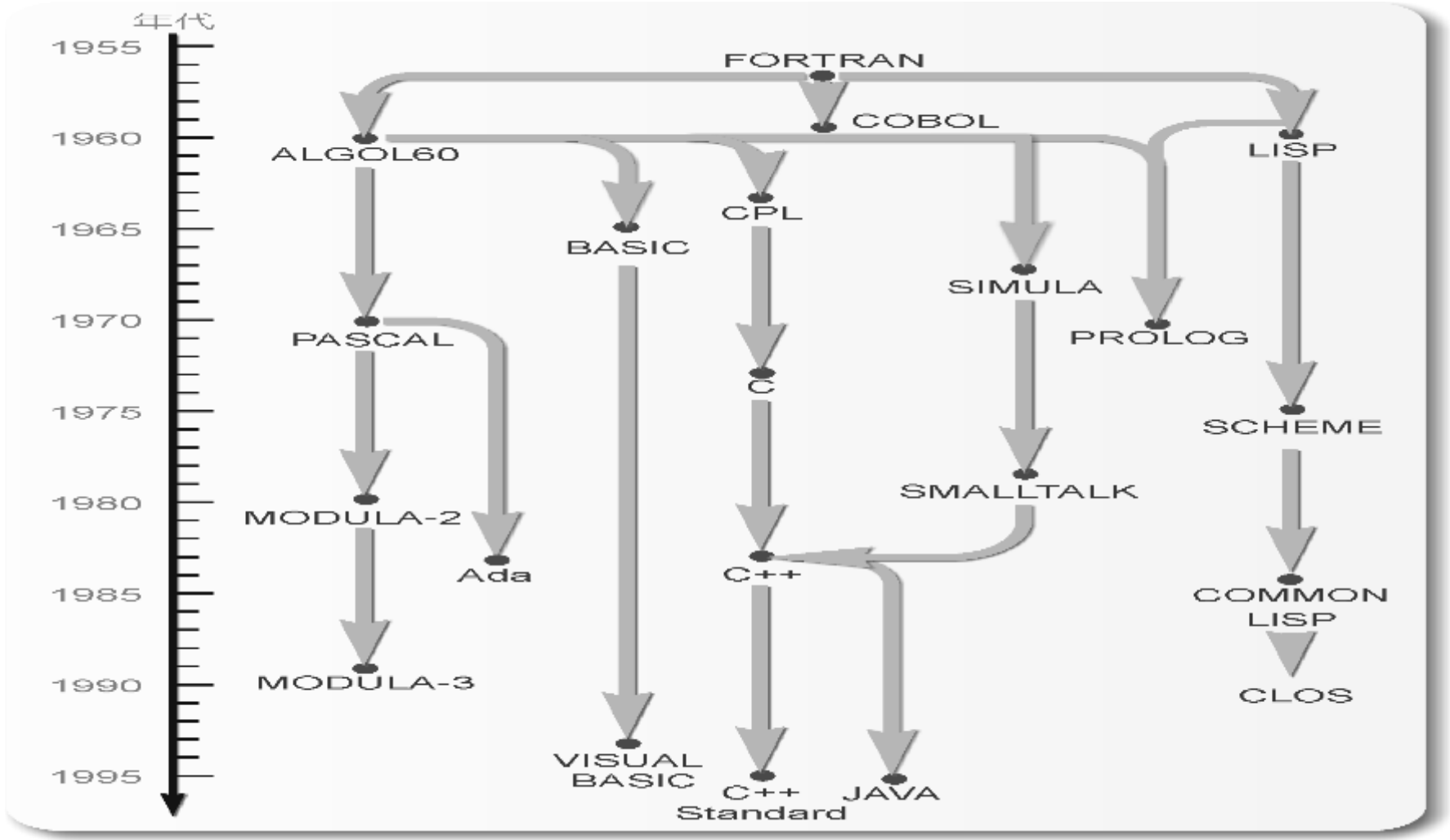
- 英文!!

我們用英文，但這不是英文課

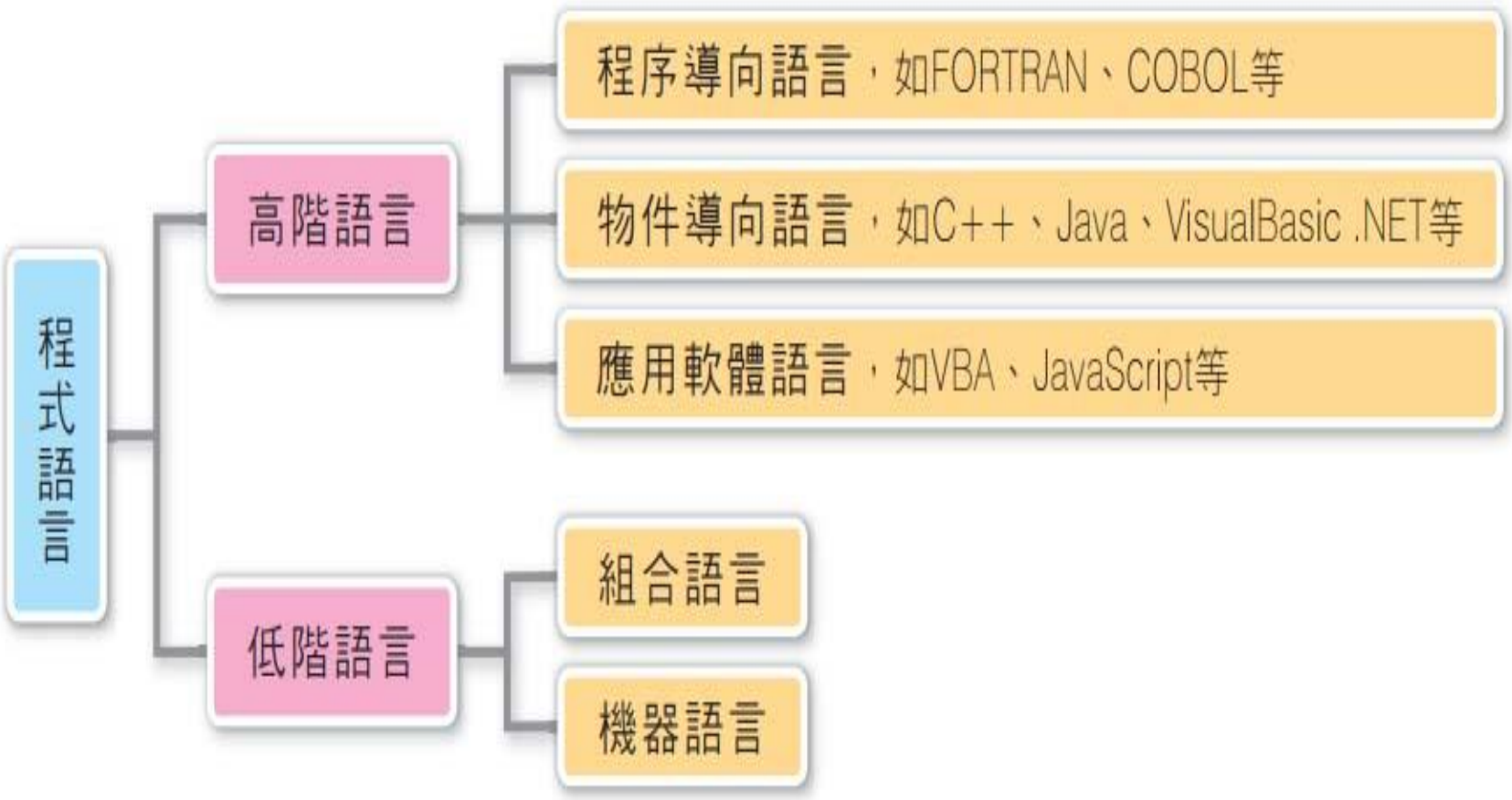
- 不用害怕，你要克服的不是英文，是心理障礙！
- 程式用到的英文很簡單，沒有甚麼過去式、未來式這些，只有簡單的命令



程式語言發展的歷史


年代
1955
1960
1965
1970
1975
1980
1985
1990
1995
FORTRAN
COBOL
ALGOL60
LISP
BASIC
CPL
SIMULA
PROLOG
PASCAL
C
SCHEME
SMALLTALK
MODULA-2
Ada
C++
COMMON LISP
MODULA-3
CLOS
VISUAL BASIC
C++ Standard
JAVA

程式語言的分類


程式語言
高階語言
程序導向語言，如FORTRAN、COBOL等
物件導向語言，如C++、Java、VisualBasic .NET等
應用軟體語言，如VBA、JavaScript等
低階語言
組合語言
機器語言

程式語言的比較

表1-1 低階語言與高階語言的比較

比較項目	機器語言	組合語言	高階語言
程式的撰寫	難 ⟷		易
維護與除錯	難 ⟷		易
可讀性	低 ⟷		高
可攜性[註]	低 ⟷		高
執行速度	快 ⟷		慢
佔用記憶體的空間	小 ⟷		大

其他方式與電腦溝通

- 現在也有這種積木式語言，方便易學，但功能有限，尚無法做出非常複雜的東西，適合小朋友

運算思維

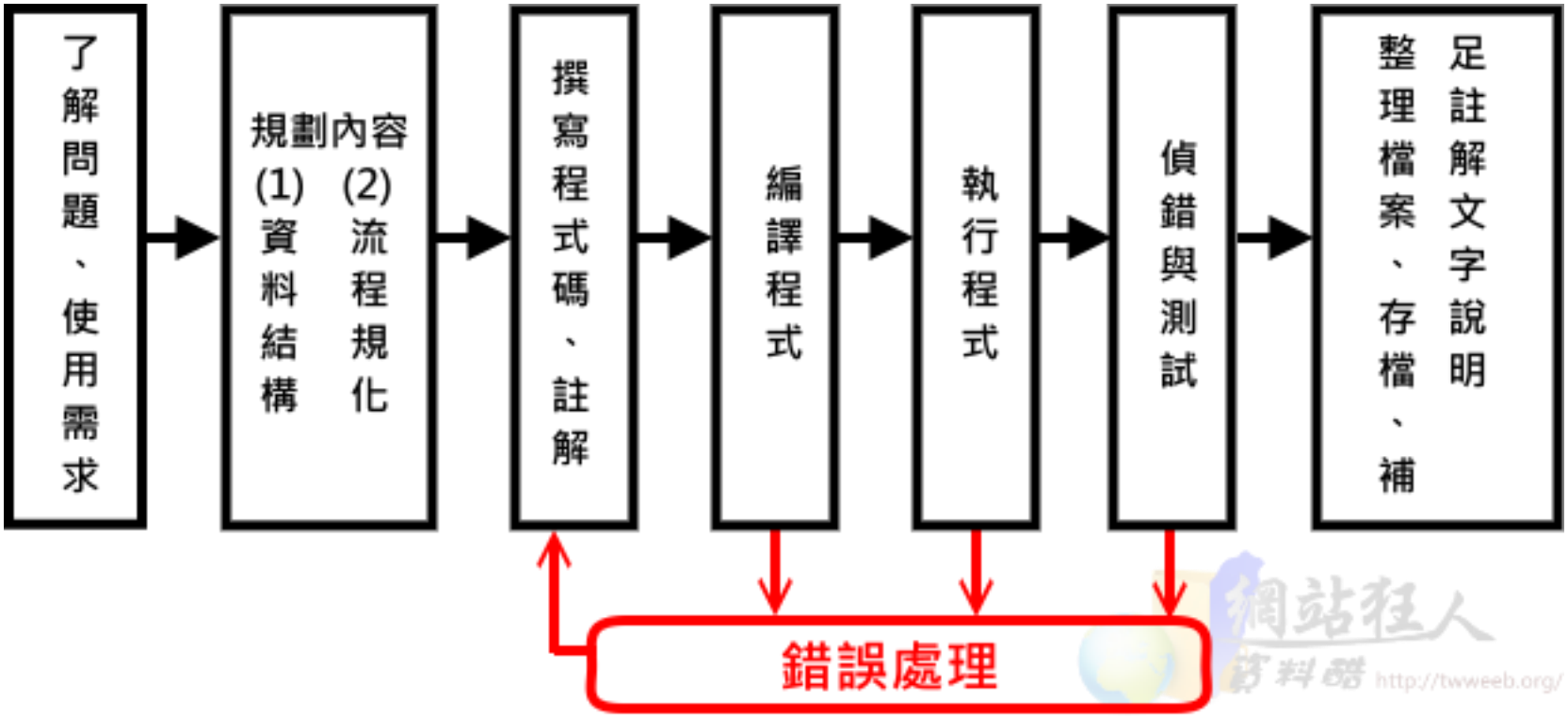
- 你平時如何解決問題?
- 面對問題，你會先好好想一下嗎?
- 你的思考程序是甚麼?

運算思維

- 運算思維是指能構思一個有條理的程序，應用各種運算方法或工具來解決問題的思維能力。
- 把你的問題分析好，再分解成一個個電腦可以幫你解決的步驟。
- 你的大腦就是一台電腦，先在大腦中跑過一遍，再去用電腦實現。

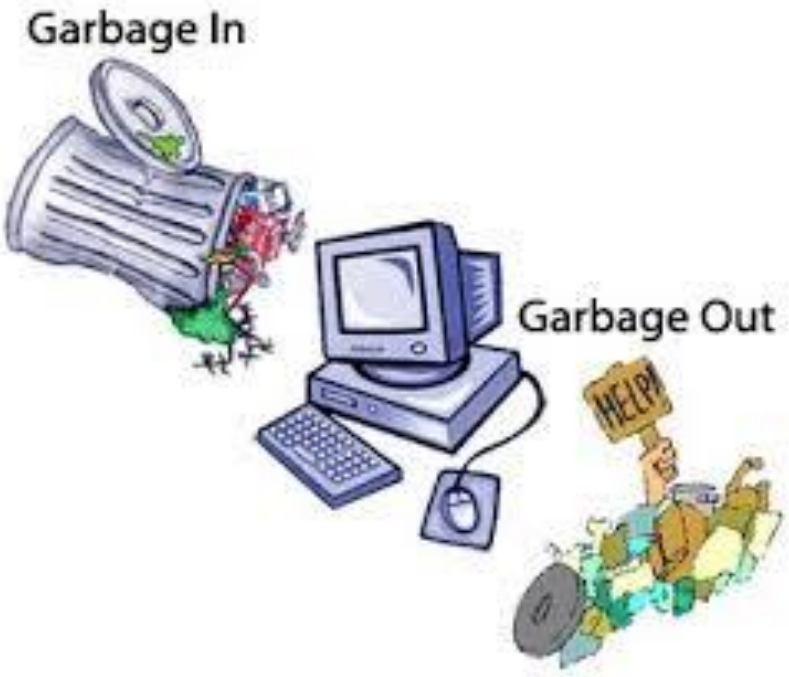
寫程式的流程

- 不是拿到問題就傻傻地做到電腦前
- 是如下圖



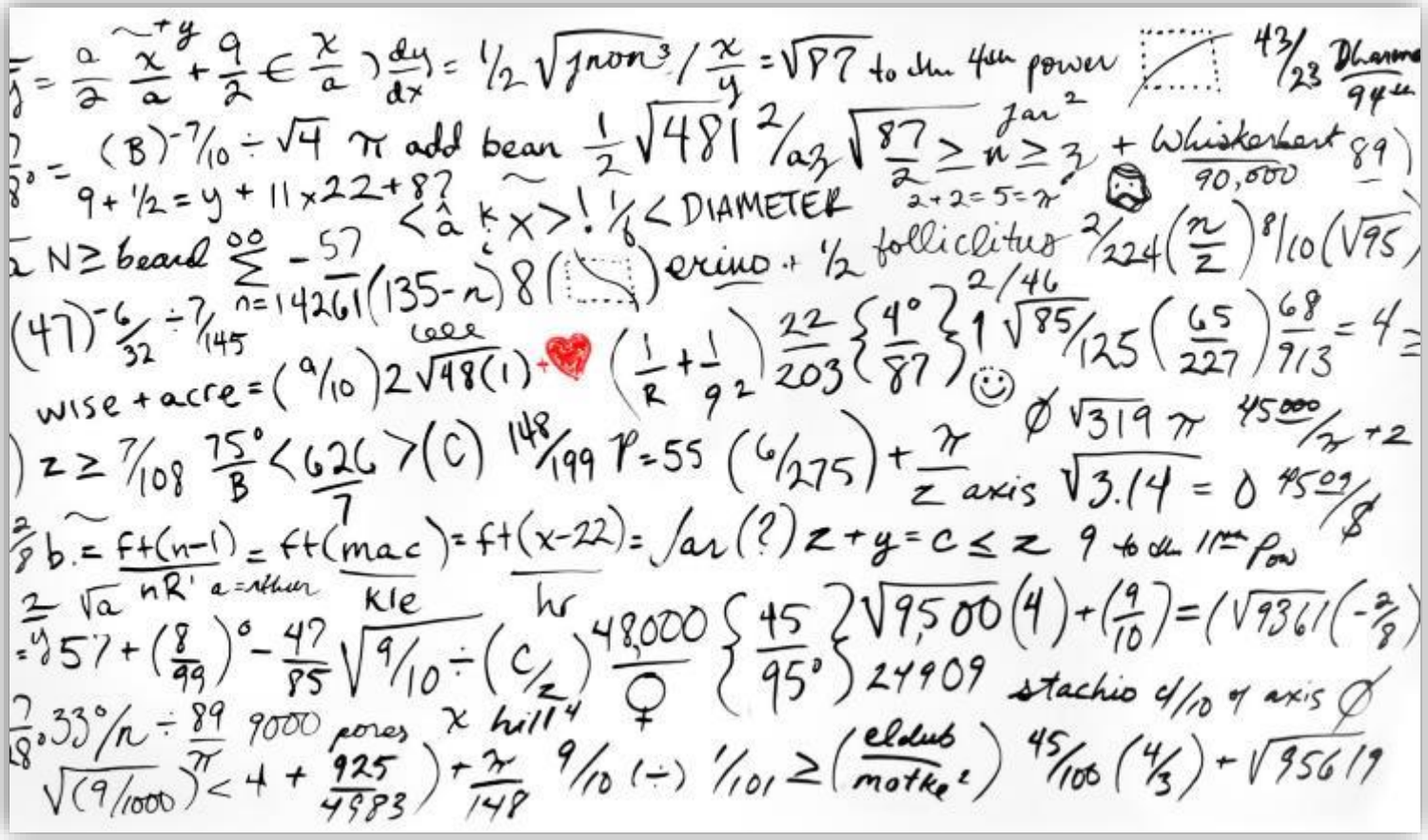
先想清楚

- 垃圾進，垃圾出(英語：Garbage in, garbage out ，GIGO)
- 先確認：
- 你要解決甚麼問題?
- 需要輸入甚麼?
- 需要輸出甚麼?
- 要用哪一種工具(語言)?



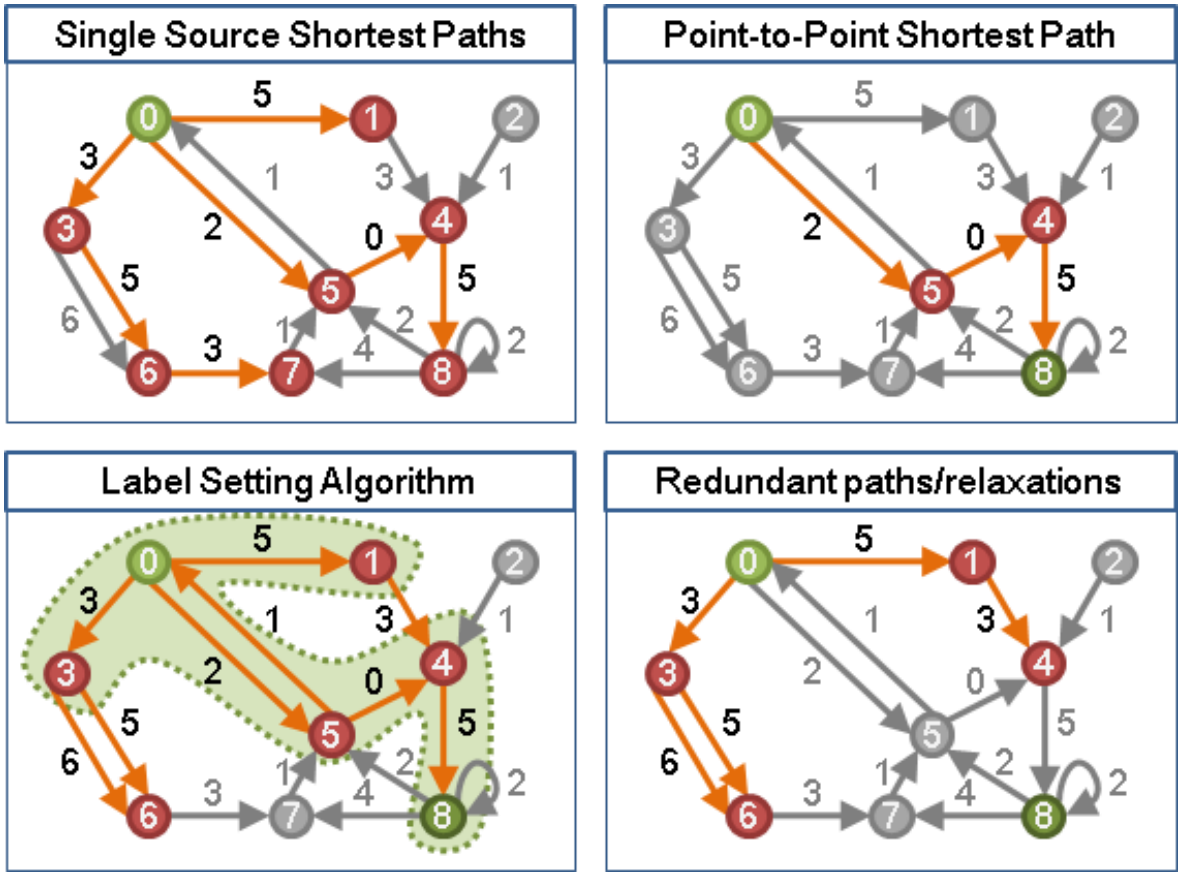
思考過程

- 分析你要解決的問題



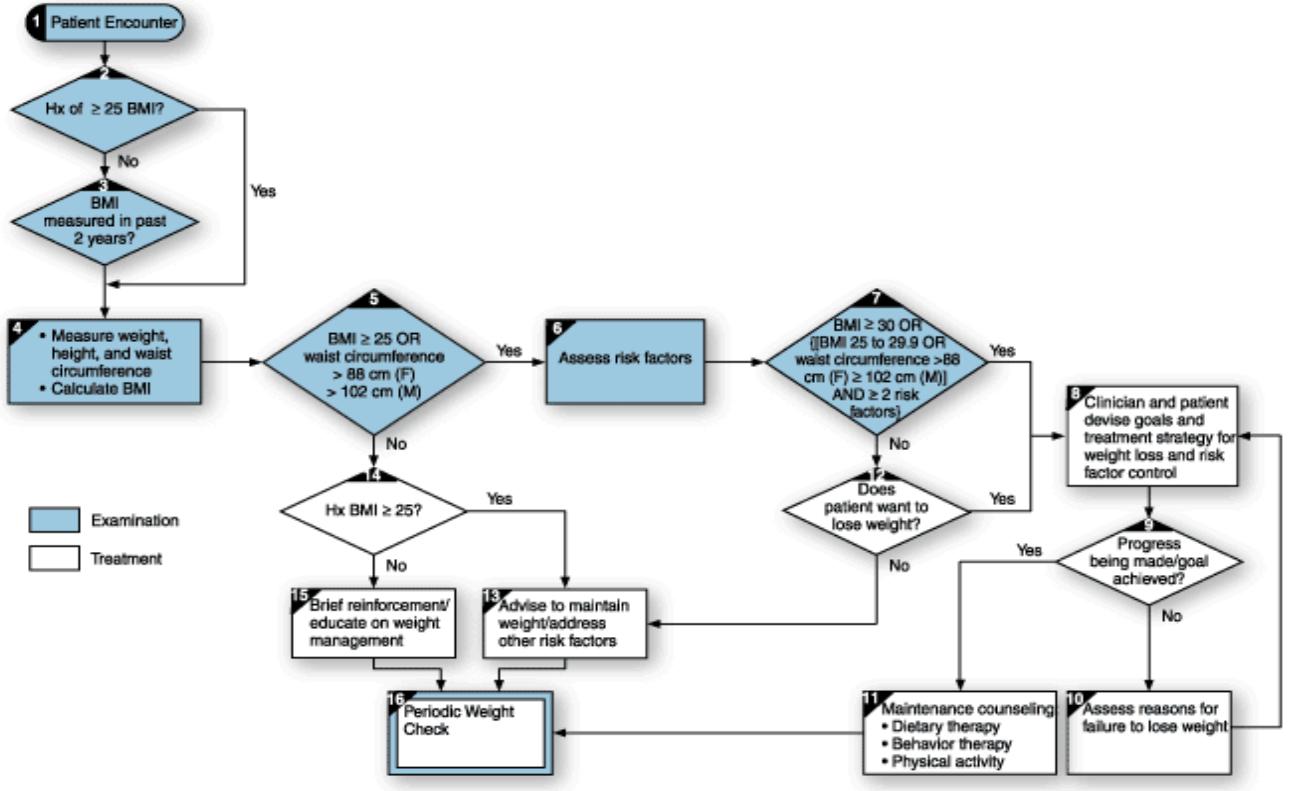
思考過程

- 抽象化和模型化



思考過程

- 規劃成可執行的步驟



* This algorithm applies only to the assessment for overweight and obesity and subsequent decisions based on that assessment. It does not include any initial overall assessment for cardiovascular risk factors or diseases that are indicated.

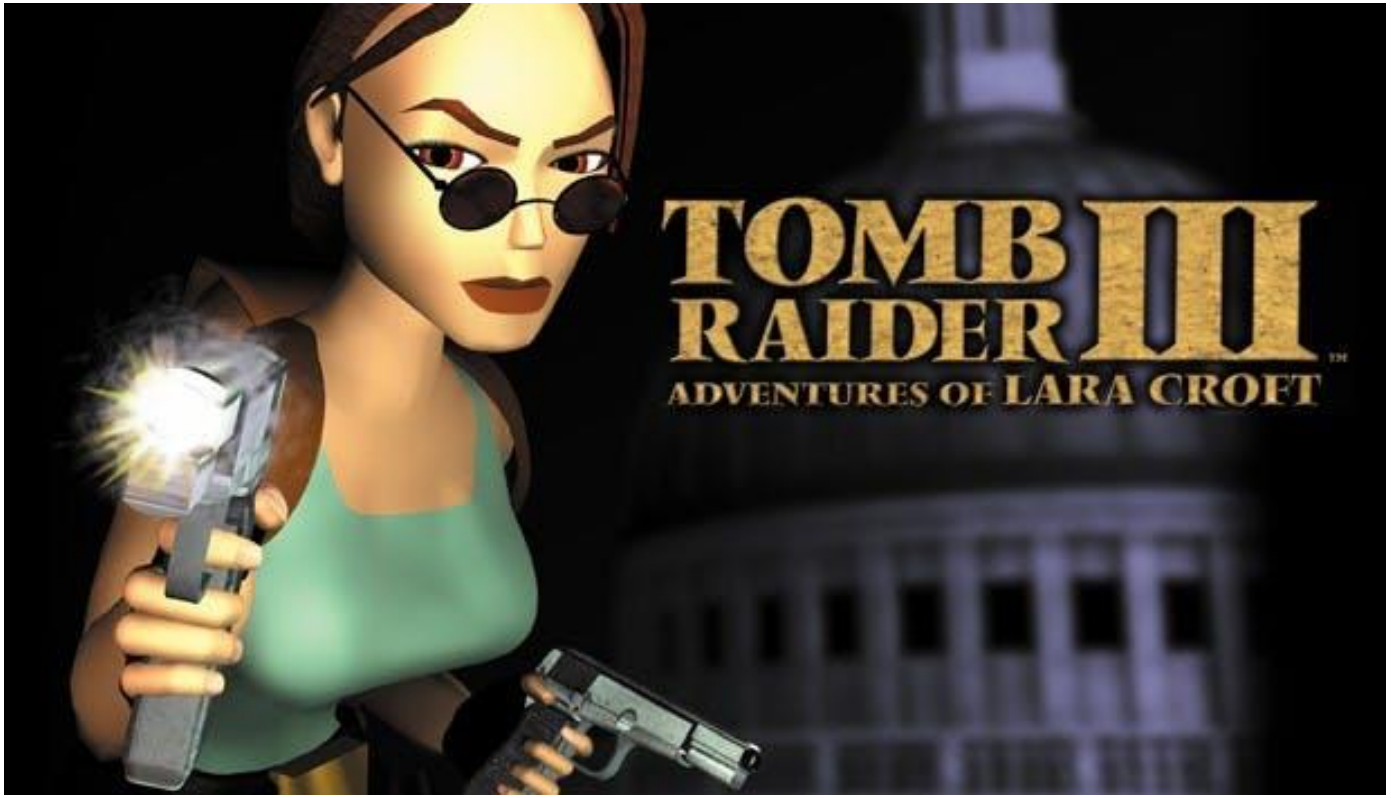
思考過程

- 寫程式(Coding)

```c
/* This line basically imports the "stdio" header file, part of
 * the standard library. It provides input and output functionality
 * to the program.
 */
#include <stdio.h>

/*
 * Function (method) declaration. This outputs "Hello, world" to
 * standard output when invoked.
 */
void sayHello() {
    // printf() in C outputs the specified text (with optional
    // formatting options) when invoked.
    printf("Hello, world!");
}

/*
 * This is a "main function". The compiled program will run the code
 * defined here.
 */
void main() {
    // Invoke the sayHello() function.
    sayHello();
}
```

思考過程

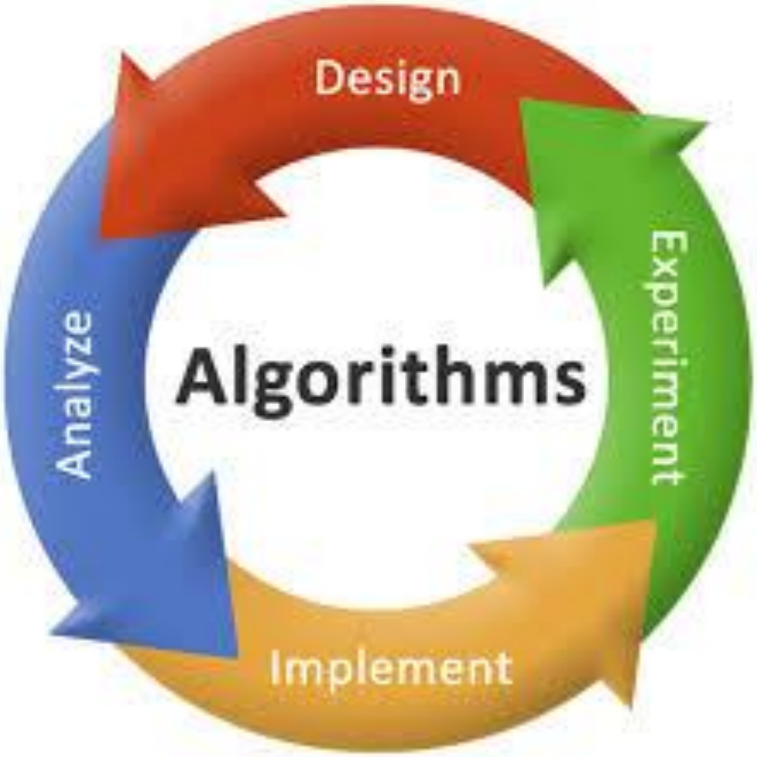
- 欣賞執行成果



- (所有的遊戲都是程式寫出來的~)

演算法(Algorithms)

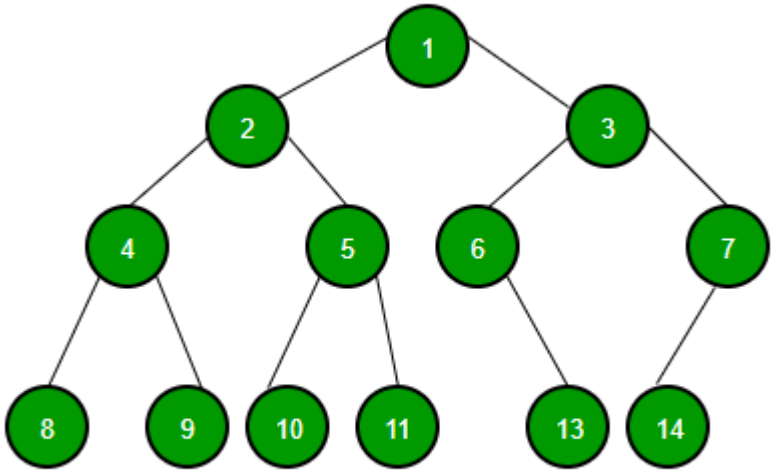
- 把問題抽象化和模型化，找出合理有效的解決步驟
- 就是解決問題的方法啦



演算法(Algorithms)

- 面對不同類型的問題，使用適當的演算法可以讓問題簡單很多。

- 例如：
  - 陣列(Array)
  - 鏈結串列(Link List)
  - 堆疊(Stack)
  - 佇列(Queue)
  - 樹(Tree)
  - 圖(Graph)

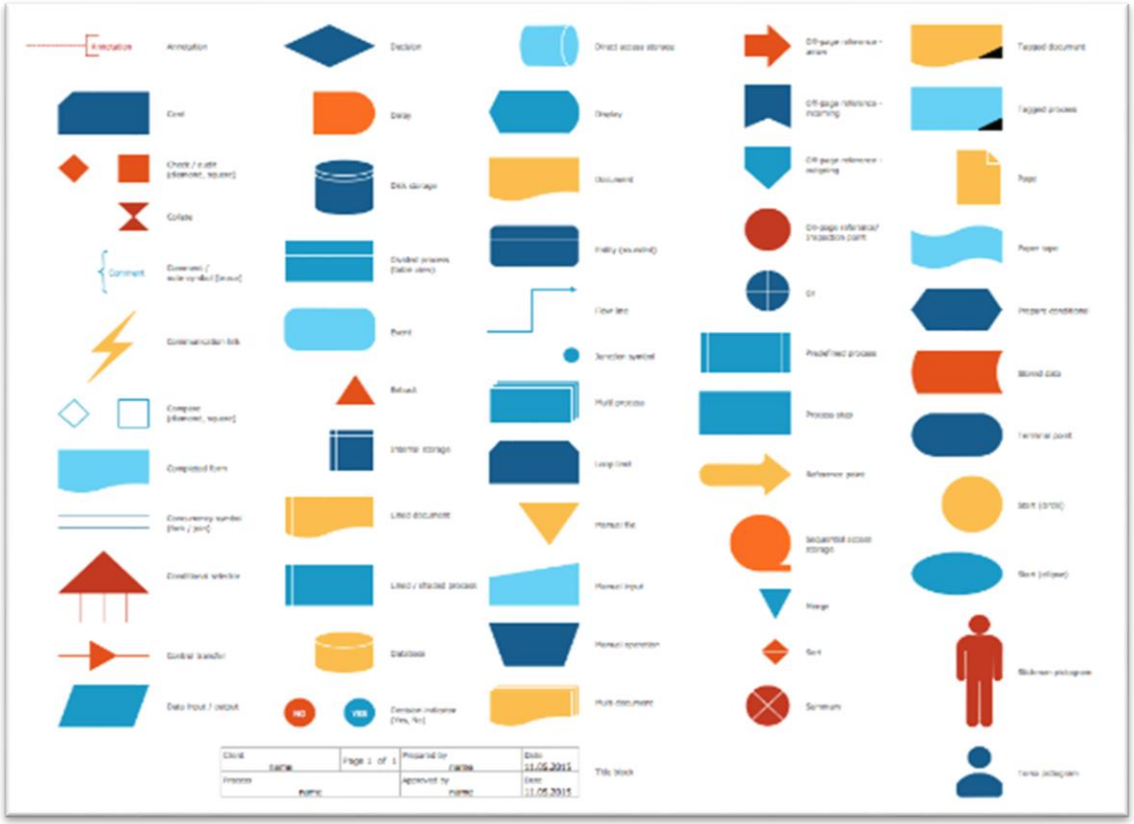


演算法(Algorithms)

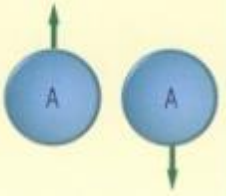
- 一個好的演算法要求：
  - **輸入**：有零個或一個以上輸入。
  - **輸出**：應有一個或以上輸出。
  - **明確性**：演算法的描述必須無歧義，以保證演算法的實際執行結果是精確地符合要求或期望，通常要求實際執行結果是確定的。
  - **有限性**：必須在有限個步驟內完成任務。
  - **有效性**：又稱可行性。能夠實現，。

演算法的工具

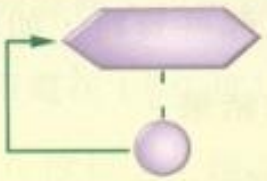
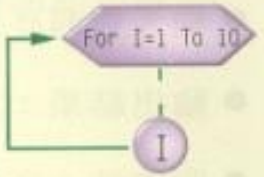
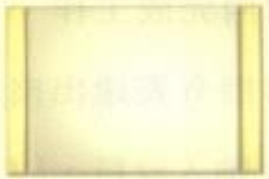
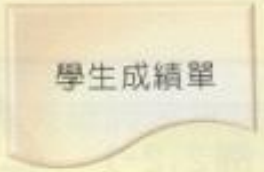
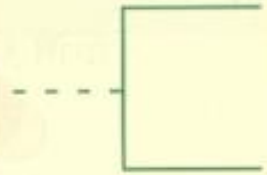
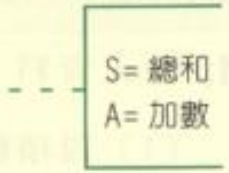
- 流程圖(Flow Chart)是個不錯的方式
- 用簡單的圖形來代表執行的步驟



常用流程圖

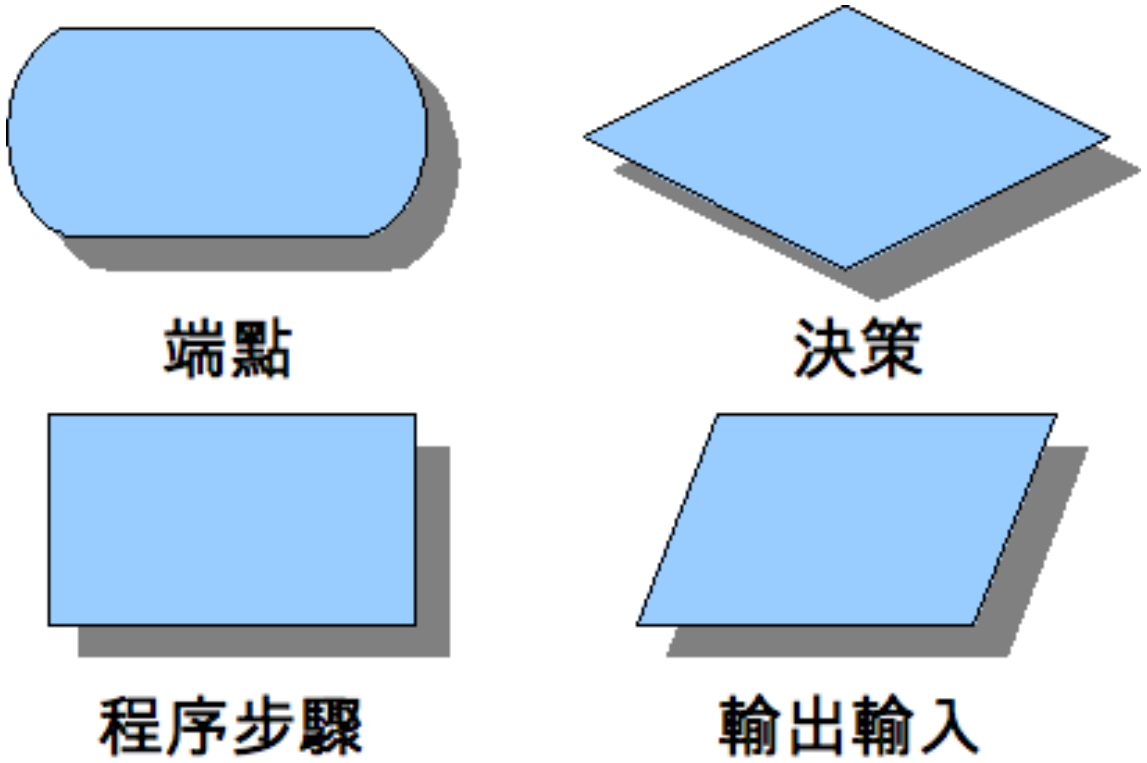
名稱	符號	意義	範例
1.起止符號		表示程式的開始或結束	
2.流程符號		表示流程進行的方向	
3.輸入/輸出符號		表示資料之輸入或結果的輸出	
4.處理符號		表示執行或處理某些工作	
5.決策判斷符號		表示對某一個條件做判斷	
6.連接符號		用於：1.轉接到另一頁 2.避免流線交叉 3.避免流線太長	

常用流程圖

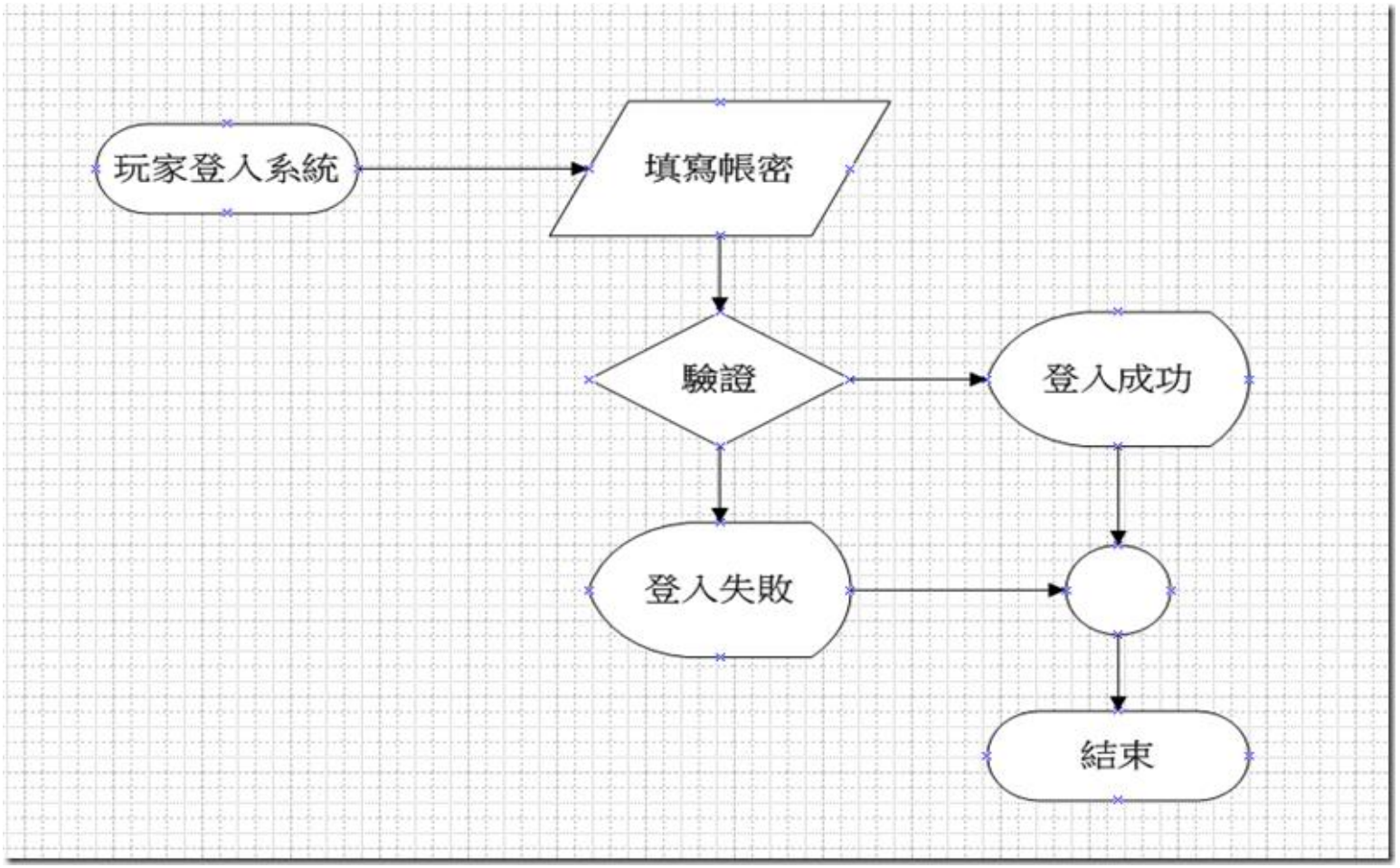
名　稱	符　號	意　義	範　例
7.迴圈符號		表示程式迴圈控制變數初值及終值的假設	
8.副程式符號		表示一群程式步驟或流程，用以說明副程式或其他流程的組合	
9.報表符號		表示以列表機印出報表文件	
10.註解符號		表示對某一流程加以註解	

常用流程圖

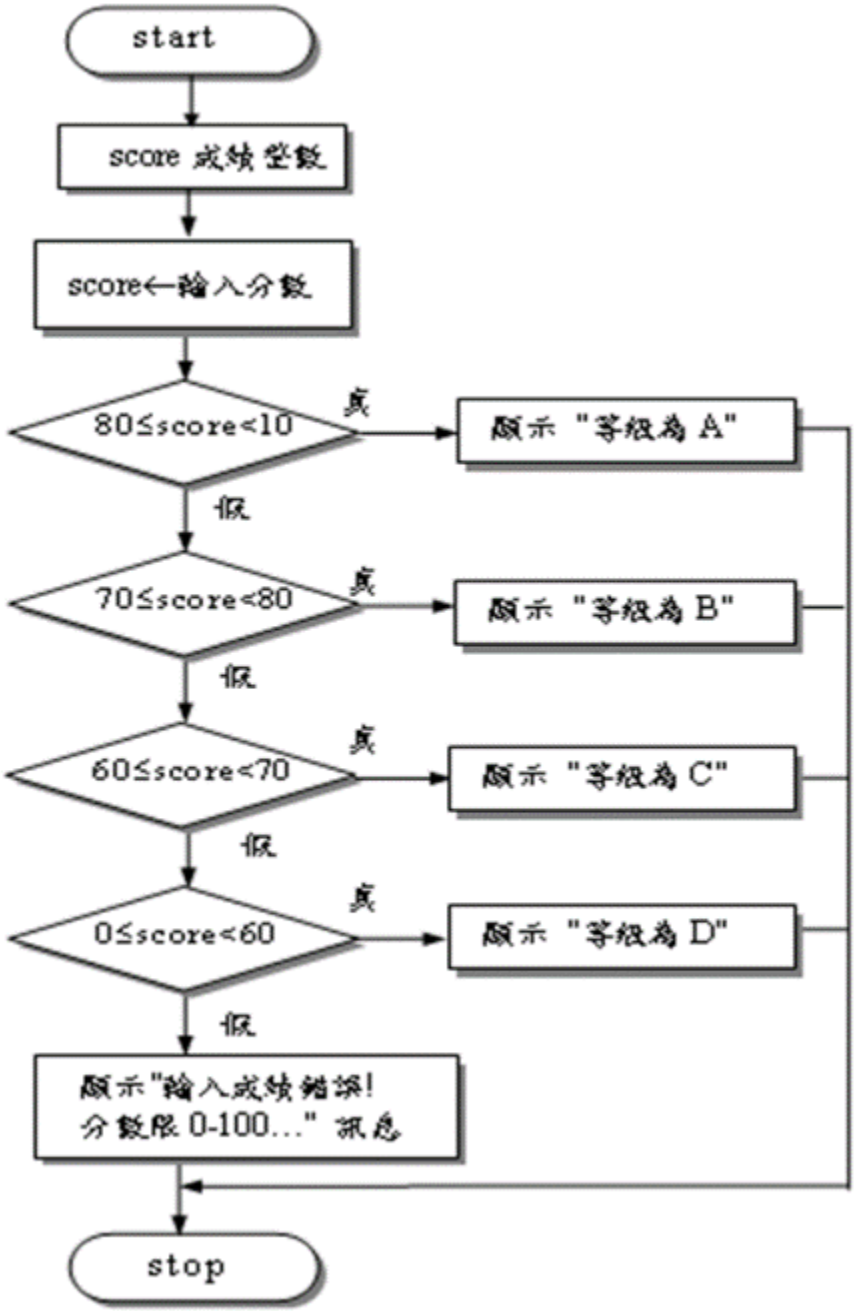
- 對於簡單的問題，其實記住這四個也就夠了



流程圖範例

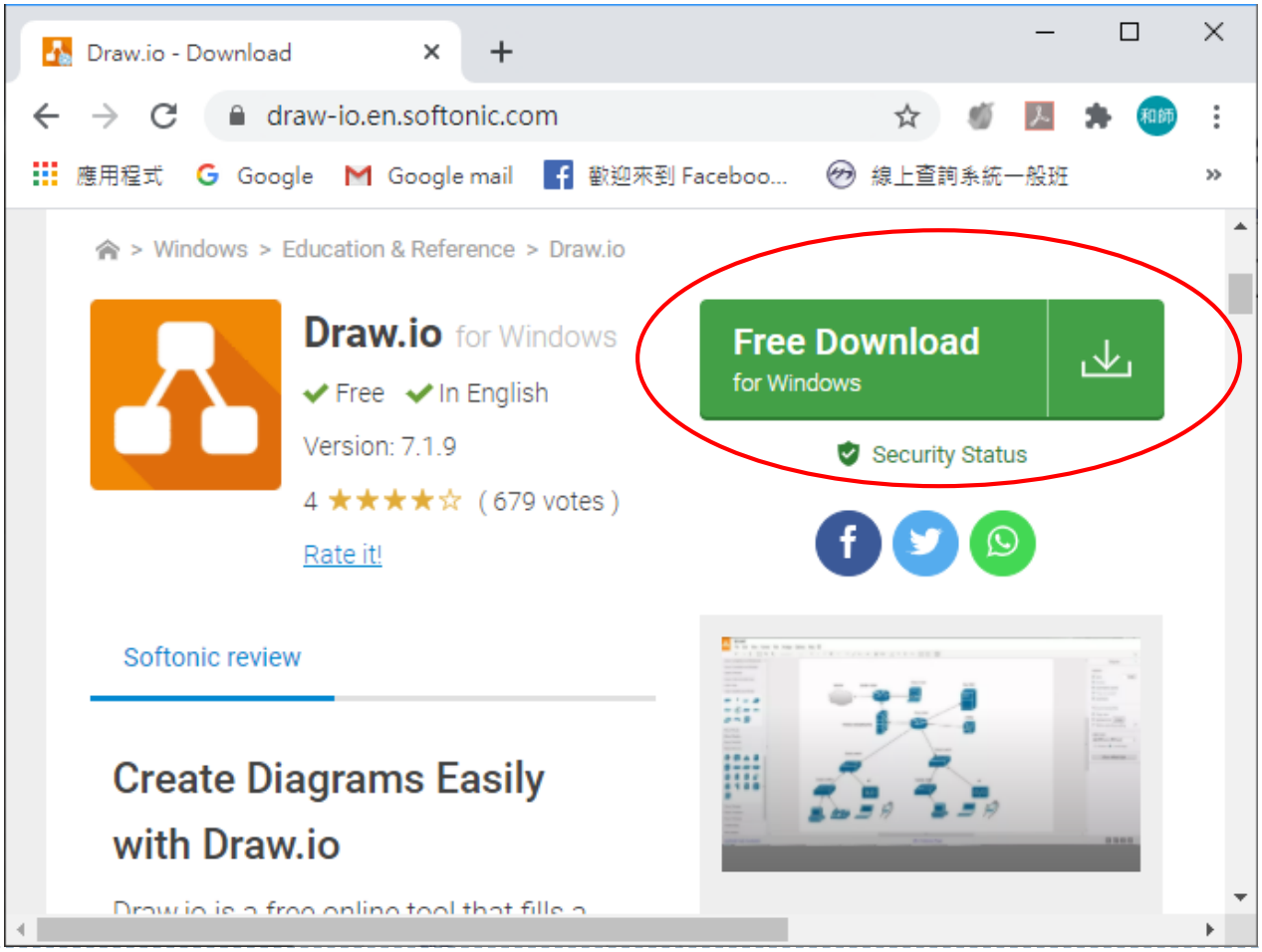


流程圖範例


start
score 成績 宣數
score←輸入分數
80≤score<10
真
顯示 "等級為 A"
假
70≤score<80
真
顯示 "等級為 B"
假
60≤score<70
真
顯示 "等級為 C"
假
0≤score<60
真
顯示 "等級為 D"
假
顯示"輸入成績錯誤!
分數限 0-100..." 訊息
stop

畫流程圖的工具

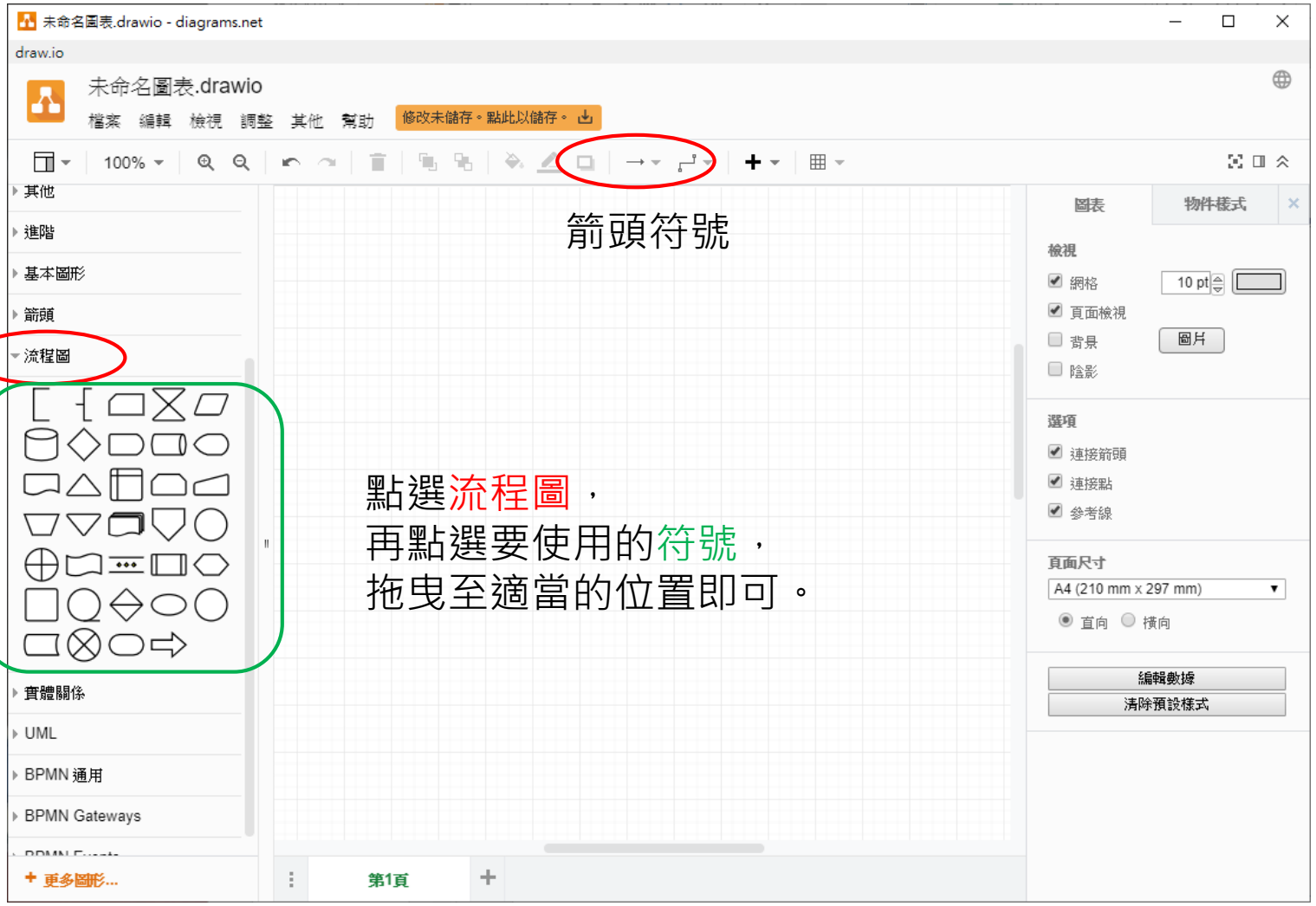
- Draw.io是個免費的畫流程圖的工具軟體



畫流程圖的工具

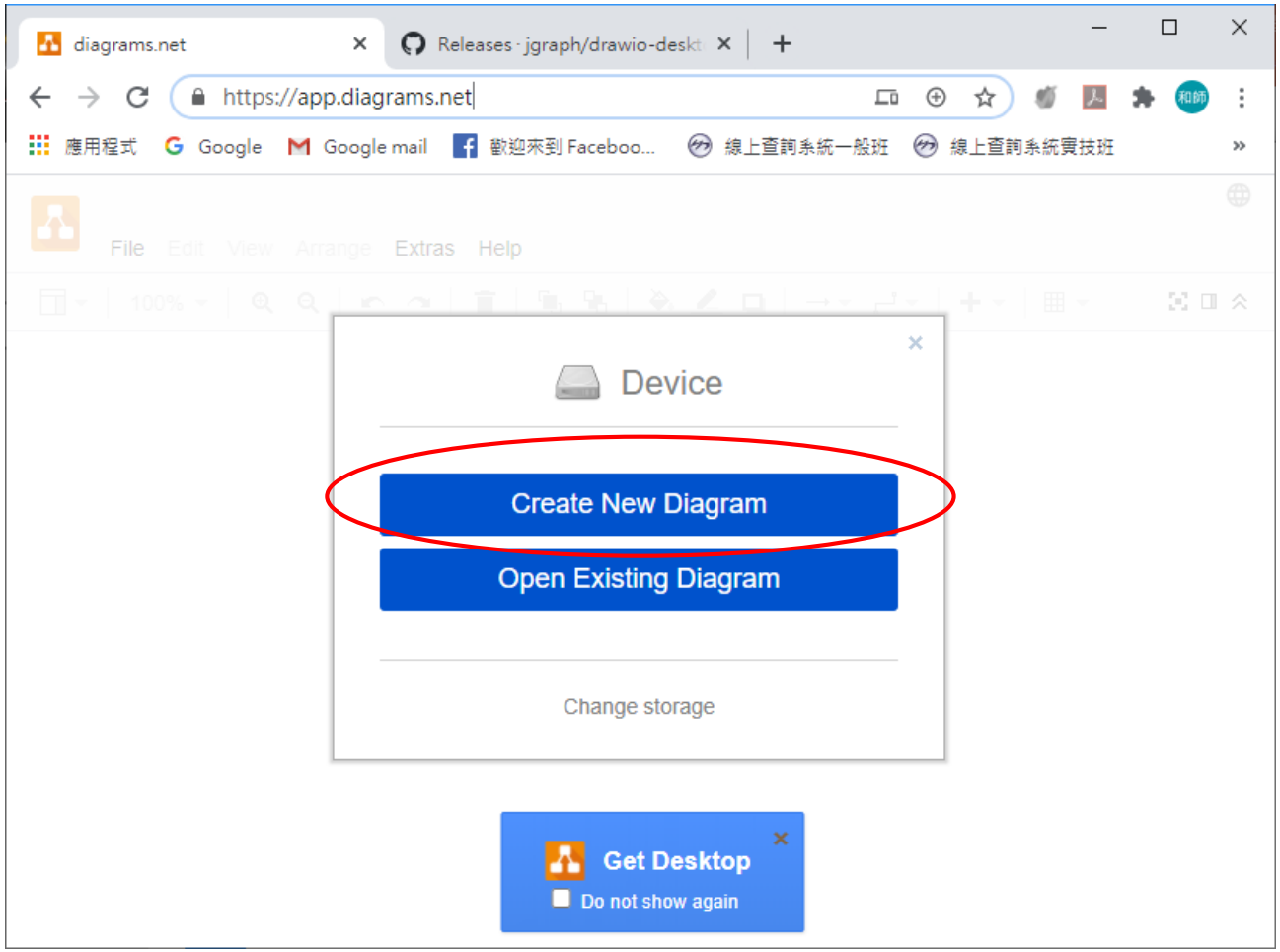
- 安裝後開啟程式選「流程圖」→「新增」

畫流程圖的工具


未命名圖表.drawio - diagrams.net
draw.io
未命名圖表.drawio
檔案 編輯 檢視 調整 其他 幫助
修改未儲存。點此以儲存。
100%
其他
進階
基本圖形
箭頭
流程圖
實體關係
UML
BPMN 通用
BPMN Gateways
更多圖形...
第1頁
箭頭符號
點選流程圖，
再點選要使用的符號，
拖曳至適當的位置即可。
圖表
物件樣式
檢視
網格
10 pt
頁面檢視
背景
圖片
陰影
選項
連接箭頭
連接點
參考線
頁面尺寸
A4 (210 mm x 297 mm)
直向
橫向
編輯數據
清除預設樣式

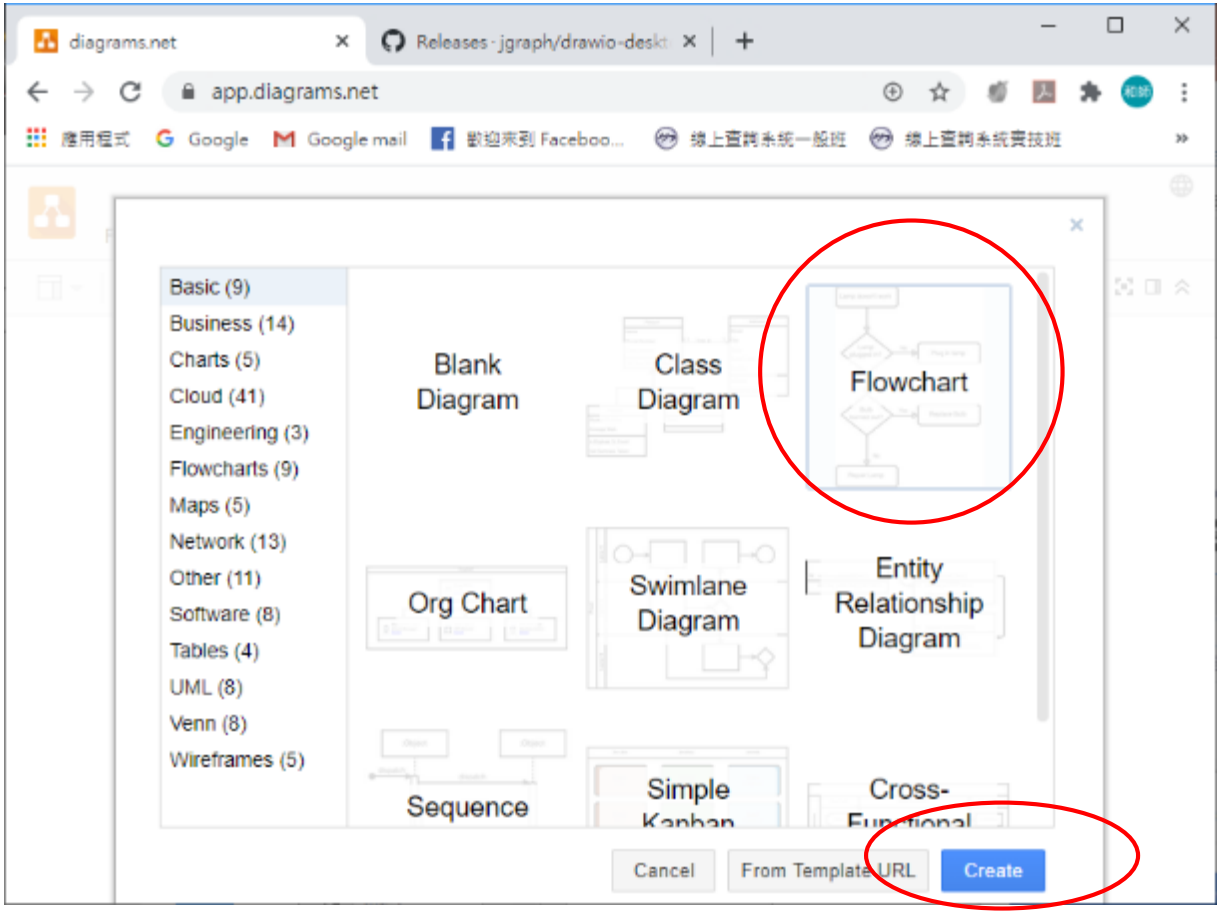
畫流程圖的工具

- 也可用線上版，直接在瀏覽器裡使用



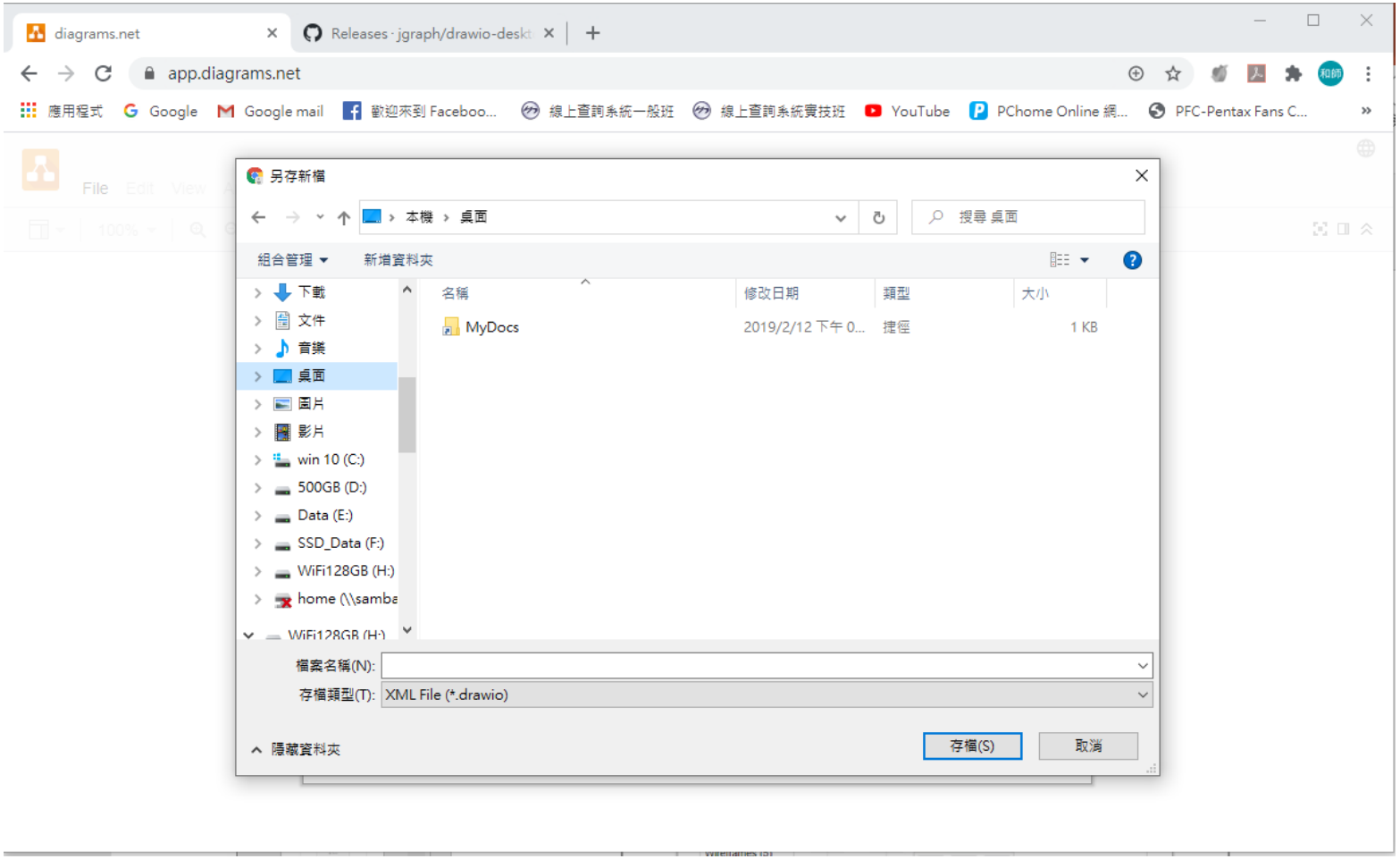
畫流程圖的工具

- 點選Flowchart → Create



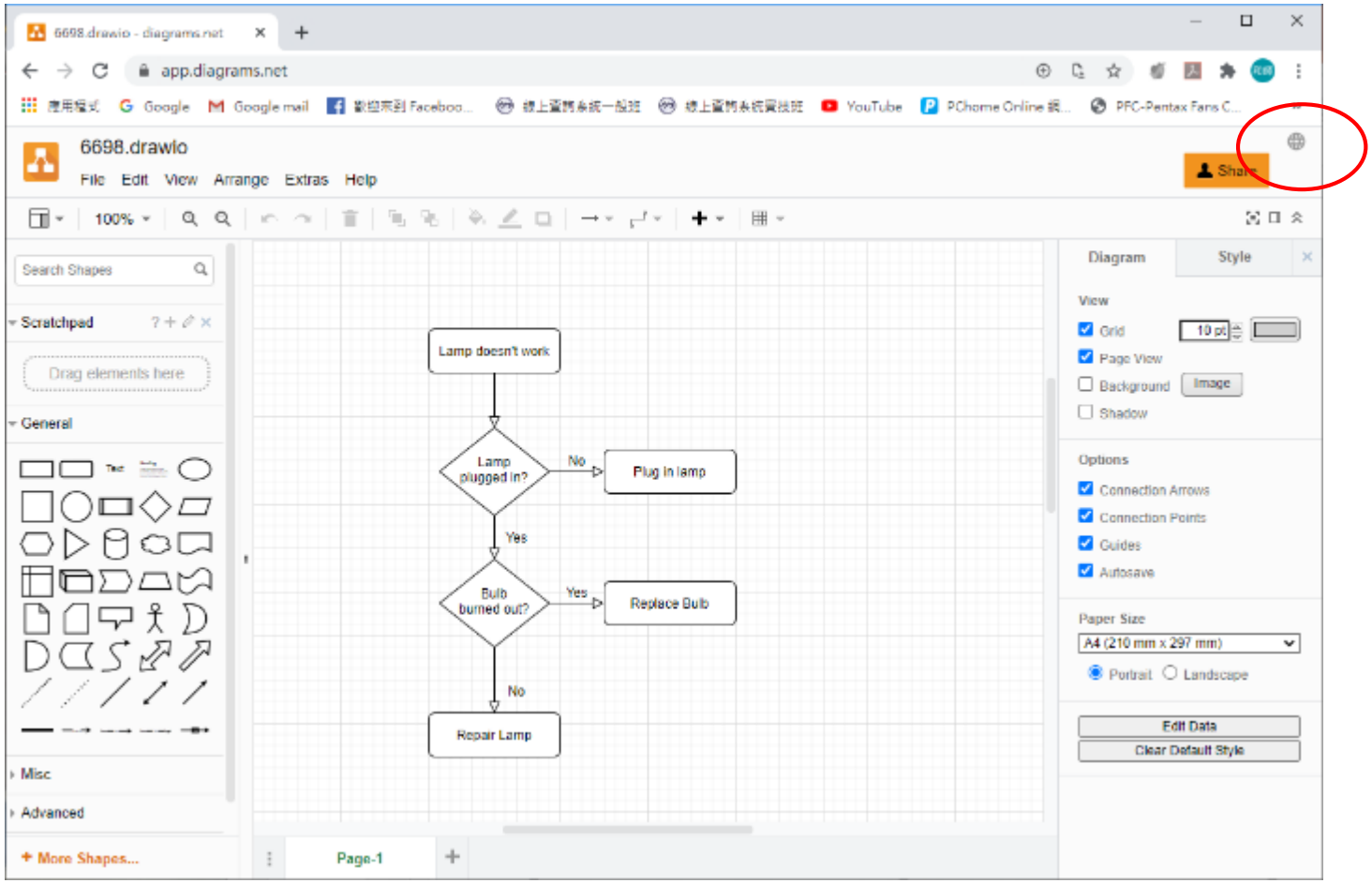
畫流程圖的工具

- 會先問你要存在哪兒



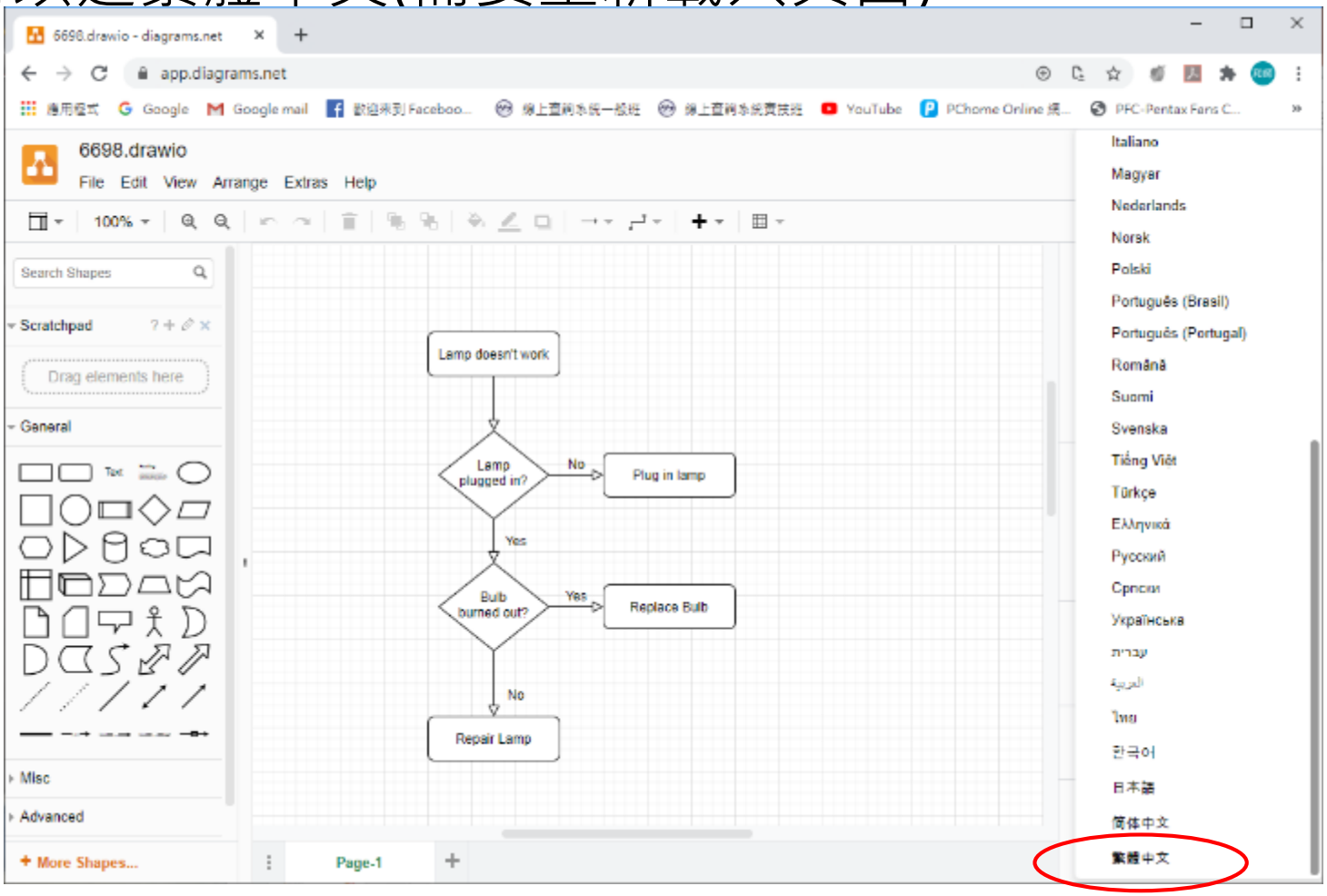
畫流程圖的工具

- 開始使用~ (右上方地球符號可以更換介面語言)



畫流程圖的工具

- 可以選繁體中文(需要重新載入頁面)



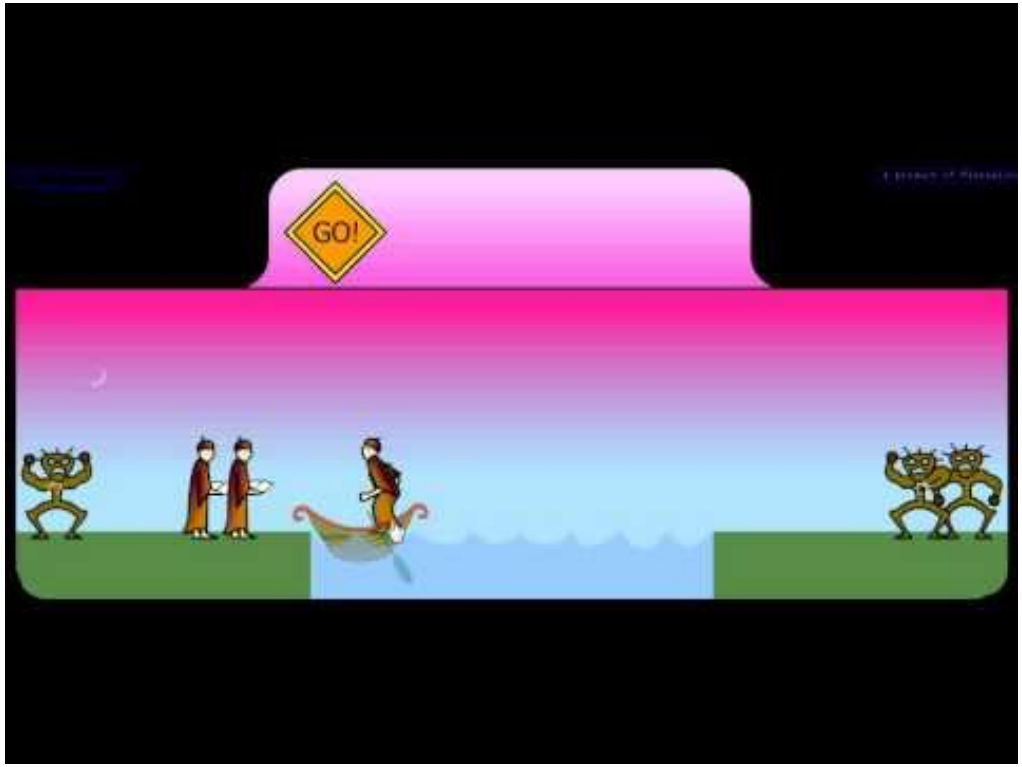
想想看

- 颱風天到了，請寫一個防颱準備的流程圖

- (你會不會把門窗封住了才想到要去X聯買泡麵?)

這些益智遊戲可以多玩玩

- 過河問題：3個人和三個鬼要過河，可是只一隻小船，一次只能載兩個，且在河兩邊鬼的數量不能比人多，否則就會把人給吃了，怎麼樣才能安全的過河呢？

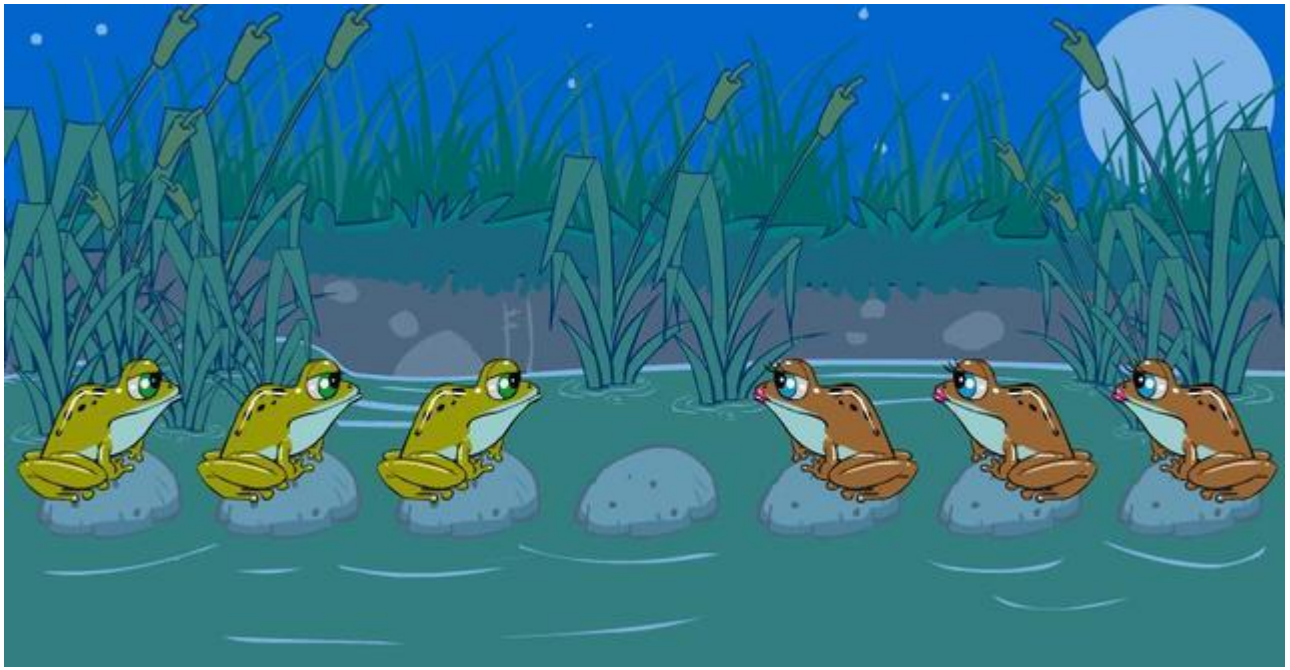


這些益智遊戲可以多玩玩

- 比較複雜的過河遊戲：一個男人帶著兩個小男孩，一個女人帶著兩個小女孩，一個警察帶著一個小偷，要過河，可是只有一條小船，一次只能載兩個人，孩子和小偷不會划船。如果男人不在則女人會教訓小男孩，如果女人不在男人則會教訓小女孩，如果警察不在小偷則會傷害這些人。怎麼才能安全過河呢？

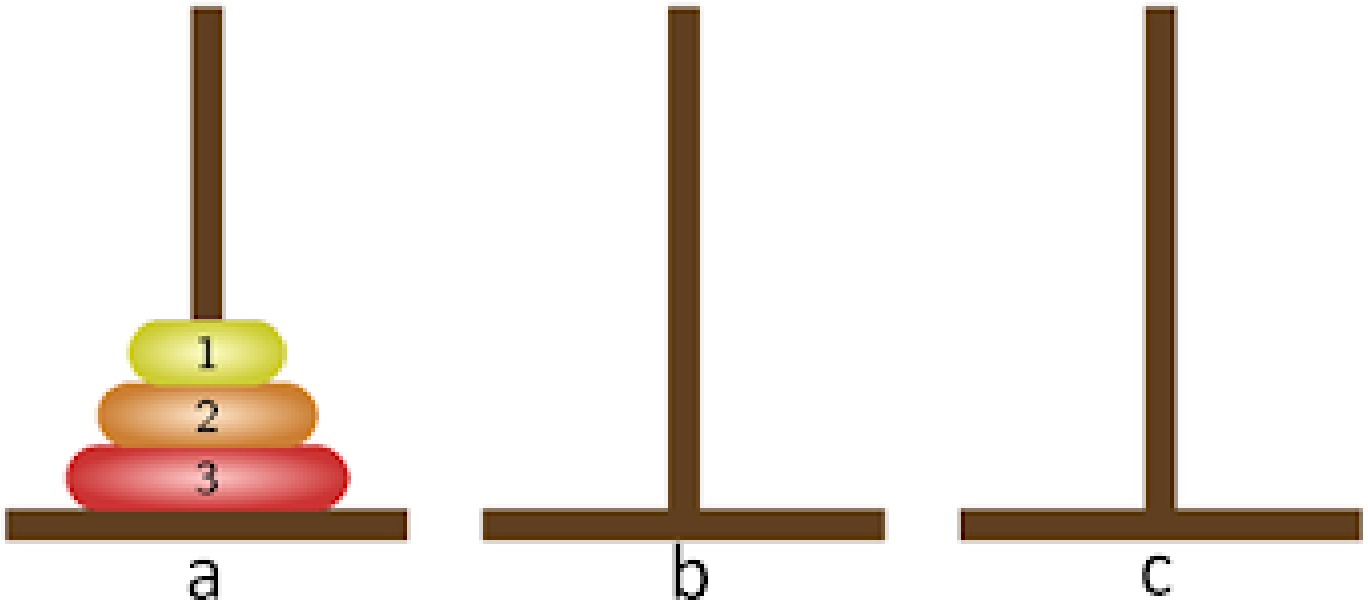
這些益智遊戲可以多玩玩

- 有六隻青蛙要過河，河的左邊有三隻，河的右邊有三隻，中間一塊空白石頭。它們互不相讓。怎麼樣才能過河呢



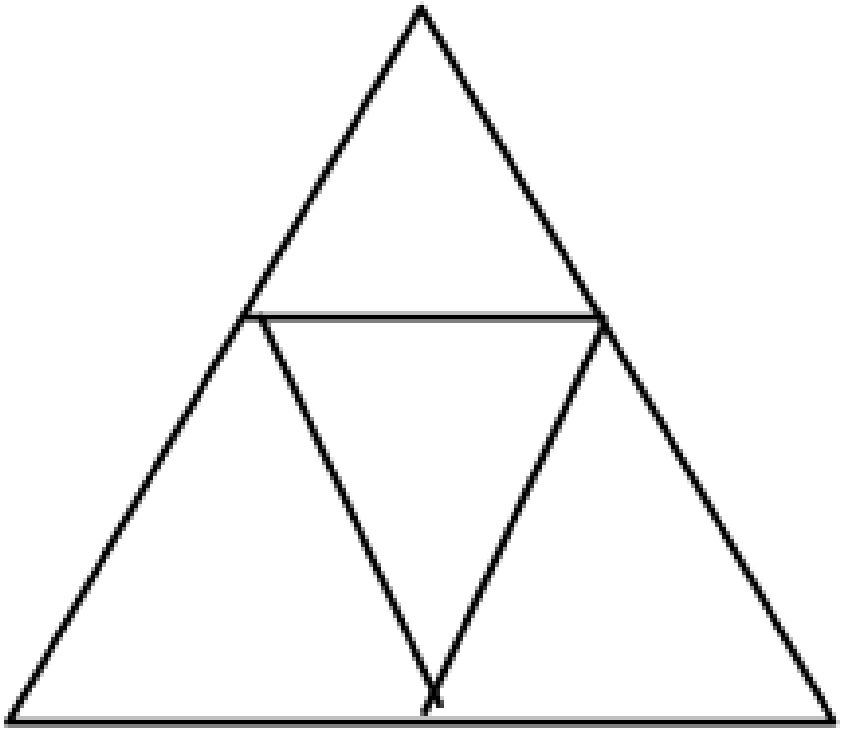
這些益智遊戲可以多玩玩

- 河內塔



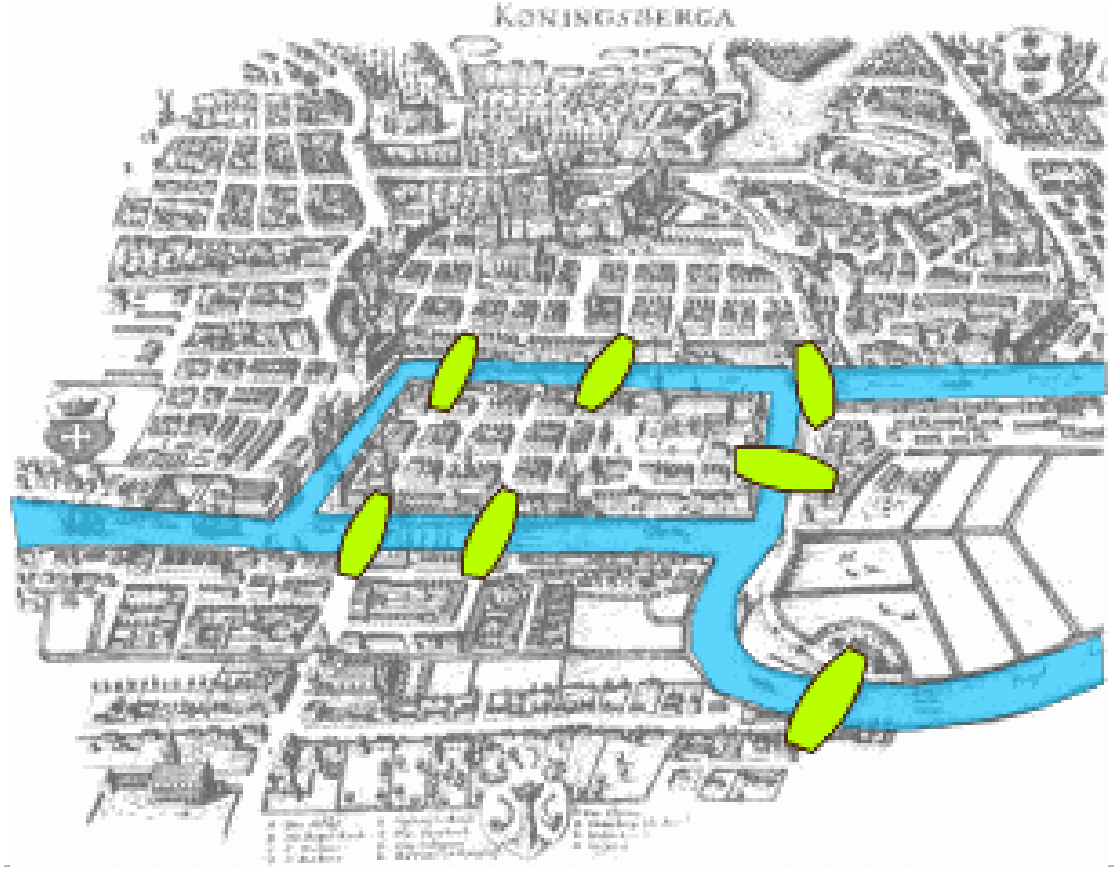
這些益智遊戲可以多玩玩

- 一筆畫問題



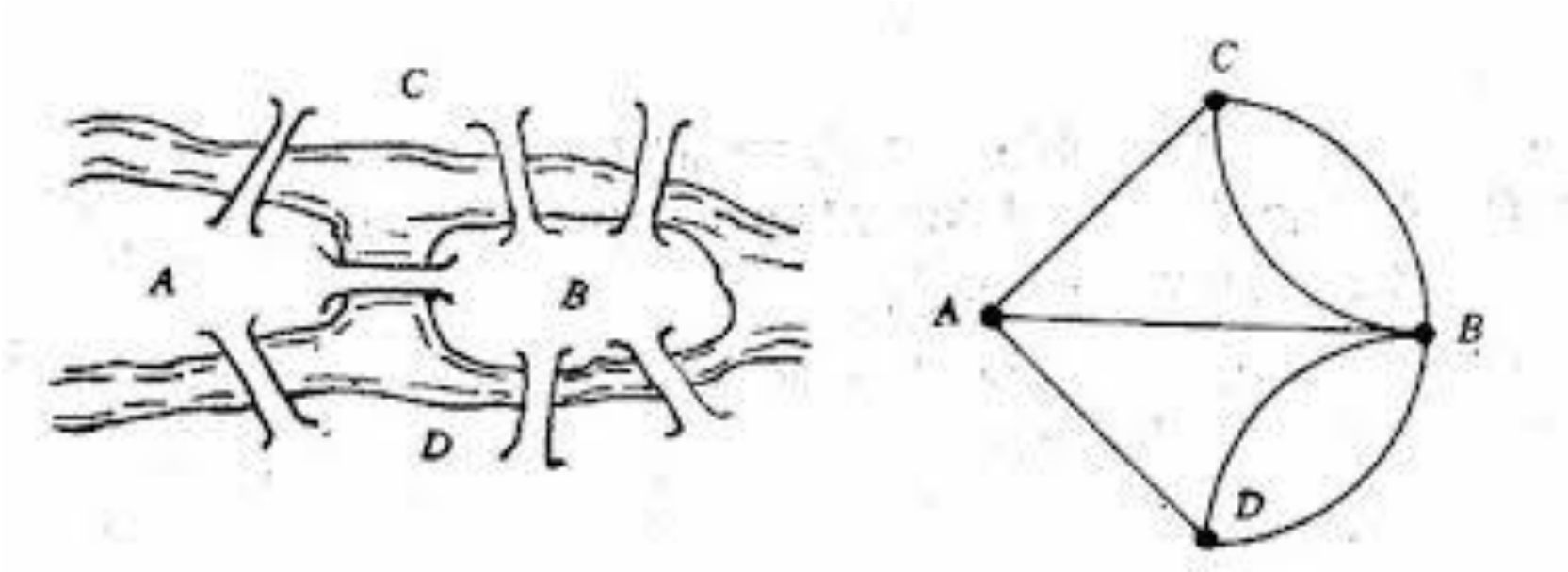
這些益智遊戲可以多玩玩

- 柯尼斯堡的七橋問題：要如何才能從某一塊土地開始，將每座橋恰好經過一次，然後回到原地?



這些益智遊戲可以多玩玩

- 柯尼斯堡的七橋問題：要如何才能從某一塊土地開始，將每座橋恰好經過一次，然後回到原地?



練習一下(畫出流程圖)

- 有兩串香蕉，比較它們的重量
  然後輸出哪一個比較重?

- 輸入三個數值a，b，c，比較它們
  的大小，輸出最大的那個

練習一下(畫出流程圖)

- 請寫一個流程圖，狀況如下：
- 今天晚餐要約小美吃飯，如果身上有$1000就吃大餐，如果只有$500就吃牛肉麵，如果只有$300就吃夜市，都不夠的話取消約會(慘...)

多練習就會了

- 流程圖是你解決問題的想法及過程，沒有標準答案，只要能解決問題就可以
- 不過**效率**就有關係了，練習讓你的思考更清晰，步驟更簡潔

Visual Basic(簡稱VB)

- 我們使用微軟公司發展的Visual Basic(簡稱VB)這個語言來練習
- 原因:
  - 1.簡單易用
  - 2.升學考試要考
  - 3.比爾蓋茲的最愛

ＶＢ版本

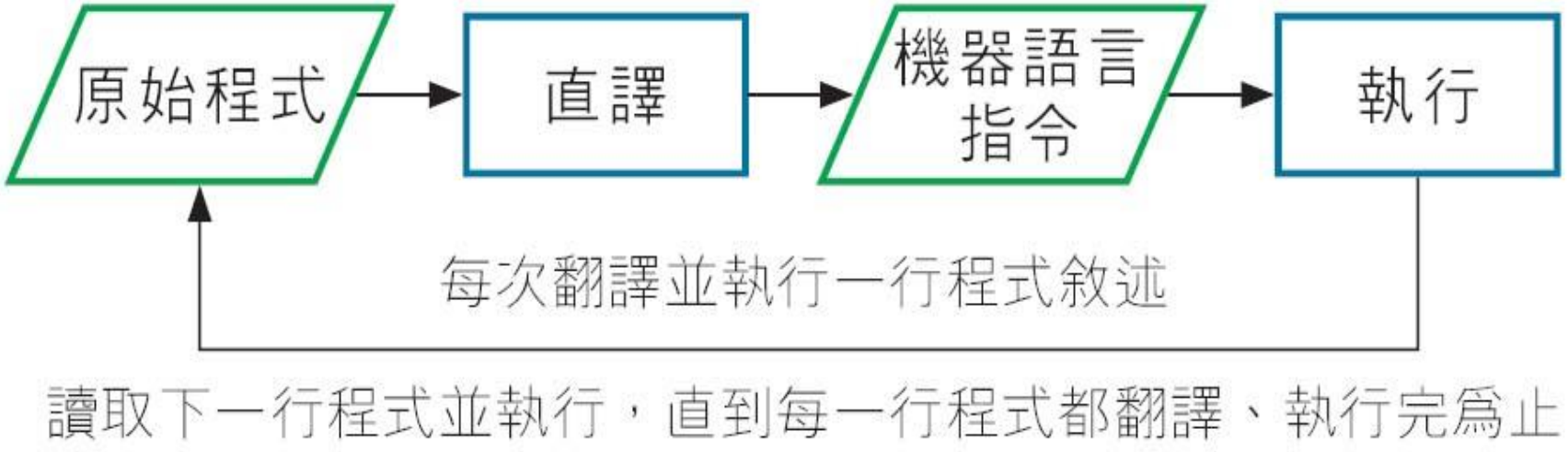
- 現在已整合到微軟的Visual Studio開發環境裡，不再單獨發行了
- 整套Visual Studio都是免費的(社群版)
- 它包含VB、C++、C#、F#、Python...等語言支援
- 目前最新是2019版
- 下載網址: https://www.visualstudio.com/zh-hant/?rr=https%3A%2F%2Fwww.google.com.tw%2F

程式如何執行

- 但是電腦並不認得高階語言，所以必須轉換成0101....才行

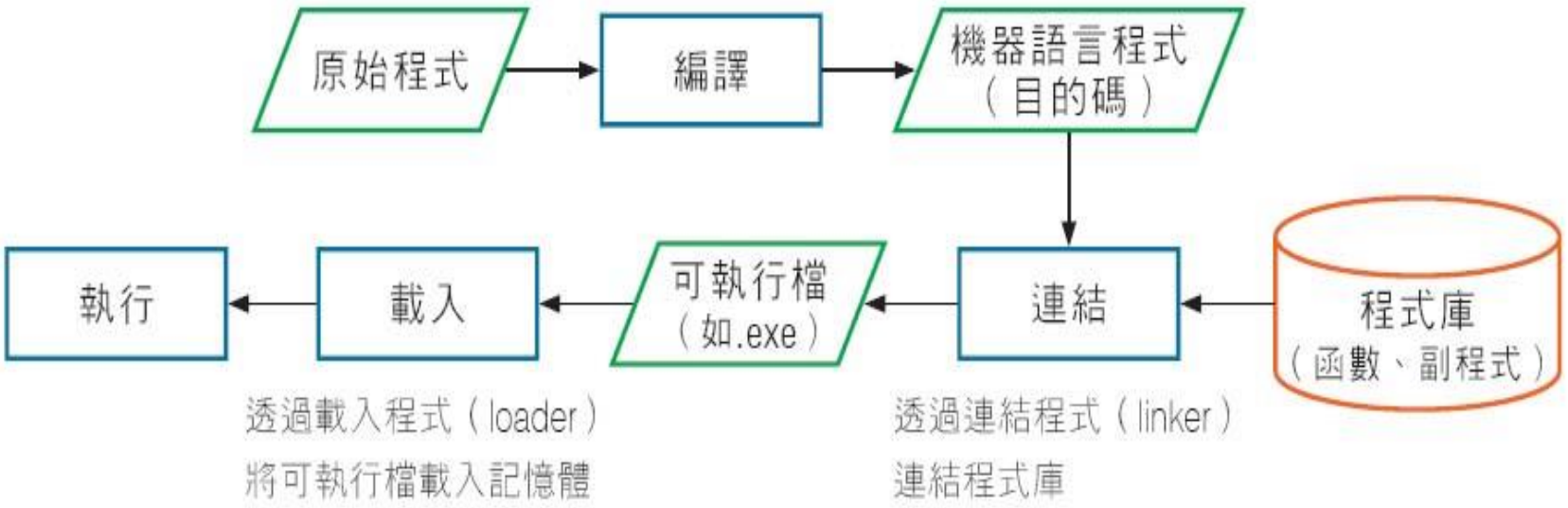
轉換我們寫的程式到可執行檔

- 直譯式(Interpreter)



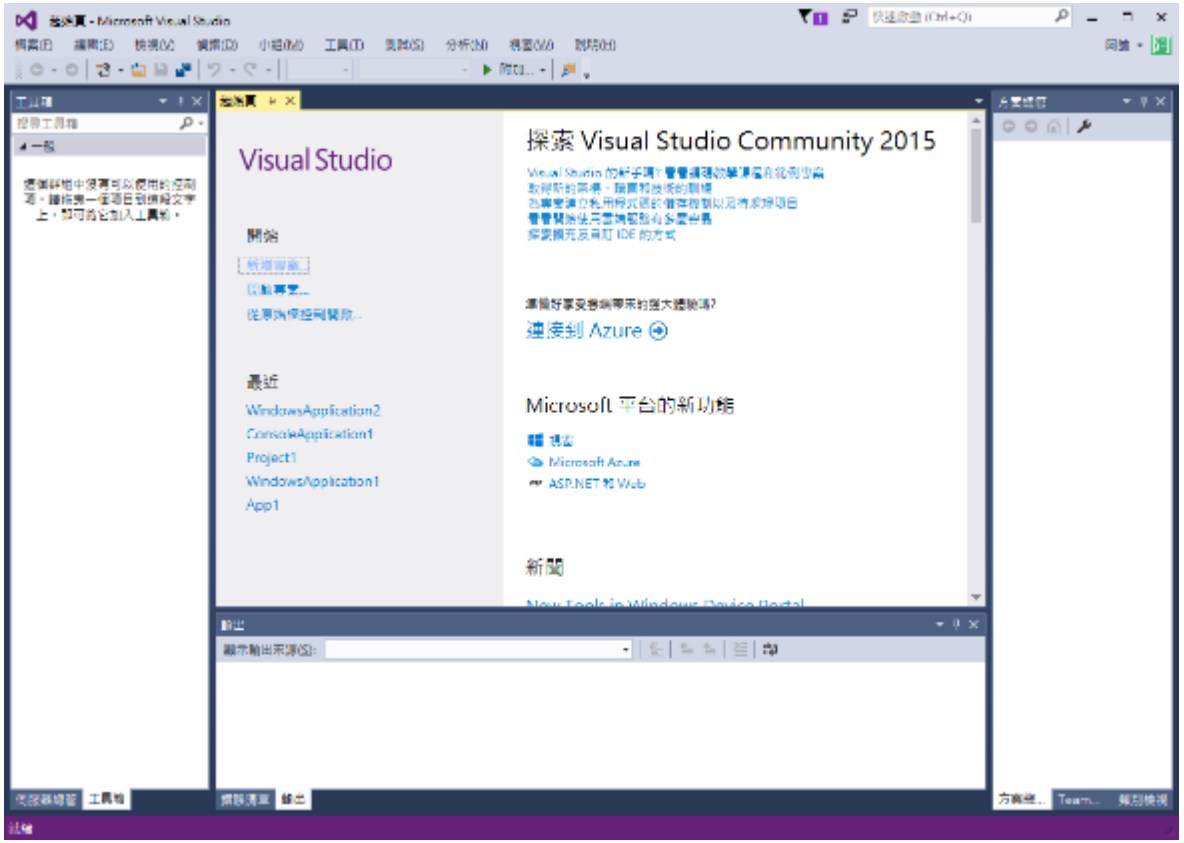
轉換我們寫的程式到可執行檔

- 編譯式(Compiler)



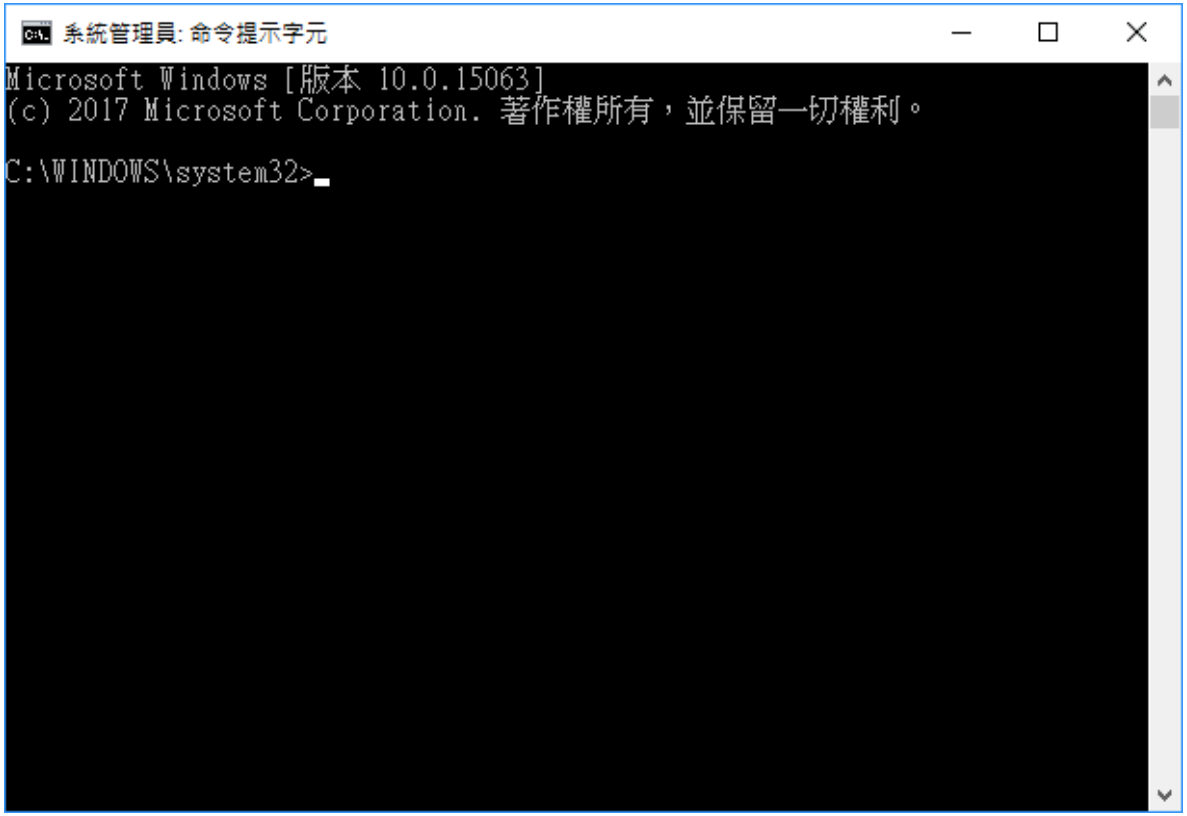
轉換我們寫的程式到可執行檔

- 不過在VB環境下所有編譯、連結的工作她都一鍵包辦了，不用擔心

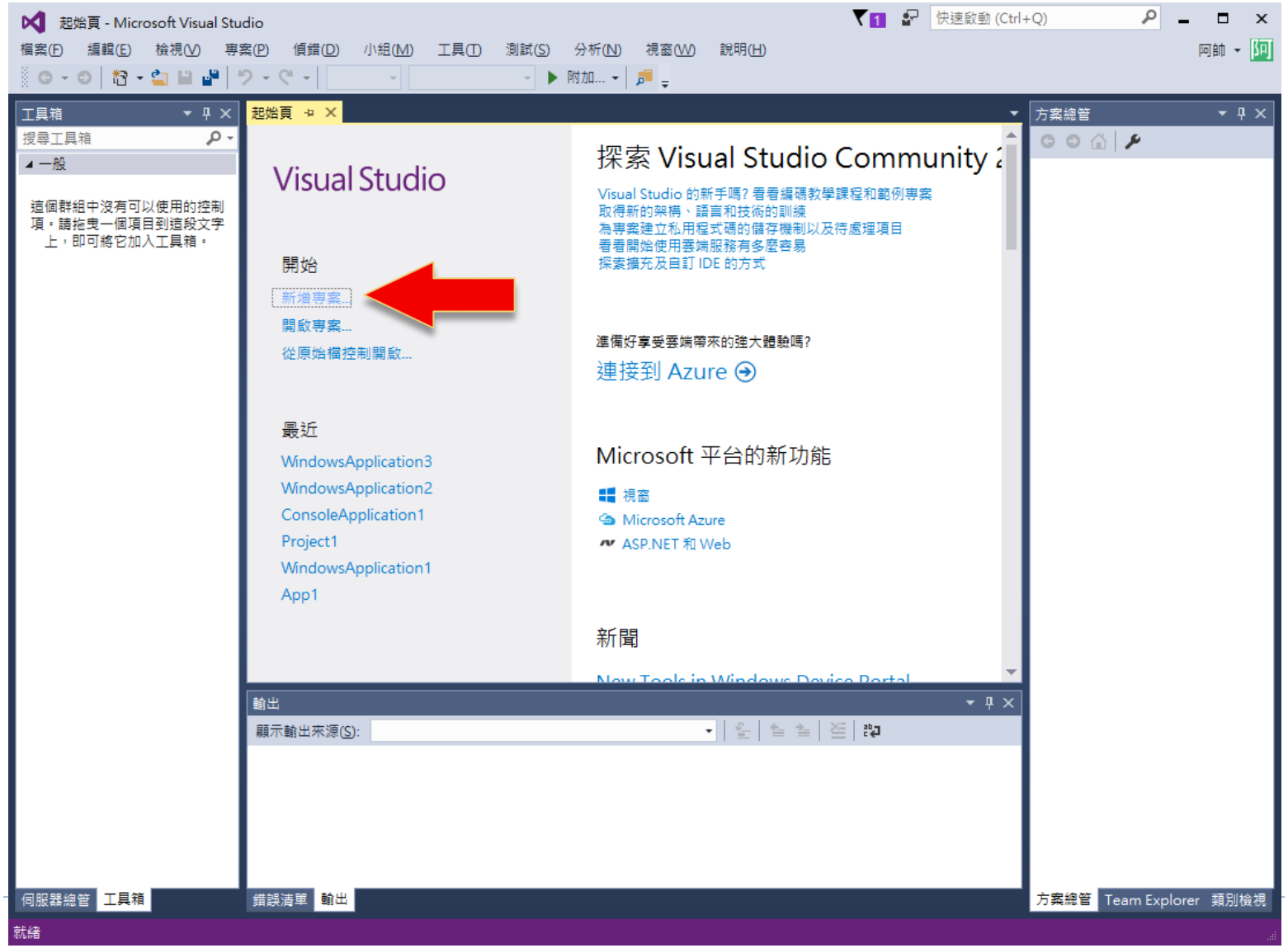


VB開發環境介紹

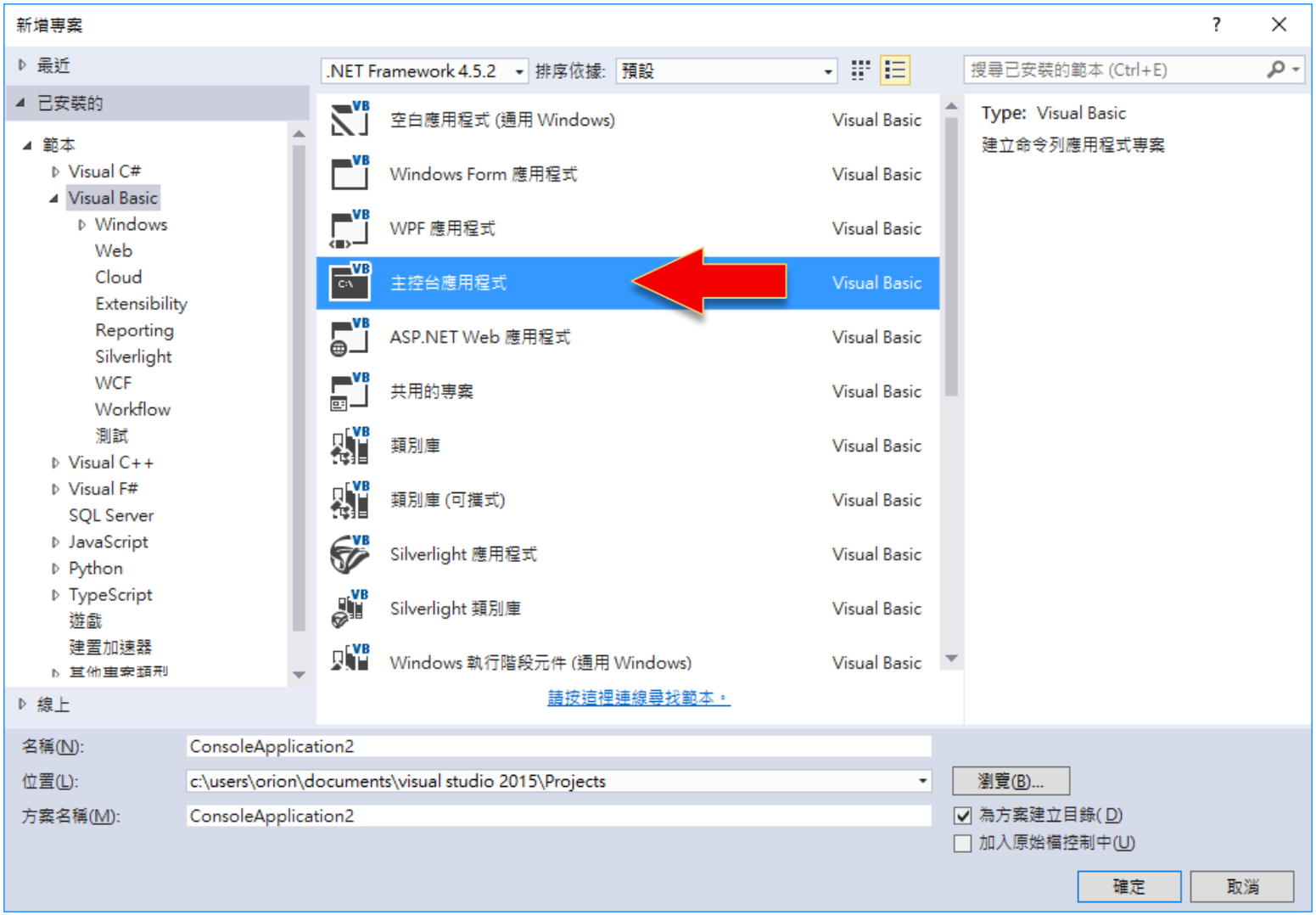
- 命令列模式(Console Mode)
- 簡單的指令練習或測試很好用



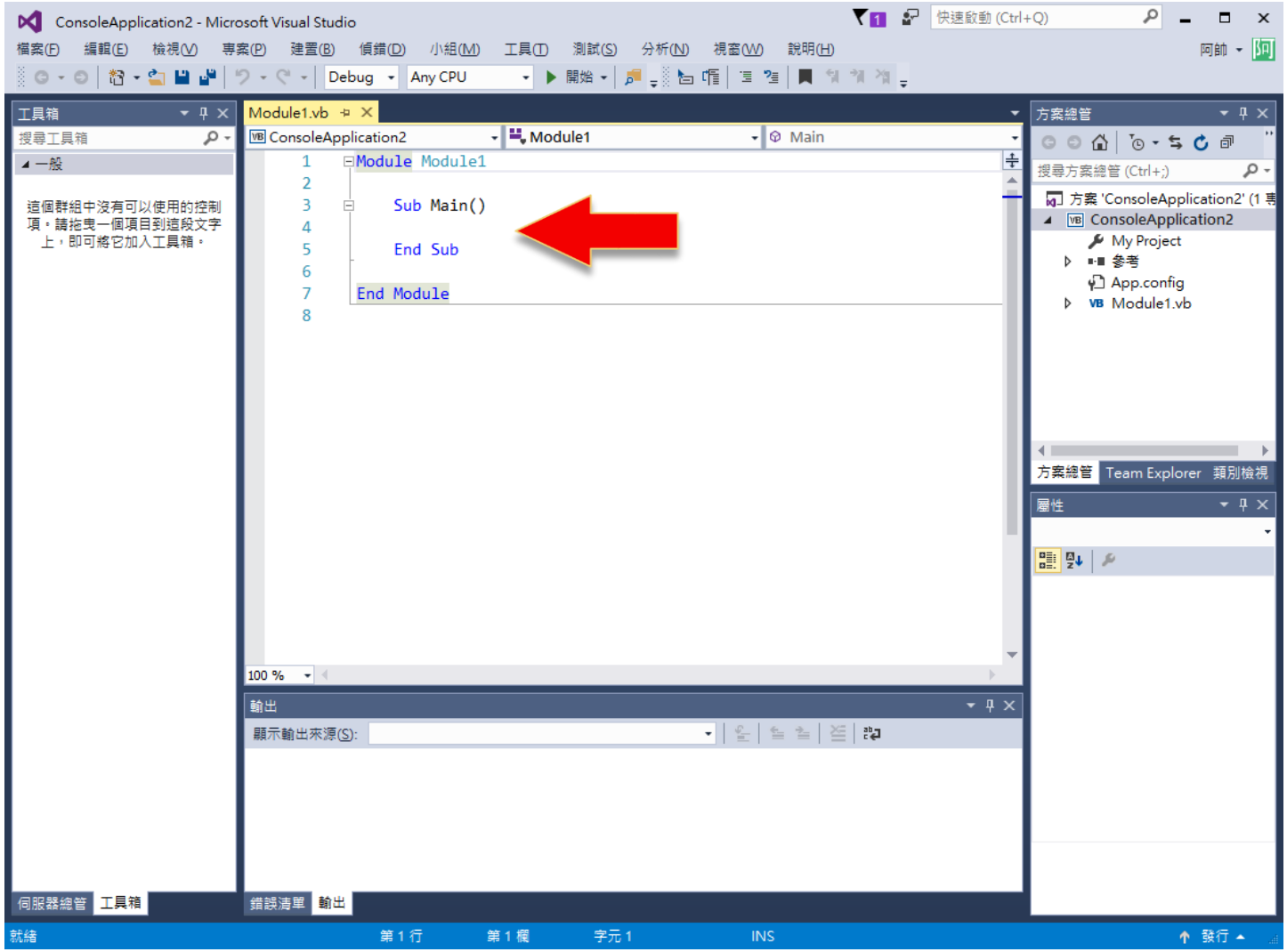
開啟Visual Studio



新增專案


新增專案
最近
已安裝的
範本
Visual C#
Visual Basic
Windows
Web
Cloud
Extensibility
Reporting
Silverlight
WCF
Workflow
測試
Visual C++
Visual F#
SQL Server
JavaScript
Python
TypeScript
遊戲
建置加速器
線上
.NET Framework 4.5.2
排序依據:
預設
搜尋已安裝的範本 (Ctrl+E)
空白應用程式 (通用 Windows) Visual Basic
Windows Form 應用程式 Visual Basic
WPF 應用程式 Visual Basic
主控台應用程式 Visual Basic
ASP.NET Web 應用程式 Visual Basic
共用的專案 Visual Basic
類別庫 Visual Basic
類別庫 (可攜式) Visual Basic
Silverlight 應用程式 Visual Basic
Silverlight 類別庫 Visual Basic
Windows 執行階段元件 (通用 Windows) Visual Basic
請按這裡連線尋找範本。
Type: Visual Basic
建立命令列應用程式專案
名稱(N):
ConsoleApplication2
位置(L):
c:\users\orion\documents\visual studio 2015\Projects
瀏覽(B)...
方案名稱(M):
ConsoleApplication2
為方案建立目錄(D)
加入原始檔控制中(U)
確定
取消

寫入你的程式碼


ConsoleApplication2 - Microsoft Visual Studio
快速啟動 (Ctrl+Q)
檔案(F) 編輯(E) 檢視(V) 專案(P) 建置(B) 偵錯(D) 小組(M) 工具(T) 測試(S) 分析(N) 視窗(W) 說明(H)
阿帥
Debug
Any CPU
開始
工具箱
搜尋工具箱
一般
這個群組中沒有可以使用的控制項。請拖曳一個項目到這段文字上，即可將它加入工具箱。
Module1.vb
ConsoleApplication2
Module1
Main
1 Module Module1
2
3 Sub Main()
4
5 End Sub
6
7 End Module
8
100 %
輸出
顯示輸出來源(S):
方案總管
搜尋方案總管 (Ctrl+;)
方案 'ConsoleApplication2' (1
ConsoleApplication2
My Project
參考
App.config
Module1.vb
方案總管 Team Explorer 類別檢視
屬性
伺服器總管 工具箱
錯誤清單 輸出
就緒 第 1 行 第 1 欄 字元 1 INS 發行

寫入你的程式碼

```
Module Module1

    Sub Main()

        Console.WriteLine("Hello World~")
        Console.WriteLine("我會寫程式")
        Console.Read()

    End Sub

End Module
```

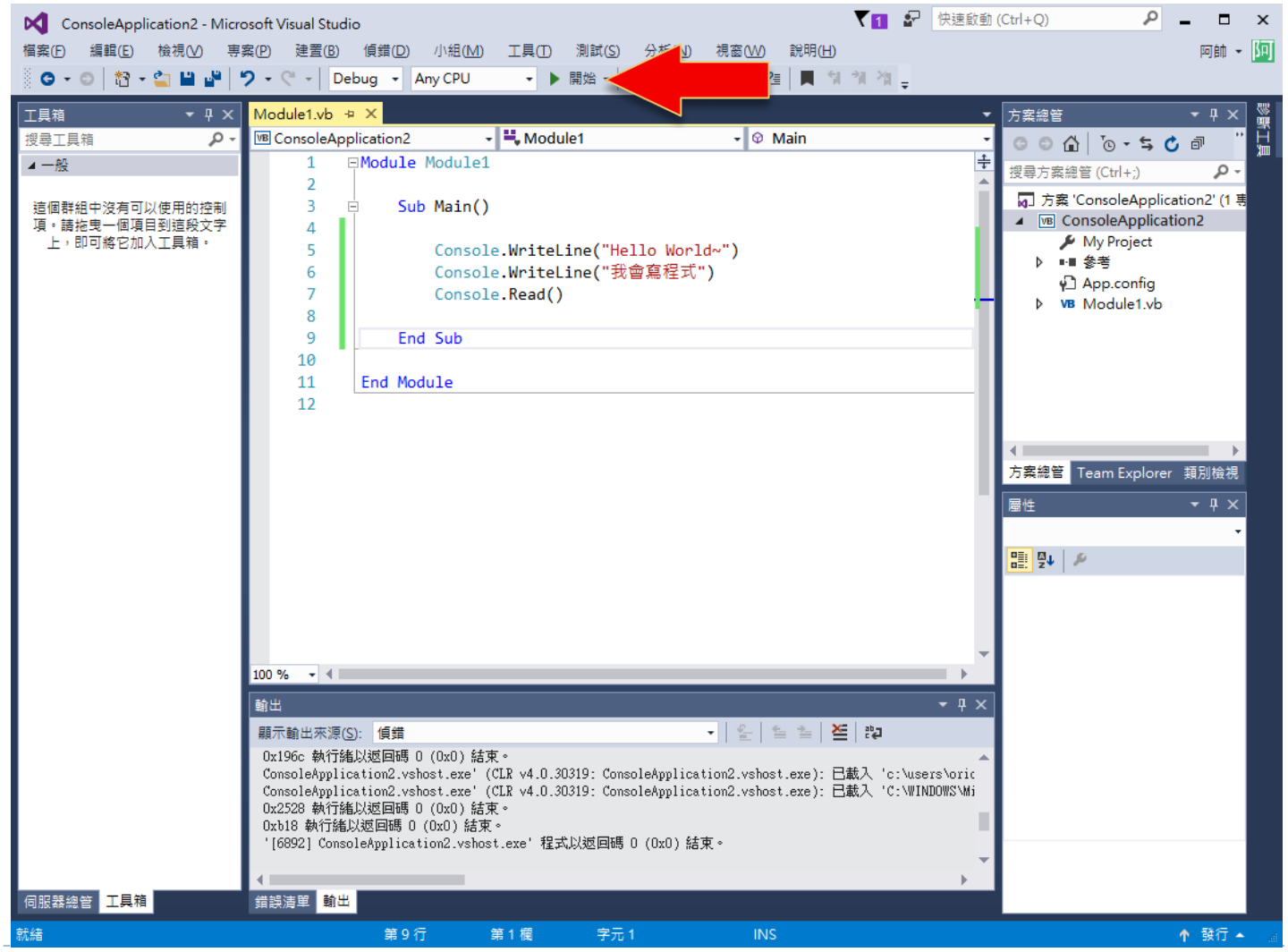
這三行

指令說明

- Console.Write(“字串”)
- 將字串輸出到螢幕
- Console.Write**Line** (“字串”)
- 將字串輸出到螢幕，並加上**換行**的動作

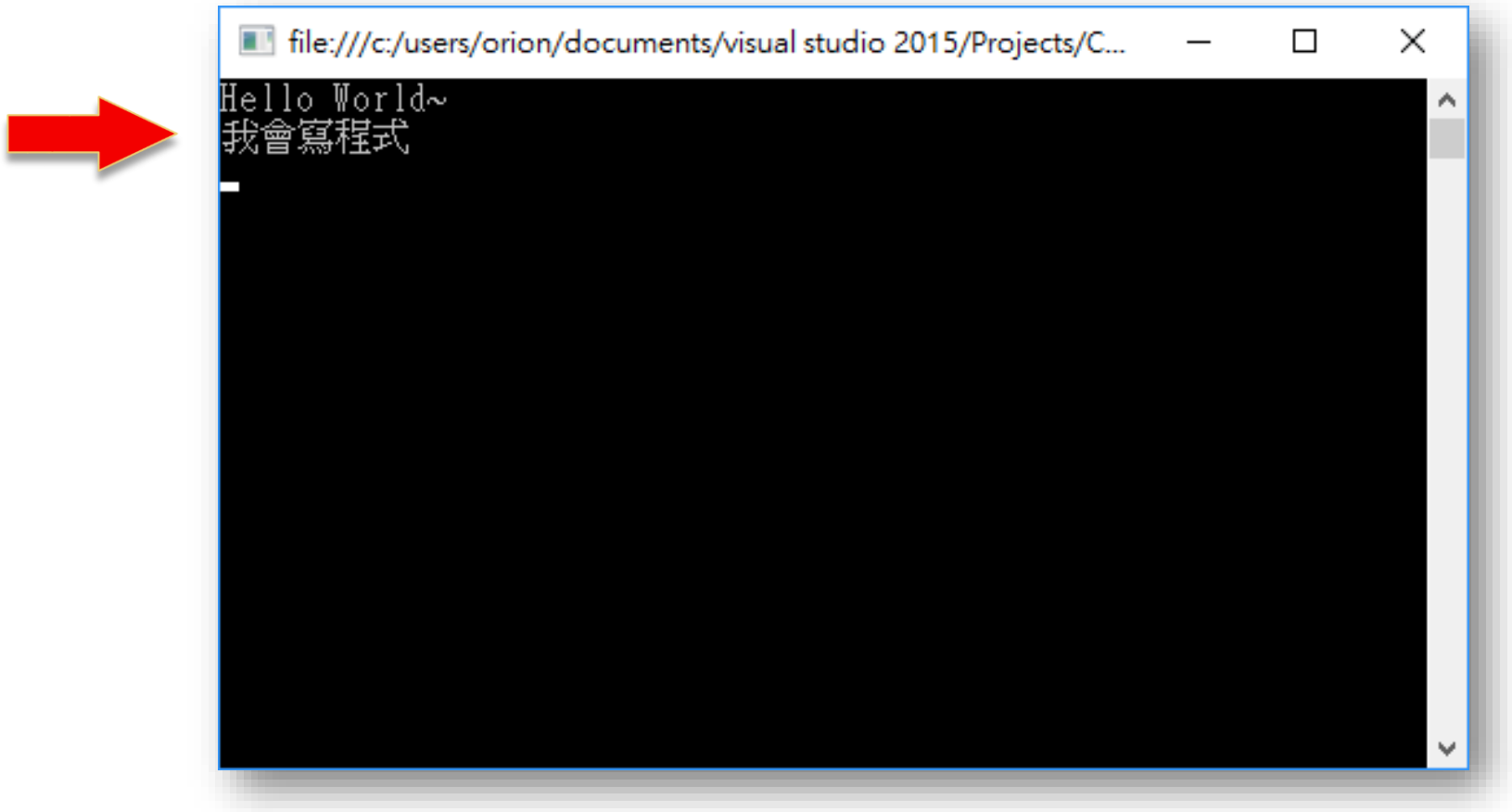
- 變數 = Console.Read( )
- 從鍵盤讀入一個字元
- 變數 = Console.Read**Line**( )
- 從鍵盤讀入一行字串
- 從鍵盤輸入的資料要找個變數接收，不然會被丟棄

執行程式

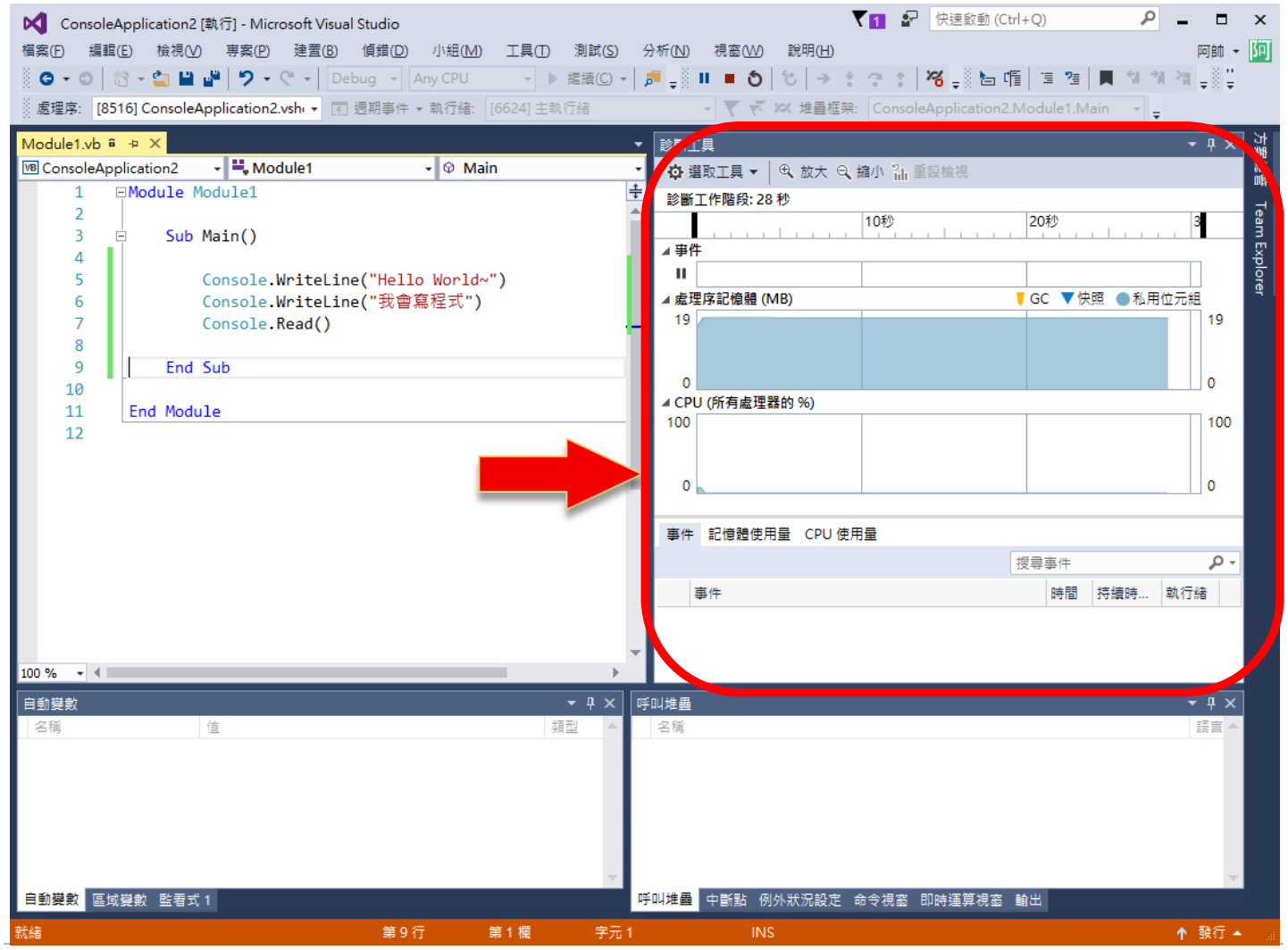

ConsoleApplication2 - Microsoft Visual Studio
Debug
Any CPU
開始
Module1.vb
ConsoleApplication2
Module1
Main
Module Module1
Sub Main()
Console.WriteLine("Hello World~")
Console.WriteLine("我會寫程式")
Console.Read()
End Sub
End Module
方案總管
ConsoleApplication2
My Project
參考
App.config
Module1.vb
輸出
顯示輸出來源(S): 偵錯
0x196c 執行緒以返回碼 0 (0x0) 結束。
0x2528 執行緒以返回碼 0 (0x0) 結束。
0xb18 執行緒以返回碼 0 (0x0) 結束。
'[6892] ConsoleApplication2.vshost.exe' 程式以返回碼 0 (0x0) 結束。
工具箱
這個群組中沒有可以使用的控制項。請拖曳一個項目到這段文字上，即可將它加入工具箱。
就緒
第 9 行
第 1 欄
字元 1
INS

執行結果

- 出現在主控台(Consloe)命令列模式下



VB也會進入監控的模式



執行與結束

執行

結束

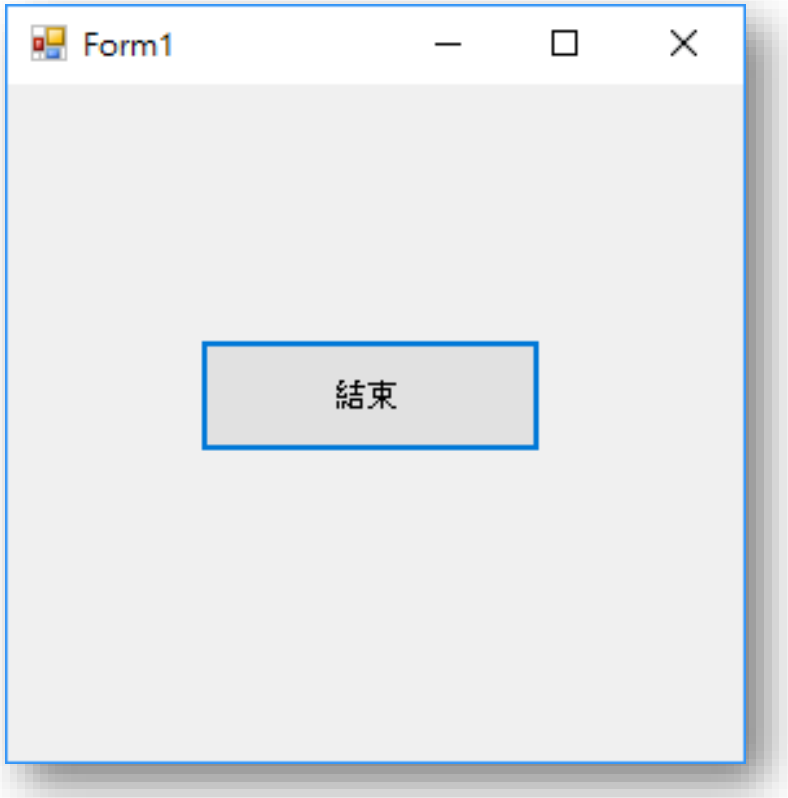
試試看

- 鍵入並執行下列程式

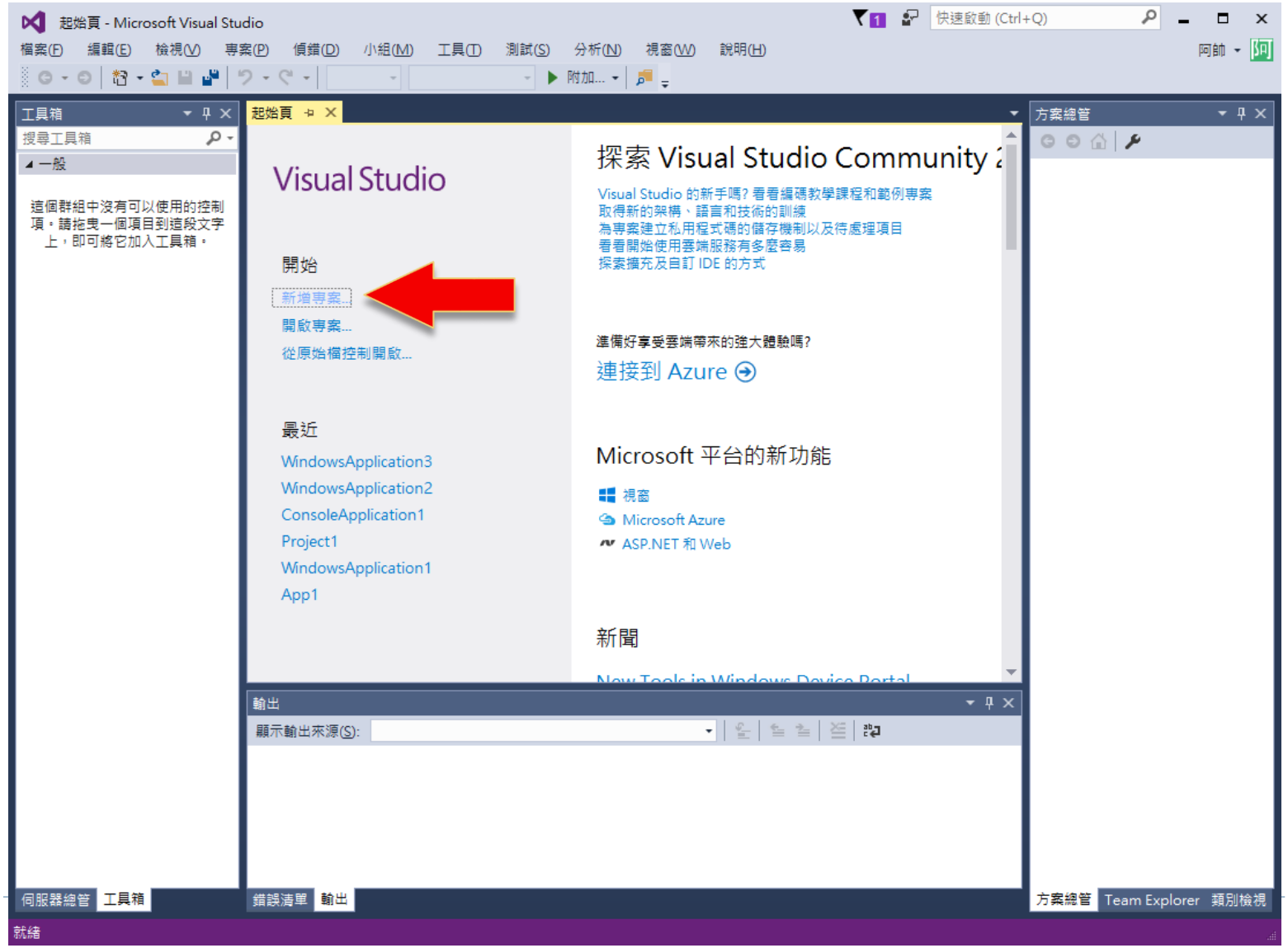
```
1  Module Module1
2
3      Sub Main()
4          Dim Name As String
5          Console.Write("What is your name? ")
6          Name = Console.ReadLine()
7          Console.WriteLine("Hello " & Name)
8          Console.ReadLine()
9      End Sub
10
11 End Module
```

VB開發環境介紹

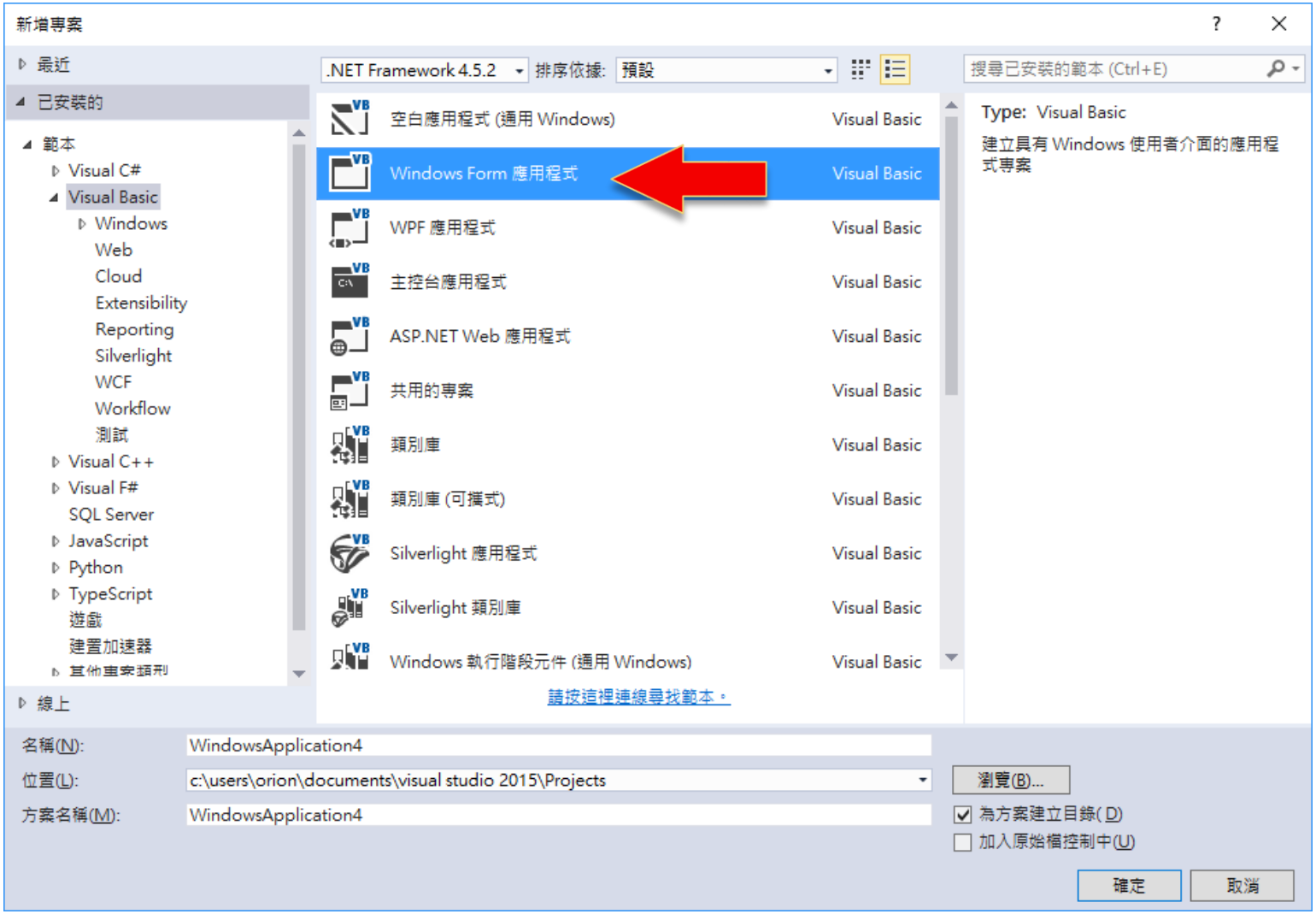
- 視窗模式(Windows Form)
- 正式的寫一個視窗應用程式



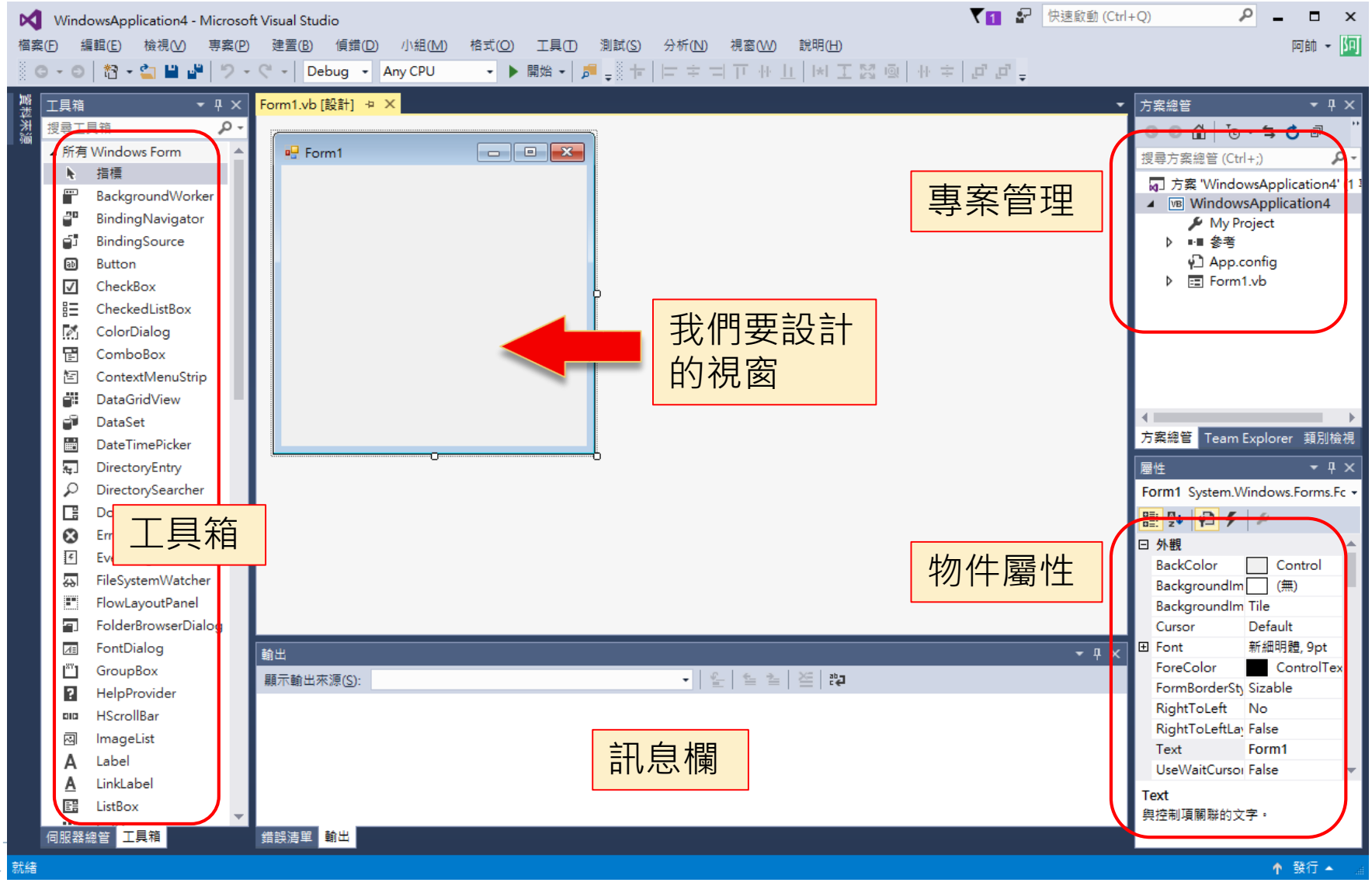
開啟Visual Studio



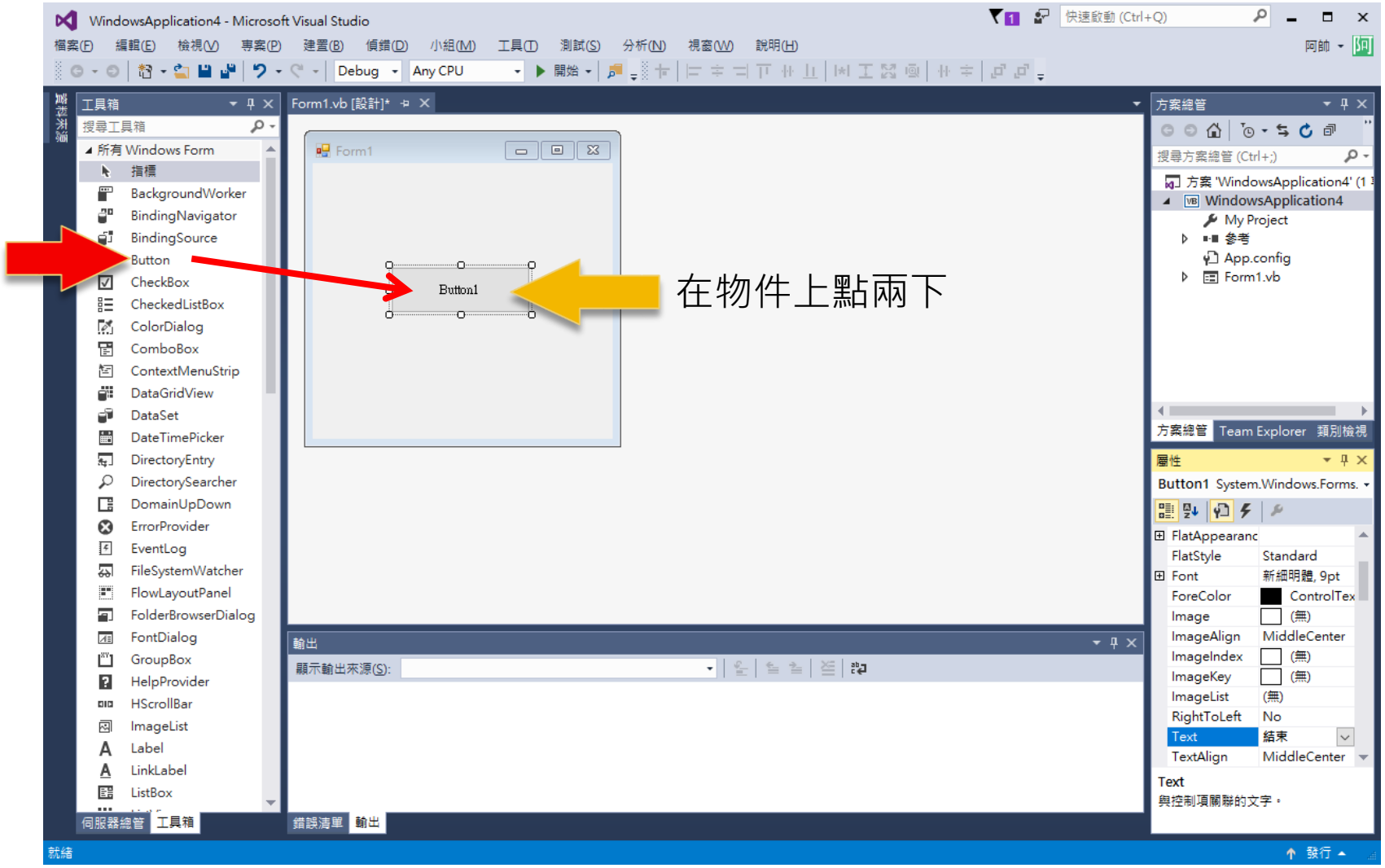
新增專案


新增專案
最近
已安裝的
範本
Visual C#
Visual Basic
Windows
Web
Cloud
Extensibility
Reporting
Silverlight
WCF
Workflow
測試
Visual C++
Visual F#
SQL Server
JavaScript
Python
TypeScript
遊戲
建置加速器
線上
.NET Framework 4.5.2
排序依據:
預設
搜尋已安裝的範本 (Ctrl+E)
空白應用程式 (通用 Windows) Visual Basic
Windows Form 應用程式 Visual Basic
WPF 應用程式 Visual Basic
主控台應用程式 Visual Basic
ASP.NET Web 應用程式 Visual Basic
共用的專案 Visual Basic
類別庫 Visual Basic
類別庫 (可攜式) Visual Basic
Silverlight 應用程式 Visual Basic
Silverlight 類別庫 Visual Basic
Windows 執行階段元件 (通用 Windows) Visual Basic
請按這裡連線尋找範本。
Type: Visual Basic
建立具有 Windows 使用者介面的應用程式專案
名稱(N):
WindowsApplication4
位置(L):
c:\users\orion\documents\visual studio 2015\Projects
瀏覽(B)...
方案名稱(M):
WindowsApplication4
為方案建立目錄(D)
加入原始檔控制中(U)
確定
取消

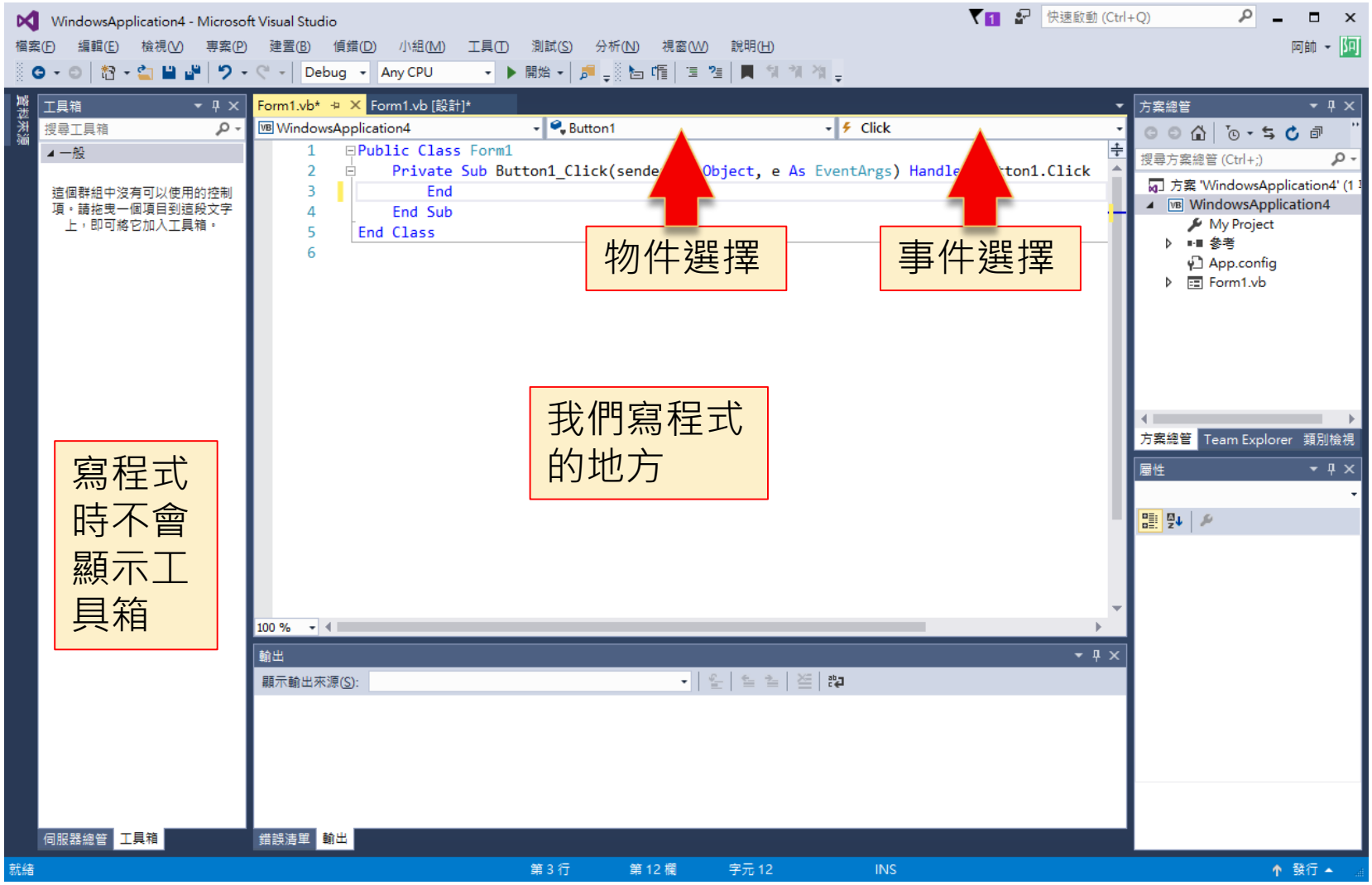
環境說明


WindowsApplication4 - Microsoft Visual Studio
工具箱
Form1.vb [設計]
Form1
專案管理
我們要設計的視窗
工具箱
物件屬性
訊息欄
方案總管
屬性
輸出
就緒

新增我們要的物件

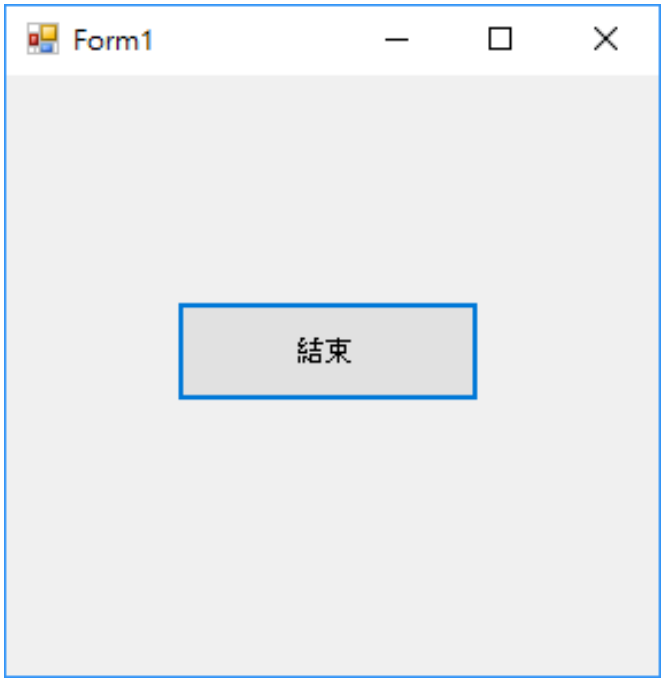

WindowsApplication4 - Microsoft Visual Studio
Button
Button1
在物件上點兩下

寫入你的程式碼

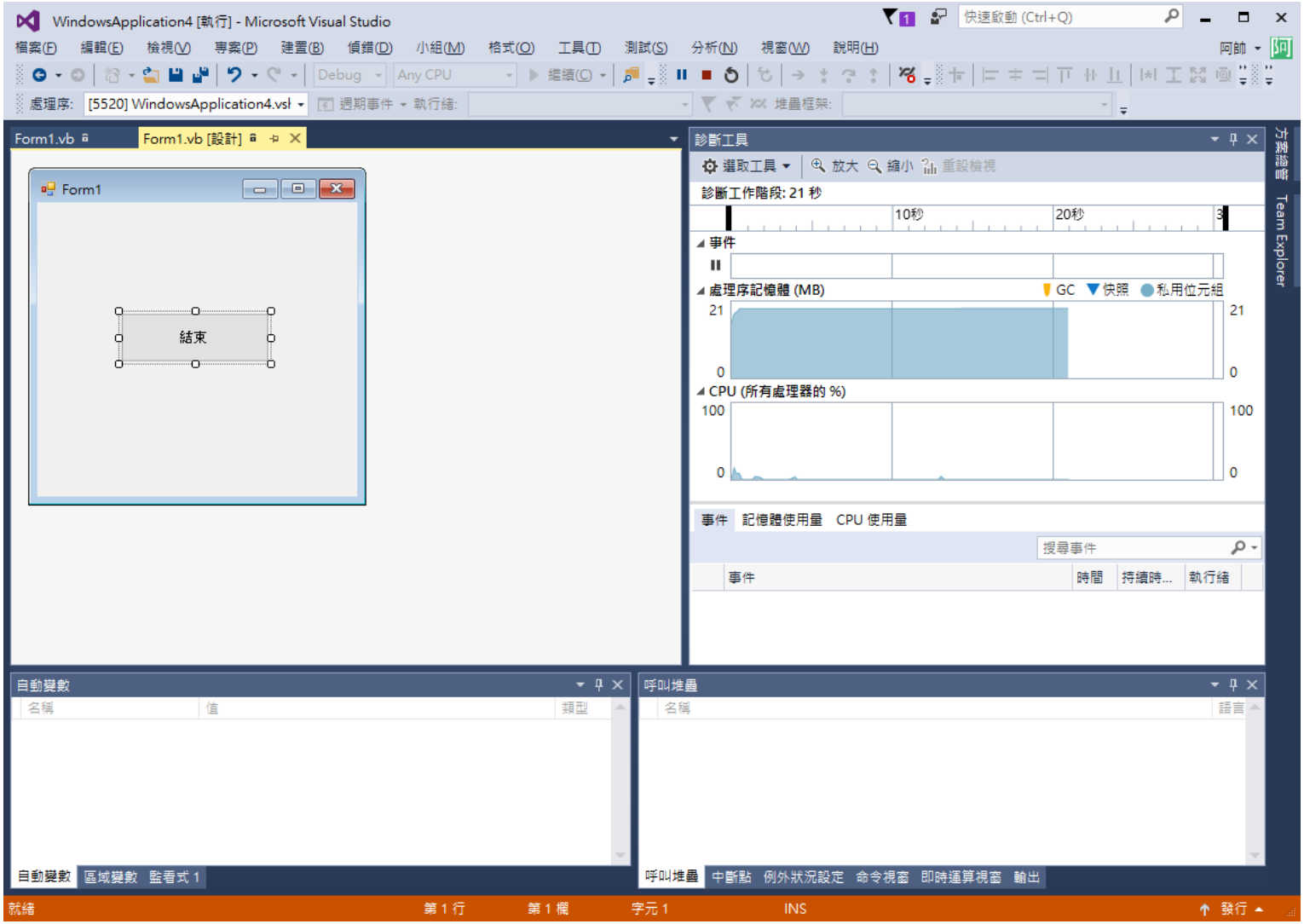

WindowsApplication4 - Microsoft Visual Studio
Debug
Any CPU
Form1.vb*
Form1.vb [設計]*
WindowsApplication4
Button1
Click
Public Class Form1
Private Sub Button1_Click(sende
Object, e As EventArgs) Handle
ton1.Click
End
End Sub
End Class
物件選擇
事件選擇
我們寫程式的地方
寫程式時不會顯示工具箱
工具箱
方案總管
WindowsApplication4
My Project
參考
App.config
Form1.vb
輸出
100 %
就緒
第 3 行
第 12 欄
字元 12
INS

執行程式

執行結果，產生一個視窗

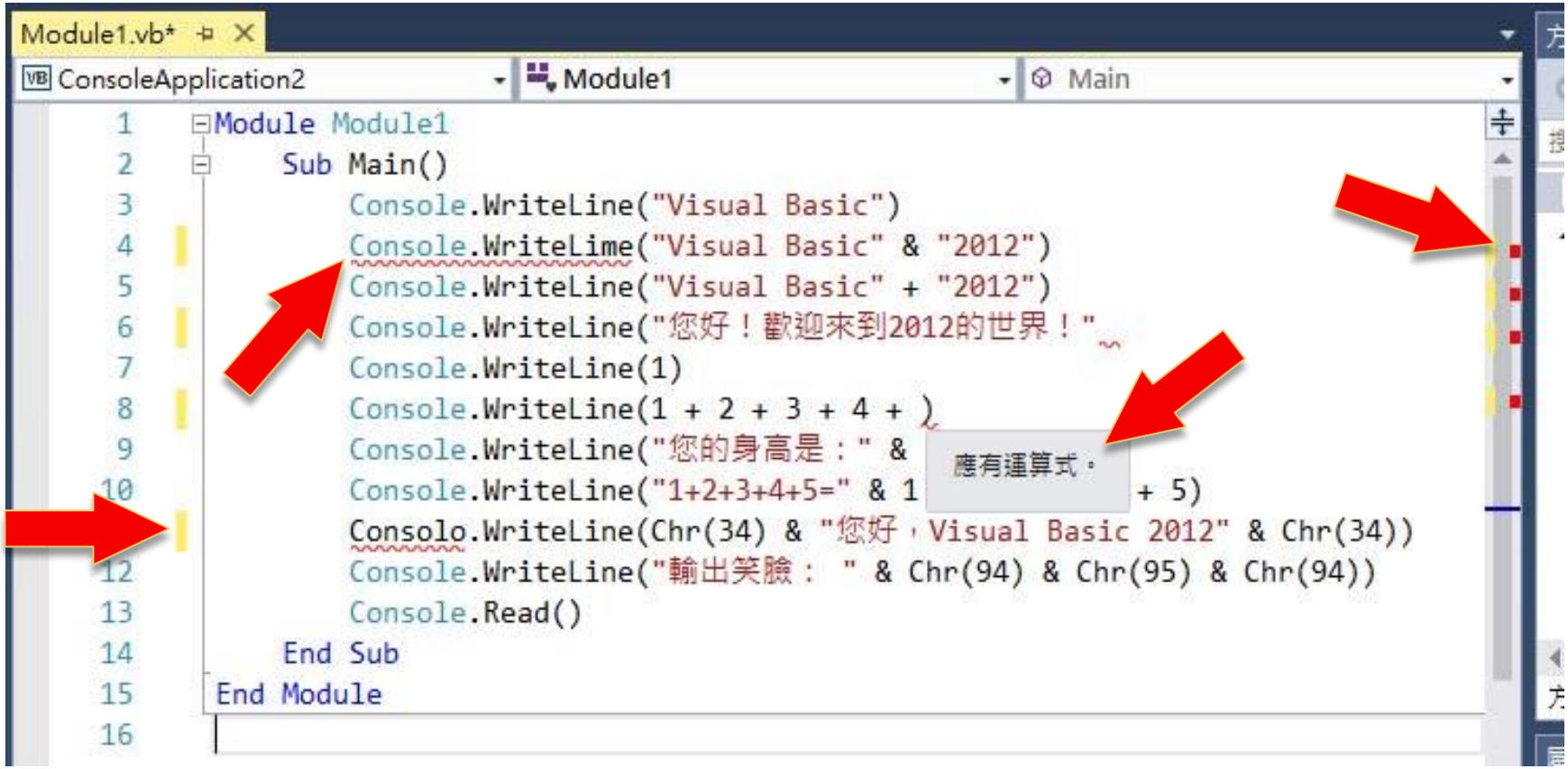


VB也會進入監控的模式



VB很聰明，有錯誤會告訴你

- 只要有紅色毛毛蟲就是有問題的地方，前後也會有黃色、紅色方塊標示，停在毛毛蟲上他會告訴你為甚麼



錯誤！！！

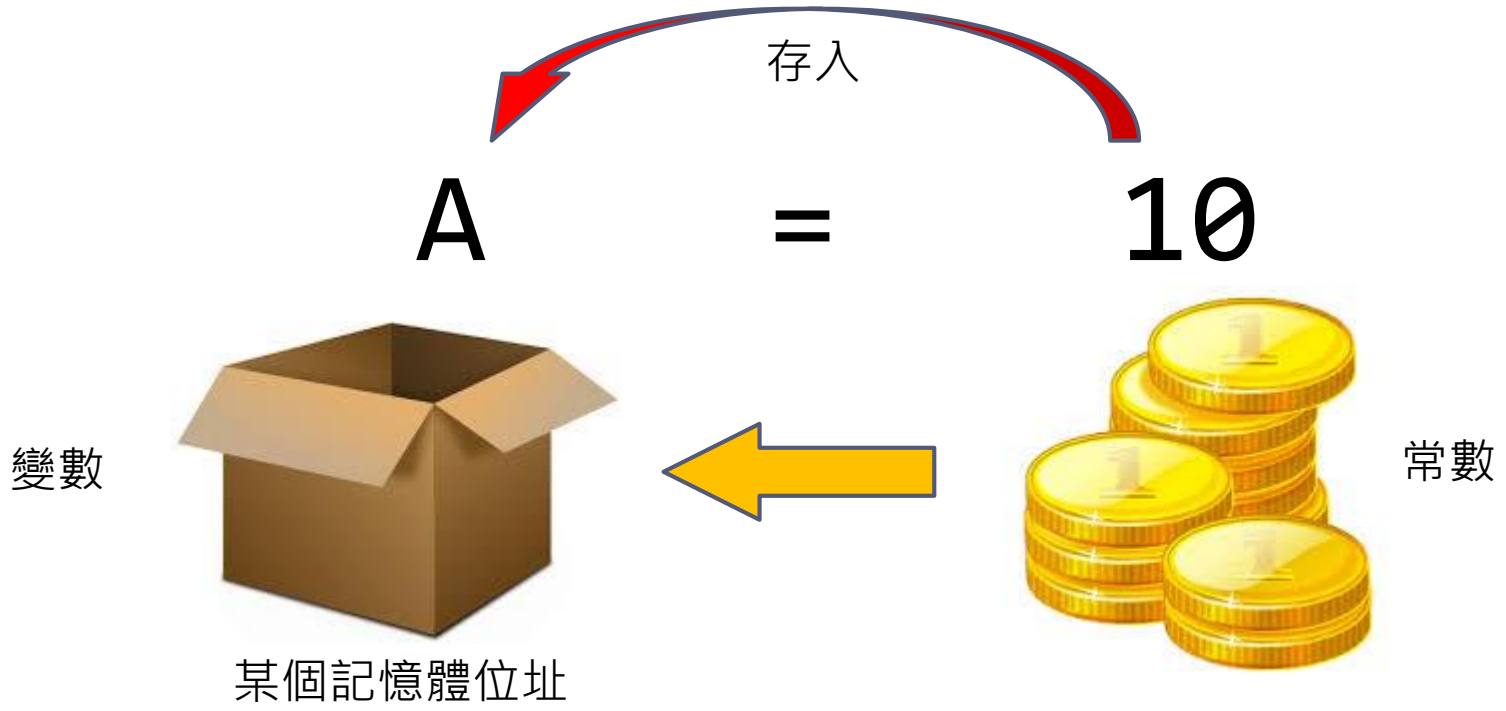
- **語法**錯誤（Syntax Error）：
  - 不符合該語言規定的寫法，在執行前都可被檢查出來
- **語意**錯誤（Semantic Error）：
  - 是邏輯上的錯誤，系統無法查知，要執行後才知道，所以程式要經過測試驗證，不是會跑就行了
  - 就是會跑，但卻不是你要的結果～
- **執行時期**錯誤（RunTime Error）：
  - 通常與系統環境有關，例如記憶體不夠了

休息一下

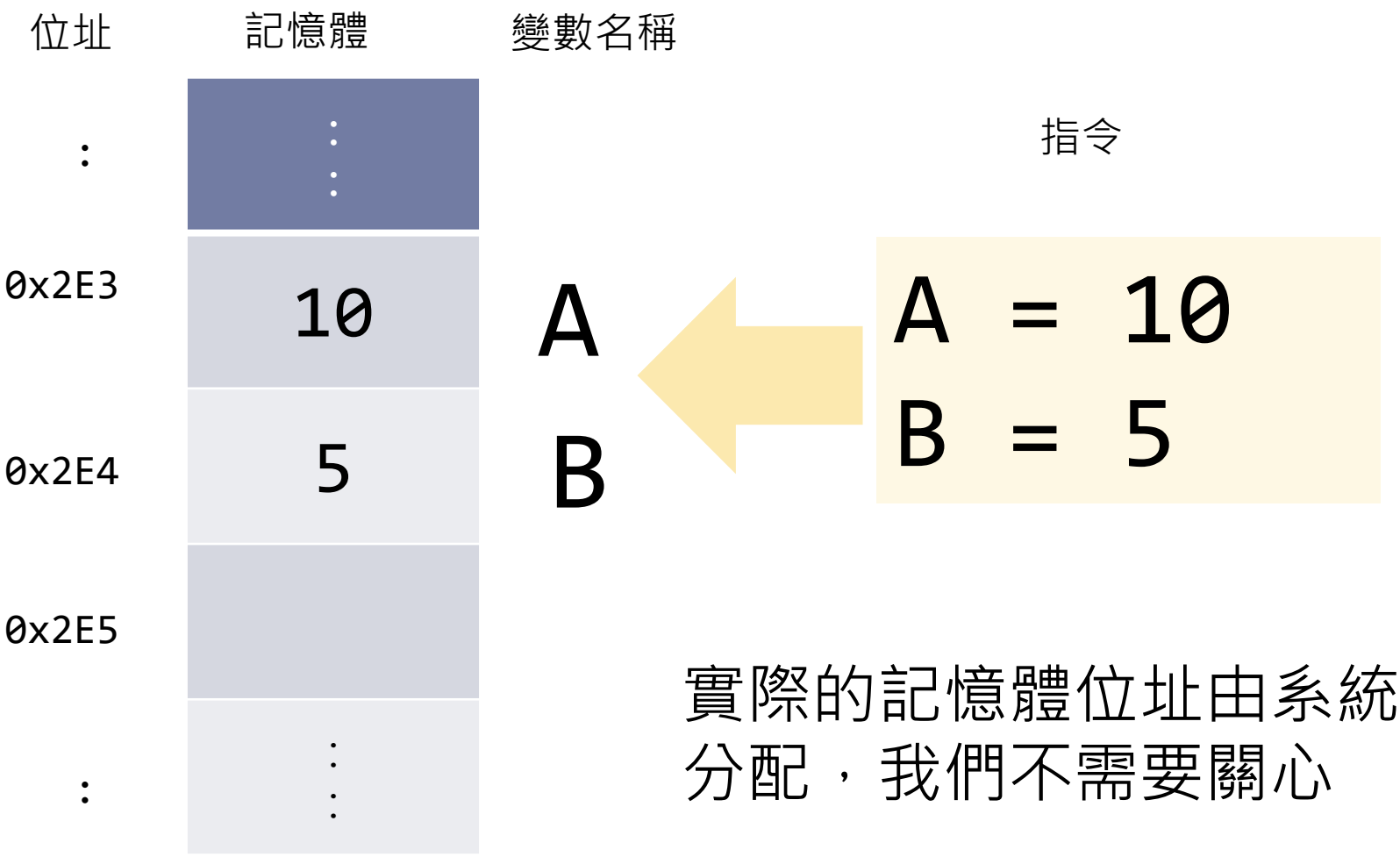
- 超級程式設計師~

基本資料處理

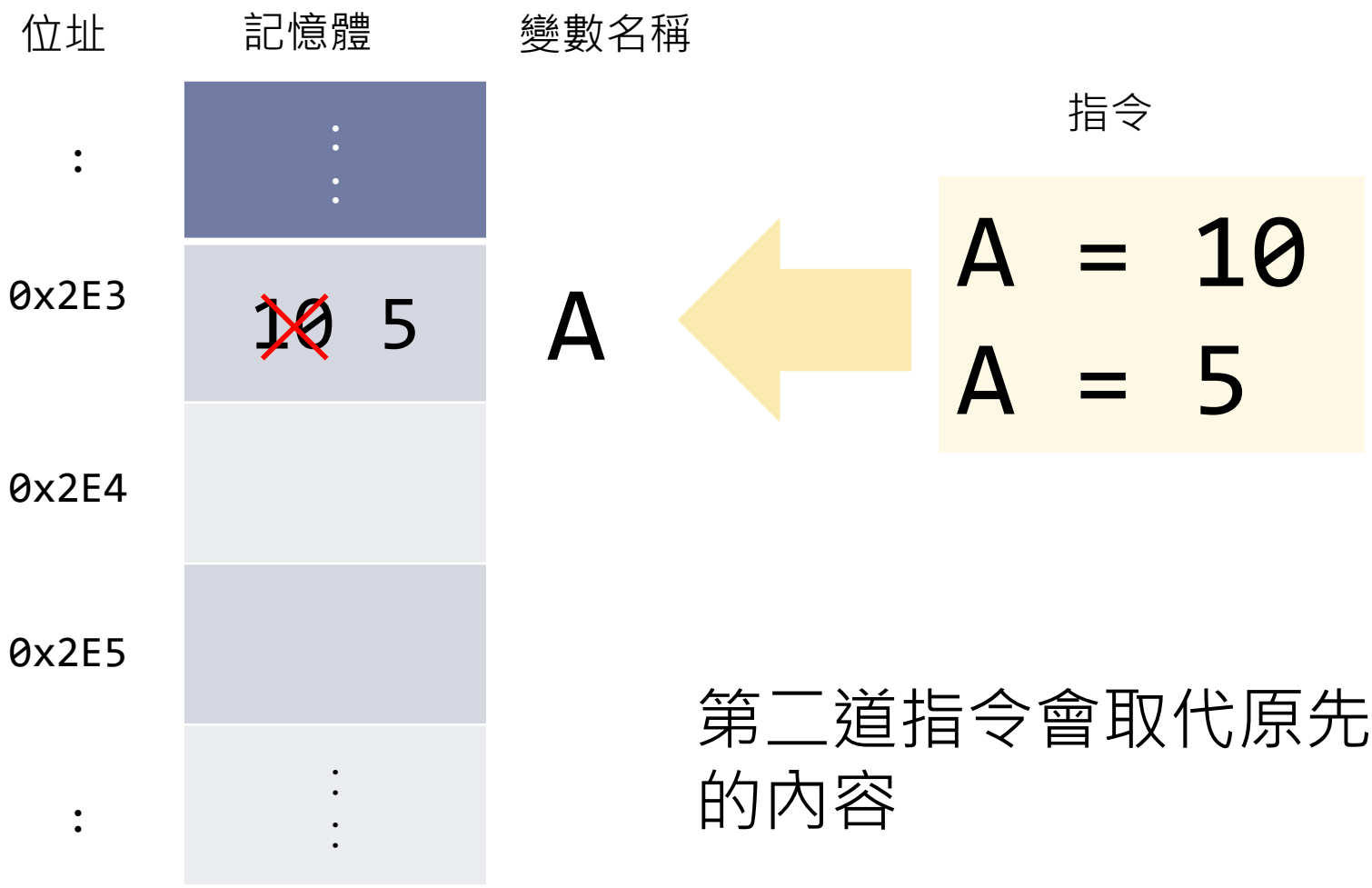
- **常數**：程式執行時固定不變的值
- **變數**：是程式中不可或缺的部分，代表一個可存放資料的記憶體位置



變數


位址
記憶體
變數名稱
:
0x2E3
10
A
0x2E4
5
B
0x2E5
:
指令
A = 10
B = 5
實際的記憶體位址由系統分配，我們不需要關心

變數的內容是可以被改變的



第二道指令會取代原先A的內容

變數命名規則

- 變數是要自己命名的，規則如下：
  - 只能由字母、數字或底線組成，不可以用標點符號或特殊字元
  - 須以英文字母開頭
  - 不能使用關鍵字命名
  - 同一個可視範圍不能有同名的變數
  - 英文沒有大小寫之分，也就是sum與SUM是相同的
  - 長度不可以超過16383個字元
  - 不要使用中文命名

變數命名規則

- 雖然英文沒有大小寫之分，但幾乎其他程式語言都有嚴格區分，所以還是養成好習慣區分一下
- 在VB， sum，SUM，Sum 都是同一個變數，但在其它語言則是三個不同的變數

要區分，不然會弄錯！

保留的關鍵字

- 關鍵字是「保留的」，表示不能使用這些關鍵字做為程式設計項目 (例如變數或程序) 的名稱
- 請參閱Page 4-4

Visual Basic Keywords			
AddHandler	AddressOf	Alias	And
AndAlso	Ansi	As	Assembly
Auto	Boolean	ByRef	Byte
ByVal	Call	Case	Catch
CBool	CByte	CChar	CDate
CDec	CDbl	Char	CInt
Class	CLng	CObj	Const
CShort	CSng	CStr	CType
Date	Decimal	Declare	Default
Delegate	Dim	Do	Double
Each	Else	ElseIf	End
Enum	Erase	Error	Event
Exit	ExternalSource	False	Finally
For	Friend	Function	Get
GetType	GoTo	Handles	If
Implements	Imports	In	Inherits
Integer	Interface	Is	Lib
Like	Long	Loop	Me
Mod	Module	MustInherit	MustOverride
MyBase	MyClass	Namespace	New
Next	Not	Nothing	NotInheritable
NotOverridable	Object	On	Option
Optional	Or	OrElse	Overloads
Overridable	Overrides	ParamArray	Preserve

常數

- 單純的**數字**或**字元**、**字串**就是一個常數
  - 例如：52、A、Apple
- 如果一個變數的內容不希望被改變，可以使用Const宣告為常數
  - 例如： Const PI = 3.141592653589
  - 這樣在程式中變數 PI 的內容就不容許再改變了

```
SUM = 199
Const Total = 65536
Name = “John”
```

哪個是常數?
哪個是變數?

為程式加上註解

- 良好的程式除了程式本身，還需要清楚的說明，為程式加上適當的註解是一個好習慣
- 註解是給人看得，系統會略過註解
- 用 ‘ 開頭的就是註解，會變成綠色字體

為程式加上註解

- 綠色字體是註解

```
1   Module Module1
2       Sub Main()
3           '----------------------
4           '我的第一個程式
5           '----------------------
6           Console.WriteLine("您好！歡迎來到Visual Basic的世界！")
7           Console.Read()      '讓畫面暫停
8       End Sub
9   End Module
10
```

- 適當的註解和縮排有助於程式閱讀

資料型態

- 由於記憶體空間有限以及效率上的考量，我們通常會給變數配置合理的記憶體空間來存放資料
- VB提供的資料型態：
  - 整數(Integer)
  - 浮點數(Float)
  - 布林(Boolean)
  - 字元(Character)
  - 字串(String)
  - 日期/時間(Date/Time)
  - 位元組(Byte)
  - 物件(Object)
  - 自訂型態(Structure)

資料型態一覽表

型態	記憶體	表示範圍
Boolean	Depends on implementing platform	`True` or `False`
Byte	1 byte	0 ~ 255 (unsigned)
Char	2 bytes	0 ~ 65535 (unsigned)
Date	8 bytes	0:00:00 (midnight) on January 1, 0001 through 11:59:59 PM on December 31, 9999
Decimal	16 bytes	0 ~ +/-79,228,162,514,264,337,593,543,950,335 (+/-7.9...E+28) † with no decimal point; 0 ~ +/-7.9228162514264337593543950335 with 28 places to the right of the decimal; smallest nonzero number is +/-0.0000000000000000000000000001 (+/-1E-28) †
Double	8 bytes	-1.79769313486231570E+308 ~ -4.94065645841246544E-324 † for negative values; 4.94065645841246544E-324 ~ 1.79769313486231570E+308 † for positive values

資料型態一覽表

型態	記憶體	表示範圍
Integer	4 bytes	-2,147,483,648 ~ 2,147,483,647 (signed)
long	8 bytes	-9,223,372,036,854,775,808 ~h 9,223,372,036,854,775,807 (9.2...E+18 †) (signed)
Object (Class)	4 bytes on 32-bit platform 8 bytes on 64-bit platform	Any type can be stored in a variable of type `Object`
Sbyte	1 byte	-128 ~ 127 (signed)
Short	2 bytes	-32,768 ~ 32,767 (signed)
Single	4 bytes	-3.4028235E+38 ~ -1.401298E-45 † for negative values; 1.401298E-45 ~ 3.4028235E+38 † for positive values
String (Class)	Depends on implementing platform	0 to approximately 2 billion Unicode characters
UInteger	4 bytes	0 ~ 4,294,967,295 (unsigned)

資料型態一覽表

型態	記憶體	表示範圍
Ulong	8 bytes	0 ~ 18,446,744,073,709,551,615 (1.8...E+19 [+]) (unsigned)
User-Defined (structure)	Depends on implementing platform	Each member of the structure has a range determined by its data type and independent of the ranges of the other members
UShort	2 bytes	0 ~ 65,535 (unsigned)

變數宣告範例

- Dim sum As Integer
  - 宣告 變數名稱 是 整數
- Dim PI As Double = 3.14
  - 宣告 變數名稱 是 倍精度 初值(可有可無)
- Dim name As Char = "A"
- Dim address As String = "花蓮，台灣"
- Dim blnTest As Boolean

- 經宣告後程式就知道有這些名稱的記憶體空間可以存放資料
- 這些記憶體空間在程式執行結束後便會釋放回系統

變數宣告範例

- Dim A, B, C As Integer
  - 宣告 這三個變數 都是 整數
  - 一次宣告多個變數時用**逗點**隔開

- Dim intA, intB As Integer
- Dim chrA, chrB As Char
  - 有些人習慣在變數命名時加上其型態，這樣看到這個變數就知道它是用來存放何種資料，不易造成混亂

變數宣告範例

- 例如：
- Dim A As Integer = 0
- Dim B As Char = “0”
  - 在電腦中 0 ≠ “0”
  - 數值0是： 00000000_2　　字元0是： 00110000_2
- 可以宣告成：
- Dim intA As Integer = 0
- Dim chrB As Char = “0
  - 這樣就比較清楚什麼變數放何種資料

變數要確定初值

- 變數使用前最好先確定初值，以免發生不確定的錯誤
- 例如：

```
Dim  Sum,  A  As  Integer
A  =  Sum  +  10
```

- 因為不知道Sum一開始是多少，所以A也無法確定了
- 雖然有些程式語言會幫我們把變數初值設為0，但還是自己設定比較保險
- 應寫成：

```
Dim  Sum,  A  As  Integer
Sum = 0
A  =  Sum  +  10
```

型態轉換

- 數值和字串運算，結果是數值還是字串?
- 雖然VB會依運算方式自動轉換資料型態，但還是自己確定先轉換比較好
- 例如：

```
a = 10
b = "100"
Console.WriteLine( a + b )
執行結果： 110
```

```
a = 10
b = "100"
Console.WriteLine( a & b )
執行結果： 10100
```

強制型態轉換

- 上例雖然結果沒有錯，但還是自己確定先轉換比較好
- 應改成：

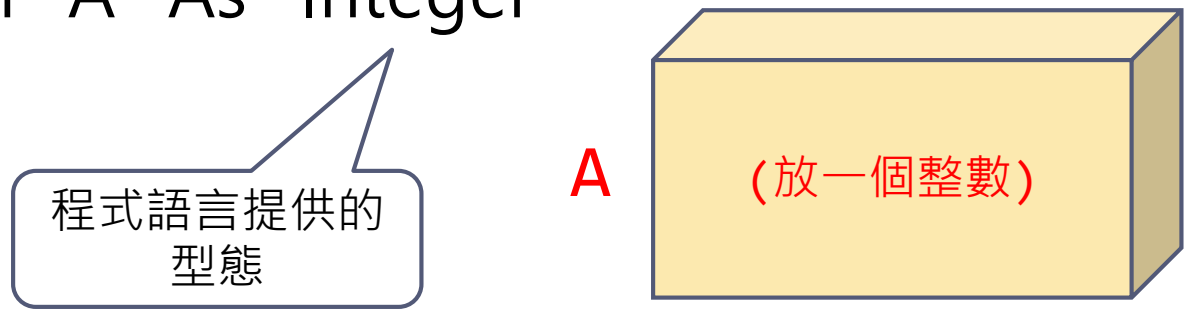
```
a = 10
b = Cint("100“)
Console.WriteLine( a + b )
執行結果： 110
```

```
a = CStr(10)
b = "100"
Console.WriteLine( a & b )
執行結果： 10100
```

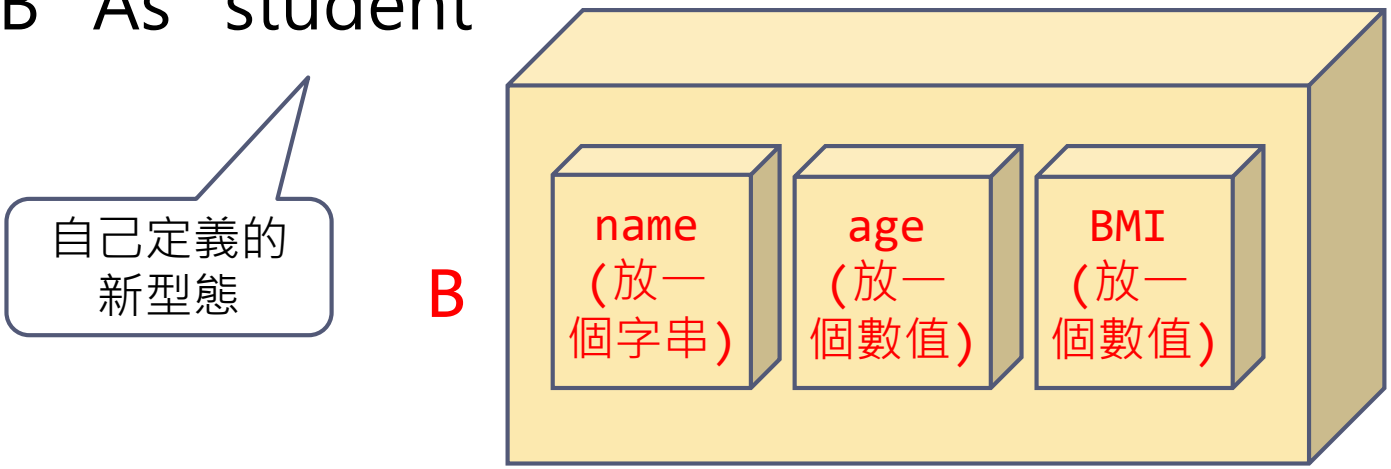
- 雖然結果相同，但可以減少非預期的錯誤

自訂型態(Structure)

- Dim A As Integer



- Dim B As student



自訂型態(Structure)

- 自己定義新的複合資料當作一個新的形態
- 例如：

```
Structure student
    Dim name As String
    Dim age As Integer
    Dim BMI As Single
End Structure
```

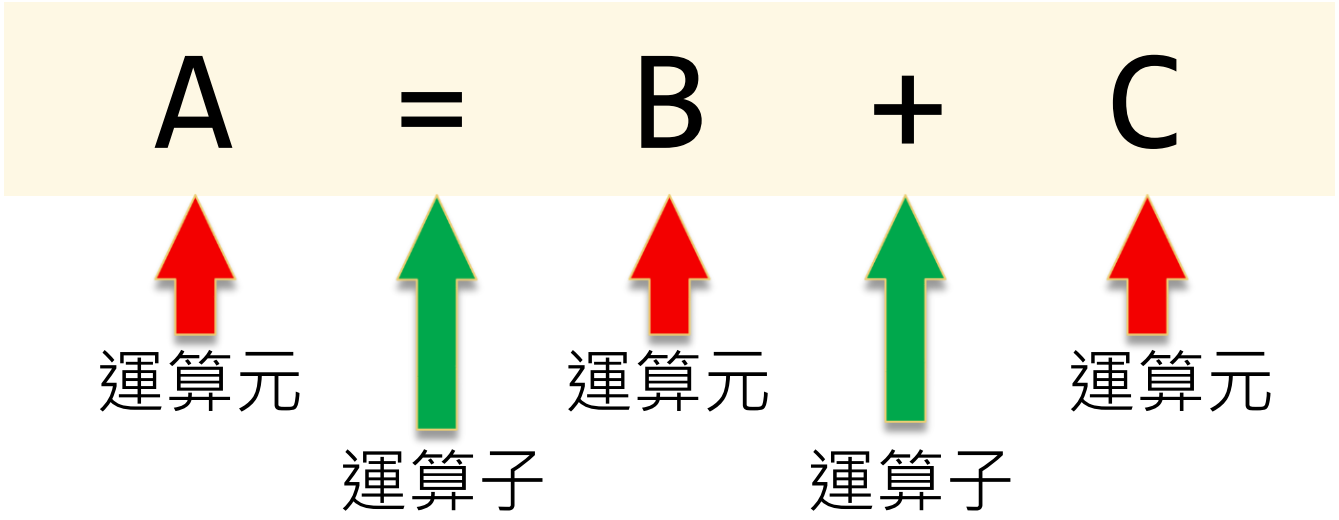
- 建立一個新的資料型態叫 student

```
Dim stu1 As student
stu1.name = “劉的華"
stu1.age = 20
stu1.BMI = 22.5
```

- 宣告stu1這個變數的型態是student，並用句點 . 來存取student內的成員

運算式與運算子

- 這是一個運算式(Expression)，由運算元(Operand)及運算子(Operator)組成



運算子類別

- 算術運算子
- 指定運算子
- 比較/關係運算子
- 串接運算子
- 邏輯運算子

- 這些是構成電腦運算式的基本符號

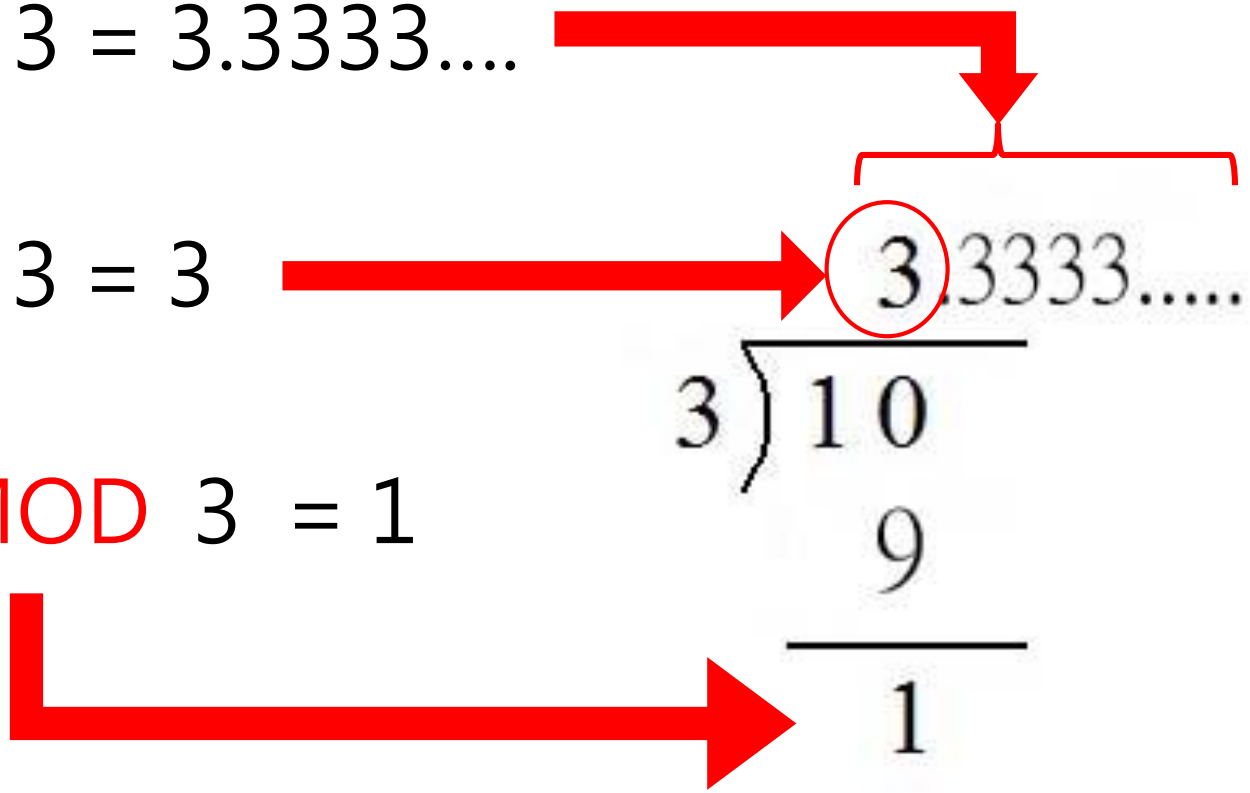
算術運算子

表6-3.3 算術運算子

名　稱	符　號	優先次序	數學運算式	VB運算式
次　方	^	1	X^2	X^2
正　號	+	2	+20	+20
負　號	-	2	-35	-35
乘　號	*	3	6×3	6*3
除　號	/	3	9÷3	9/3
整數除法	\	4	無	6\3
除法餘數	Mod	5	無	15 Mod 4
加　號	+	6	9+4	9+4
減　號	-	6	9-4	9-4

算術運算子

- 除法：
- 10 / 3 = 3.3333....
- 整除：
- 10 \ 3 = 3
- 餘數：
- 10 MOD 3 = 1



算術運算子

- 2^{10} 要寫成 2^10
- 開根號 $\sqrt{2}$ 要寫成 2 ^ (1/2) 或 2 ^ 0.5
- 開三次方 $\sqrt[3]{2}$要寫成 2 ^ (1/3)
- 請將下列運算式寫成程式的表示法：

$$\frac{-b+\sqrt{b^2-4ac}}{2a}$$

- 答案：

```
(-b + (b^2 - 4*a*c)^0.5 ) / (2*a)
```

指定運算子

表6-3.7 指派運算子

名　稱	符　號	範　例	相同運算式
指派	=	X = 2	無
相加指派	+=	X += 2	X = X + 2
相減指派	-=	X -= 2	X = X - 2
相乘指派	*=	X *= 2	X = X * 2
相除指派	/=	X /= 2	X = X / 2
整除指派	\=	X \= 2	X = X \ 2
次方指派	^=	X ^= 2	X = X ^ 2
串接指派	&=	X &= "Hello"	X = X & "Hello"

比較/關係運算子

表6-3.4 關係運算子

名　稱	符　號	範　例	範例運算結果
等　於	=	6 = 6	True
不等於	<>	4 <> 8	True
小　於	<	8 < 12	True
大　於	>	5 > 6	False
小於或等於	<=	7 <= 5	False
大於或等於	>=	9 >= 5	True

IS 比較運算

- 判斷兩者是否為**完全相同**的物件

- objX is objY 結果為 True
- objY is objZ 結果為 False
- objX is objZ 結果為 False

LIKE 比較運算

- 是功能強大的**字串**比較

結果 = 字串 LIKE 樣板

- 字串：欲比對的字串
- 樣板：由特定字元表示

樣板字元	對應字元
?	是否符合任何單一字元
*	是否符合一個或多個字元
#	是否符合任何數字
[字元集]	是否符合字元集中的內容
[!字元集]	是否不符合字元集中的內容

LIKE 比較運算

- LIKE舉例：
- Result = “J” Like “J” ➔ True
- Result = “J” Like “John” ➔ False
- Result = “John” Like “J?n” ➔ False
- Result = “John” Like “J*n” ➔ True
- Result = “J” Like “[A-M]” ➔ True
- Result = “J” Like “[!A-M]” ➔ False

想想看為甚麼？

串接運算子

- &：可串接不同型態的資料
- 例如：

```
X = “ABC” & 123 & 3.14
結果為： ABC1233.14
```

- +：只能串接文字

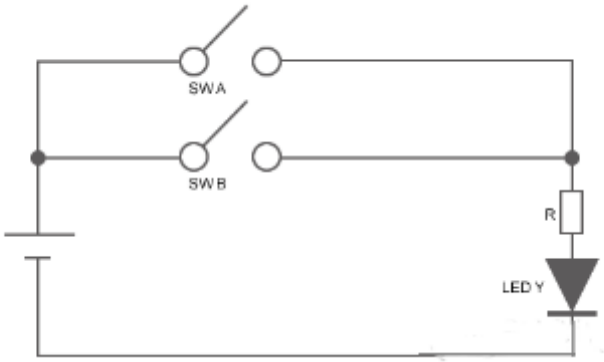
```
X = “ABC” + 123 + 3.14
結果為： (錯誤訊息)
```

- 改成：

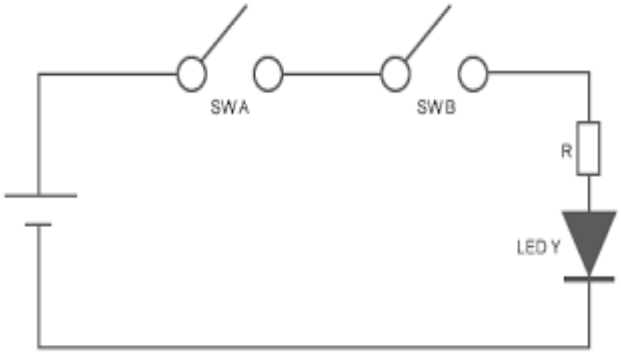
```
X = “ABC” + “123” + “3.14”
結果為： ABC1233.14
```

邏輯運算子

- OR的概念：有任一個真則結果為真



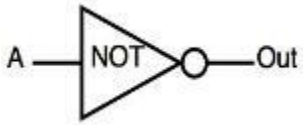
- AND的概念：有任一個假則結果為假



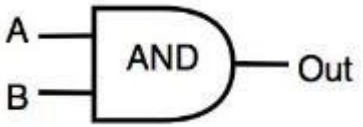
真值表(Truth Table)

- 邏輯運算子有 NOT、AND、OR、XOR
- 真值表如下：

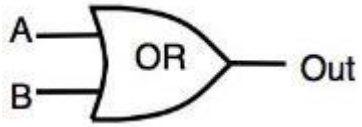
NOT 反相	
A	Out
0	1
1	0



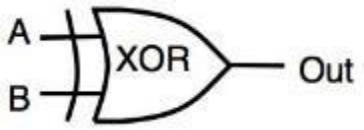
AND 且		
A	B	Out
0	0	0
0	1	0
1	0	0
1	1	1



OR 或		
A	B	Out
0	0	0
0	1	1
1	0	1
1	1	1



XOR 互斥或		
A	B	Out
0	0	0
0	1	1
1	0	1
1	1	0



- 還有 AndAlso、OrElse，不過很少用

運算的優先順序

運算子	說明
( )	括符
^	次方
+、-	正負號
*、/	乘除
\	整除
MOD	餘數
+、-	加減
&	字串串接
=、<>、<、>、<=、>=	關係比較
NOT	否
AND	且
OR	或
XOR	互斥或

運算式表示

- 因為VB的環境不是Word，所以沒有甚麼上標下標的，所有的式子就只能寫在同一行，所以像：

- $$x = \frac{(c_1 b_2 - c_2 b_1)}{(a_1 b_2 - a_2 b_1)} \quad 或 \quad c = \sqrt{a^2 + b^2}$$

- 這樣的式子是無法出現在程式語言裡的

運算式表示

- c = $\sqrt{a^2 + b^2}$ 要寫成：

```
C = ( a^2 + b^2 )^0.5
```

- x = $\frac{(c_1 b_2 - c_2 b_1)}{(a_1 b_2 - a_2 b_1)}$ 要寫成：

```
x = (c1*b2 - c2*b1) / (a1*b2 - a2*b1)
```

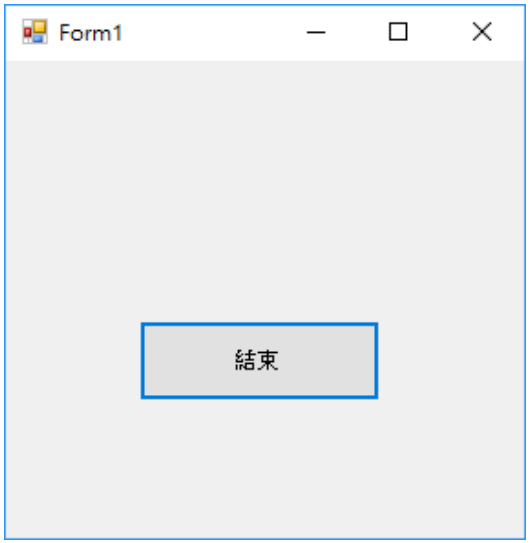
練習

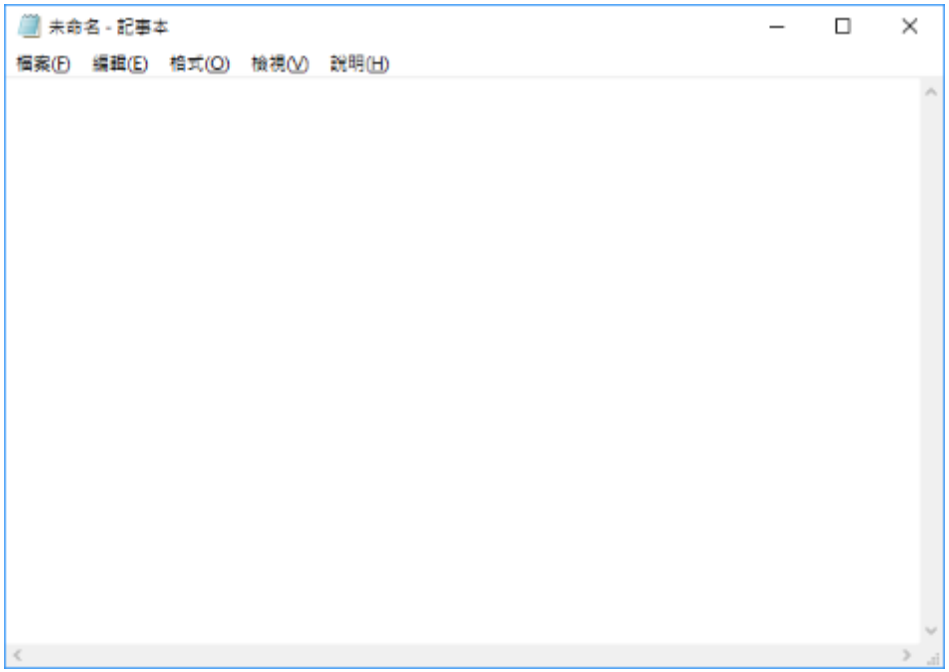
- 將下列式子寫成程式語言的寫法。

- 1. $c = a-(b+c)(3a-c)$
- 2. $\frac{(a+b)}{(c-d)} \times d$
- 3. $c = \frac{a^1+b^2}{a^2-b^2}$
- 4. $x = \frac{-b \pm \sqrt{b^2-4ac}}{2a}$

表單(Form)介紹

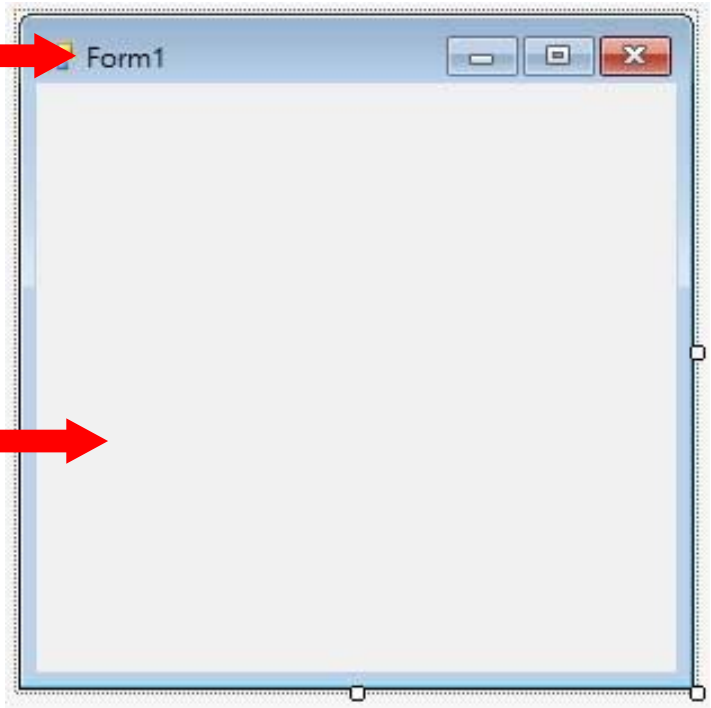
- 我們執行的每一個程式都是一個「視窗」，但在設計階段，稱為「表單(Form)」，是乘載所有物件的基礎





表單(Form)介紹

- 常用屬性：



- 還有很多，再慢慢去了解

先說一下關於顏色

- 像BackColor、ForeColor等這些有關顏色的屬性，可以在屬性表直接設定顏色外，我們也可以用函數在執行時再行設定
- 方法一：

```
Me.BackColor = Color.FromArgb(r,g,b)
```

- r、g、b代表紅、綠、藍三色的組合，數值由0~255，0代表沒有，255代表最強，
- 所以(0,0,0)是黑色，(255,255,255)是白色
- (255,0,0)是紅色，(0,255,0)是綠色.....

先說一下關於顏色

- 方法二：也可以用指定顏色名字的方式：

VB.NET Color Names

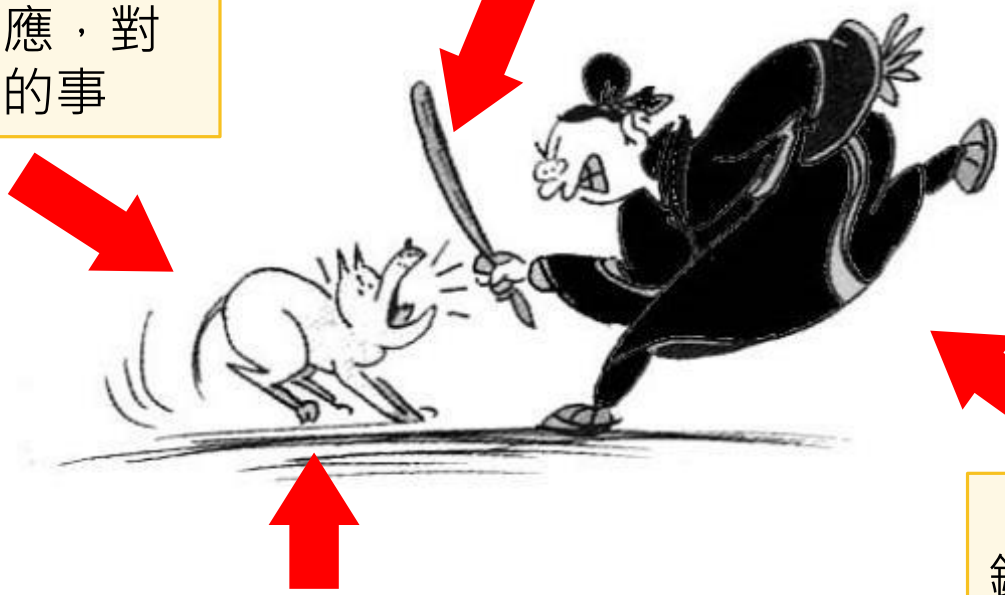
Color Name	Color	Red	Green	Blue	Color Name	Color	Red	Green	Blue
AliceBlue		240	248	255	DarkCyan		0	139	139
AntiqueWhite		250	235	215	DarkGoldenRod		184	134	11
Aqua		0	255	255	DarkGray		169	169	169
AquaMarine		127	255	212	DarkGreen		0	100	0
Azure		240	255	255	DarkKhaki		189	183	107
Beige		245	245	220	DarkMagenta		139	0	139
Bisque		255	228	196	DarkOliveGreen		85	107	47
Black		0	0	0	DarkOrange		255	140	0
BlanchedAlmond		255	235	205	DarkOrchid		153	50	204

- 例如：
```
Me.BackColor = Color.Aqua
```
- 這裡只顯示一部分，完整顏色名稱表請參考網址：
http://condor.depaul.edu/sjost/it236/documents/colorNames.htm

事件(Event)和方法(Method)

事件(Event)

方法(Method):
物件如何反應，對該反應要做的事



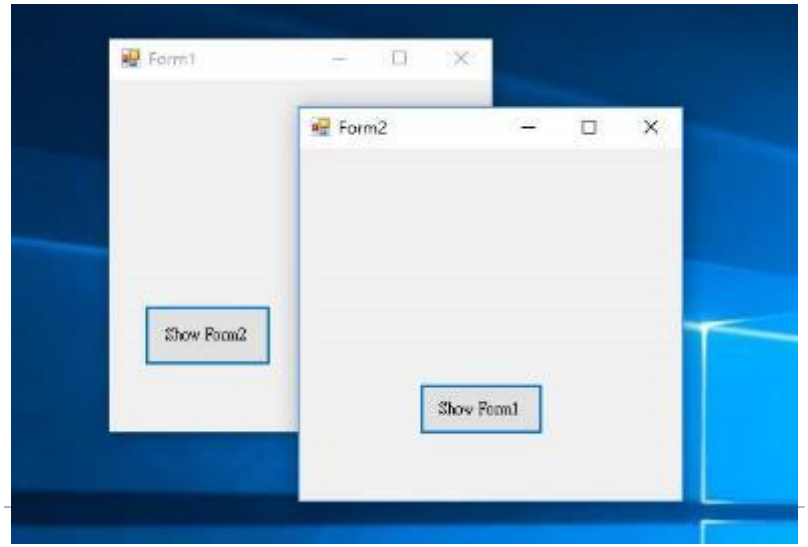
物件(Object):
程式裡的某個東西，例如表單或按鈕等

外部物件:
鍵盤、滑鼠...等，任何能發出信號的東西

表單常用事件

- Form_Load( ) :
  - 表單被載入記憶體準備執行時觸發，有什麼一開始就要設定或執行的寫在這裡，通常只在開始時執行一次
- Form_Activate( ) :
  - 如果有多重表單，當表單從非作用中表單變成作用中表單時觸發(譬如被點選)，開始要執行的動作寫在這裡

- 例如右圖，在Form1與
- Form2之間切換時就會
- 觸發Activate事件



表單常用方法

- 每一個物件都有許多內定的方法，太多 了，不用急，隨著你的功力越來越深厚慢慢就會用到
- Form最常用的就是關閉表單：

```
Me.Close()
```

- 如果是多重表單程式，這只會關閉自己，但不會結束程式，指令 END 才會完全結束程式

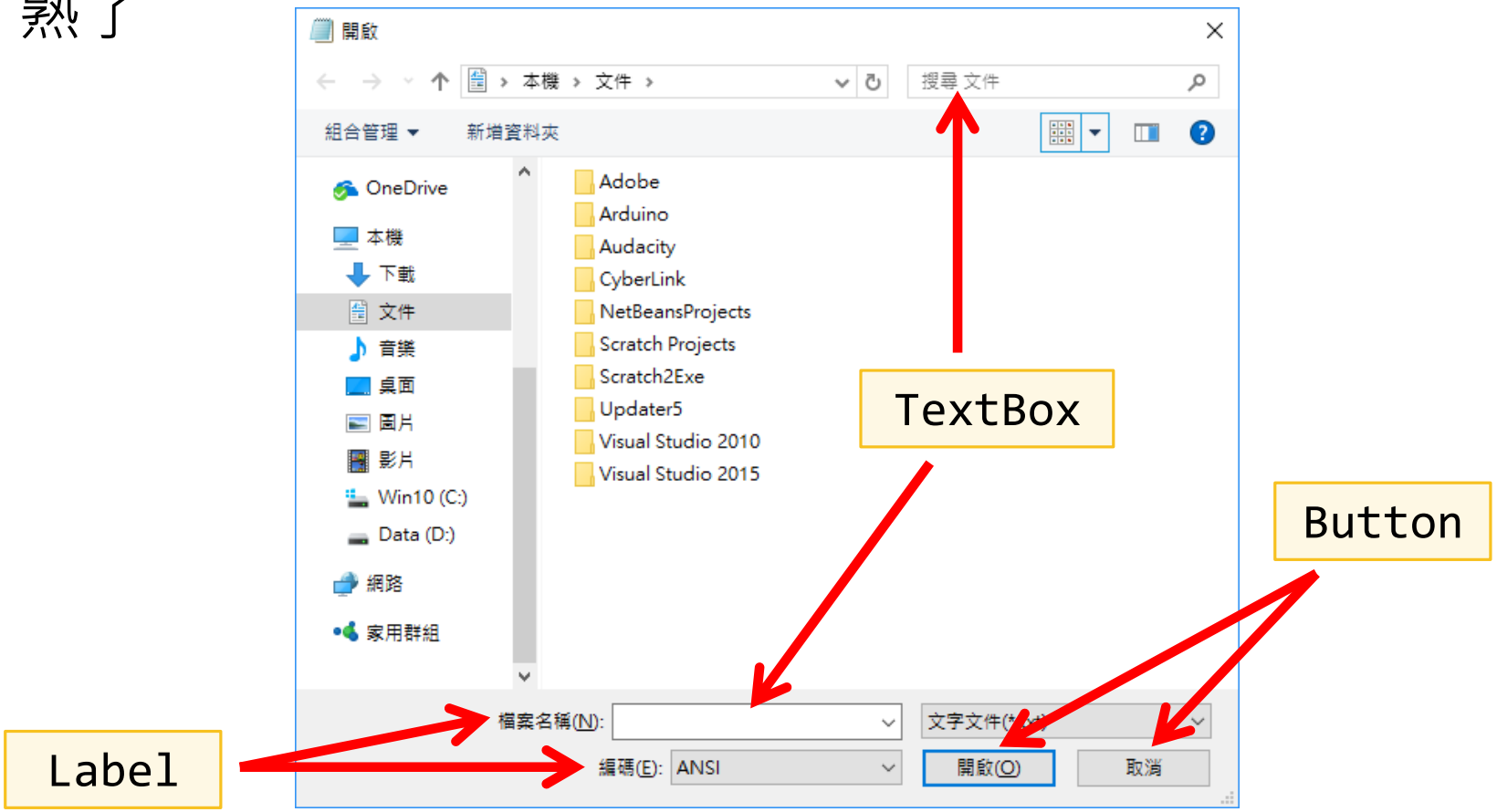
程式不難

- 書本是很好的參考資料，多看或上網查找也可以
- 觀念要懂，很重要
- 有些還是需要背一下，大部分查查書或網路就可以了

- 深厚的功力不是
  一天兩天練成的

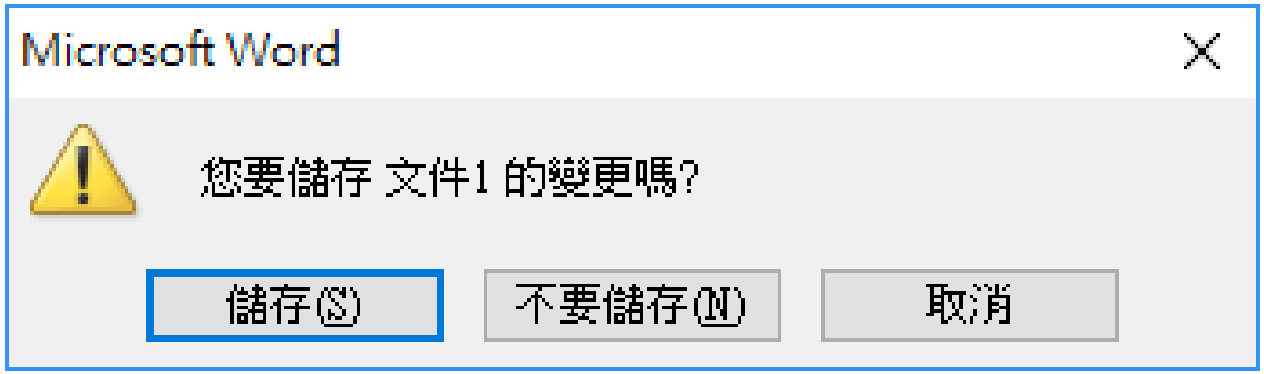
基本元件(Button、Label、TextBox)

- 視窗程式都是由這些基本元件組成的，你應該用的很熟了



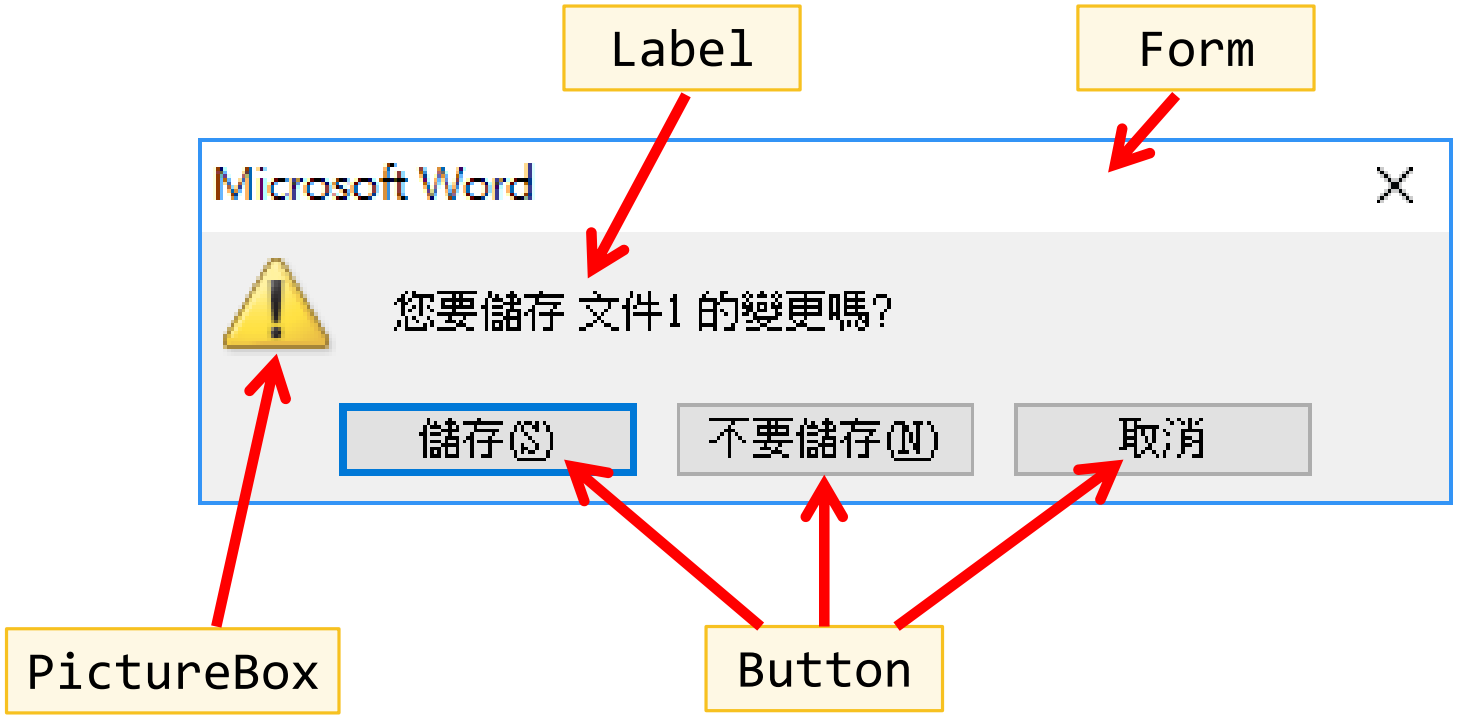
基本元件(Button、Label、TextBox)

- 這裡有幾個物件?



基本元件(Button、Label、TextBox)

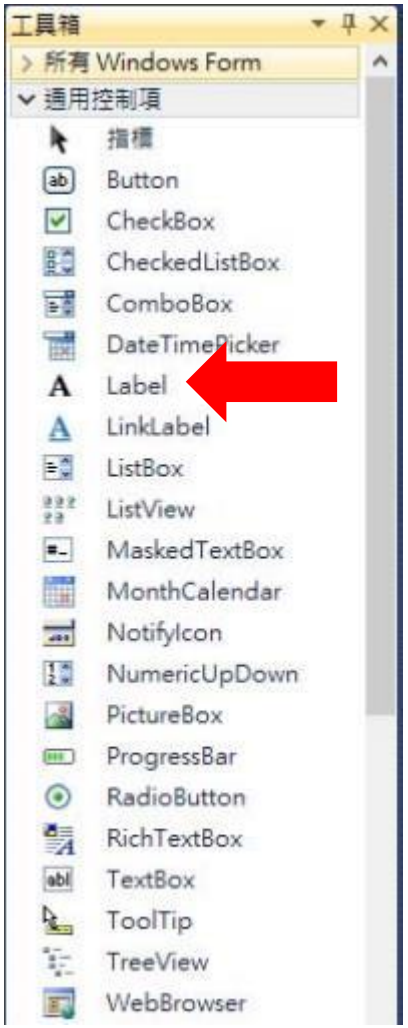
- 這裡有幾個物件?

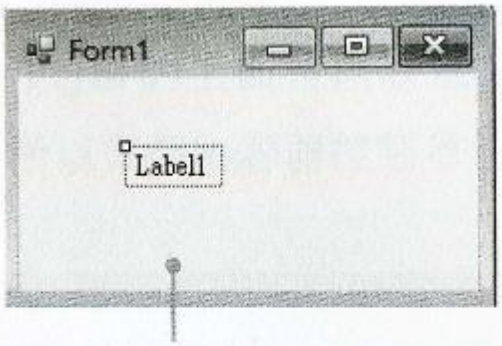


標籤(Label)

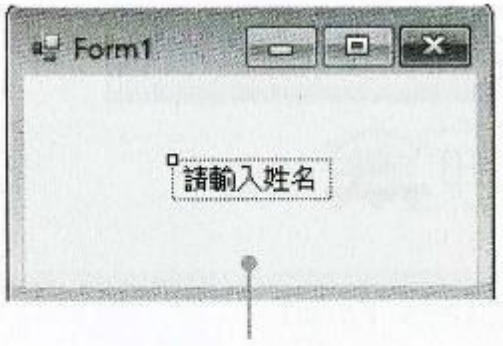
- 也是很常用到的元件，主要是用來顯示文字
- 常用屬性：
  - Text：標籤上要顯示的文字
  - Font：按鈕上顯示文字的字型
  - Visiable：該標籤是否顯示出來或隱藏
- 常用事件：
  - 比較沒有，因為通常就是顯示資訊，不當作控制項

標籤(Label)





將 Label 安排於 Form 的畫面，Text 屬性預設 Label1，且物件名稱預設為 Label1。



將 Text 屬性修改為「請輸入姓名」。

文字方塊(TextBox)

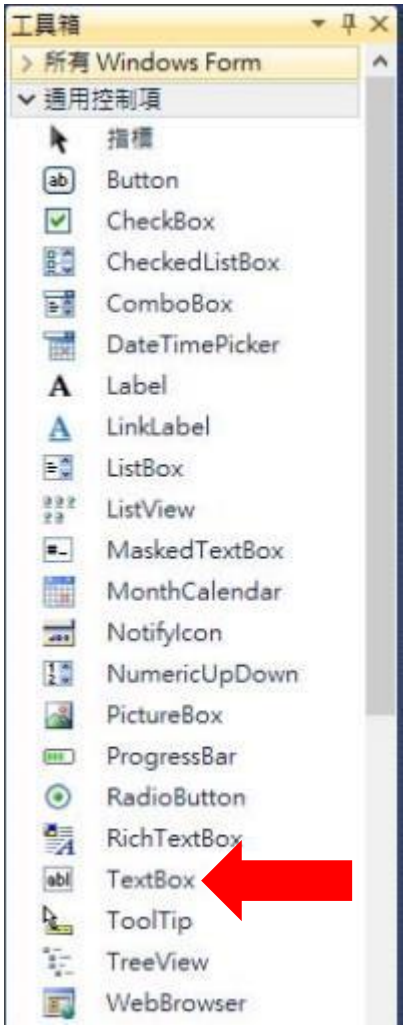
- 主要用作輸入資料，接受使用者的輸入資訊
- 常用屬性：
  - Text：文字方塊內的文字
  - Font：文字方塊顯示文字的字型和大小
  - MultiLine：是否接受多行
  - MaxLength：最多可接受多少字元
  - PasswordChar：使用者輸入以*號顯示，通常用於輸入密碼欄位
  - ScrollBars：是否使用捲軸

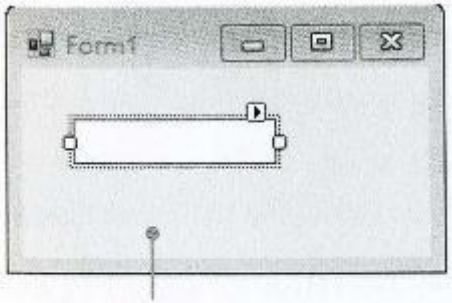
文字方塊(TextBox)

- 常用事件：
  - TextChanged：輸入項內的文字發生改變時觸發
  - MouseMove：滑鼠指標經過此TextBox上方時

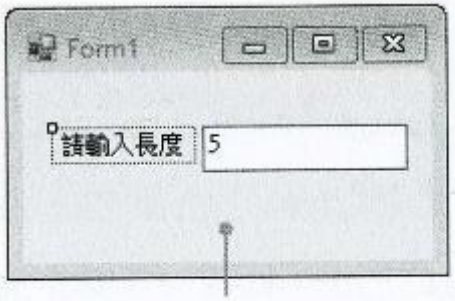
- TextBox取得的輸入皆視為「文字」，若需要數值資料，則須經過Val( )函數轉換。若要將數值寫入TextBox，需先用Str( )函數將其轉成文字。

文字方塊(TextBox)





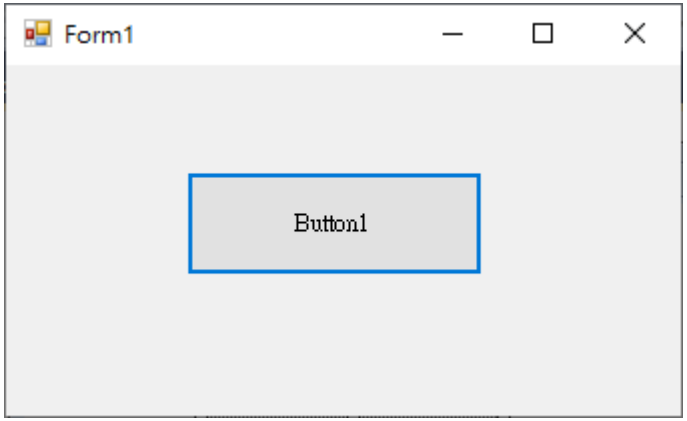
於 Form 安排一個 TextBox 的畫面，其 Text 屬性預設值是空白。



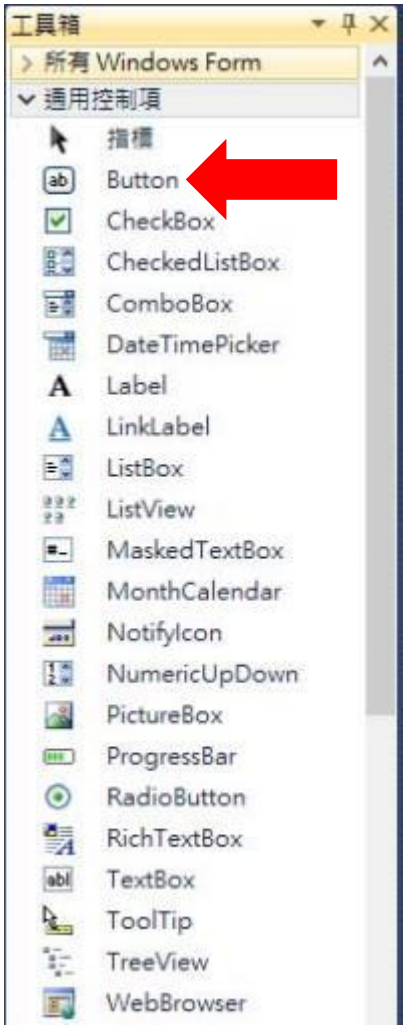
安排一個 Label 及 TextBox 的畫面，Label 通常是輸入提示或輸出資料，TextBox 則是輸入物件。

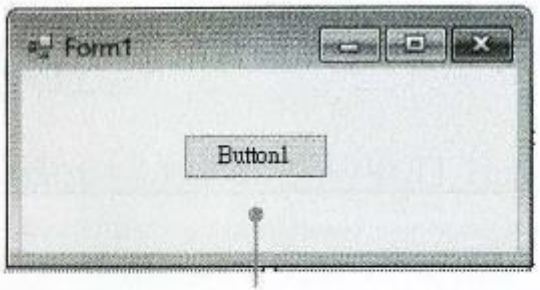
按鈕(Button)

- 大概是最常用到的元件了，主要是讓使用者點選
- 常用屬性：
  - Text：按鈕上顯示的文字
  - Font：按鈕上顯示文字的字型
- 常用事件：
  - Click：當按鈕被滑鼠點一下時

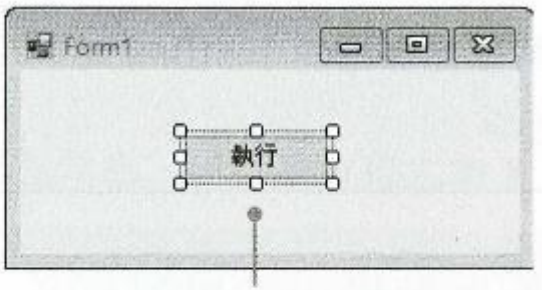


按鈕(Button)





安排一個 Button，其 Text 屬性預設值是 Button1，Name 亦為 Button1。



將 Text 屬性值改為「執行」的畫面。

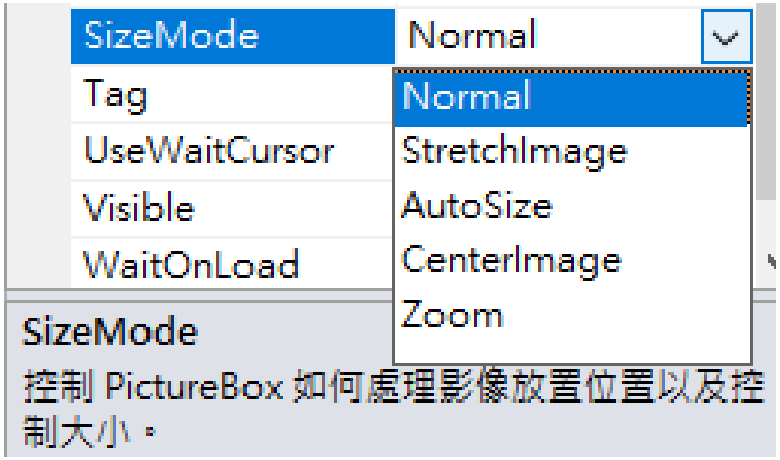
按鈕(Button)

- 在按鈕上點兩下就可以進入寫程式的畫面

```
Public Class Form1

    Private Sub Button1_Click(ByVal sender As System.Object……

        按鈕被按下後要做的動作寫在這裡

    End Sub
End Class
```

圖片盒(PictureBox)

- 用來顯示圖片或圖示
  - 可允許載入bmp、ico、jpg、png、gif、wmf等格式
- 常用屬性：
  - Image：要顯示的圖形檔
  - Height、Width：圖片的長、寬
  - Left、Top：定位圖片的位置(左上角)
  - SizeMode：顯示模式



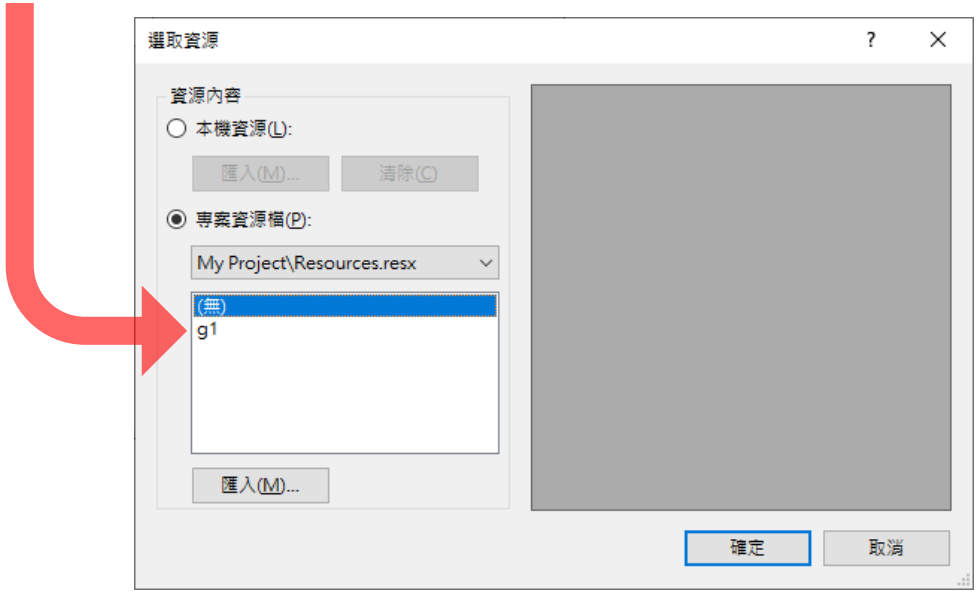
圖片盒(PictureBox)

- 圖片的來源可以是檔案或專案的資源檔
  - 1.直接指定檔案來源：

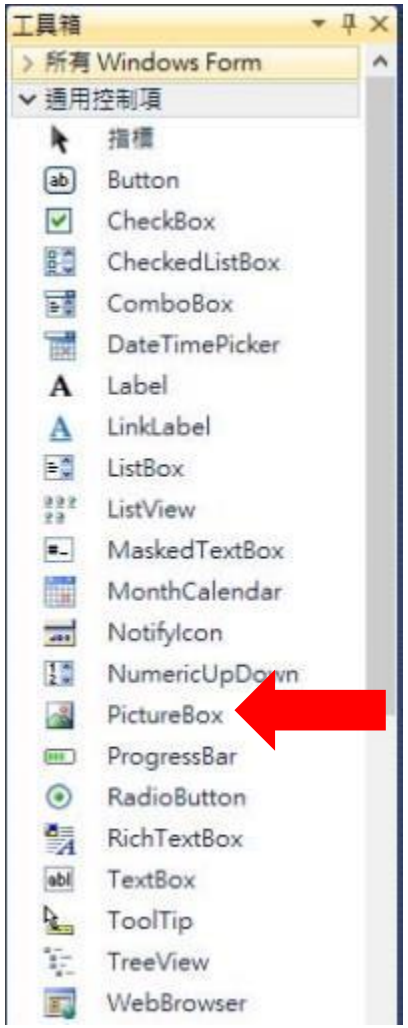
```
PictureBox1.Image = ImageFromFile(“c:\...\...”)
```

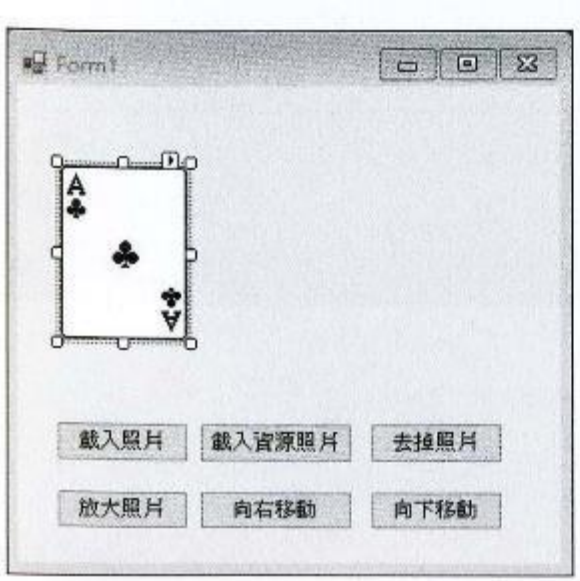
  - 2.使用已匯入資源檔的圖片：

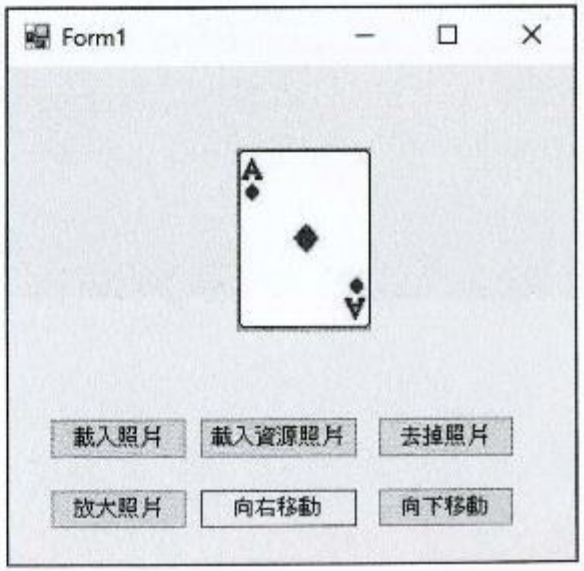
```
PictureBox1.Image = My.Resources.g1
```



圖片盒(PictureBox)







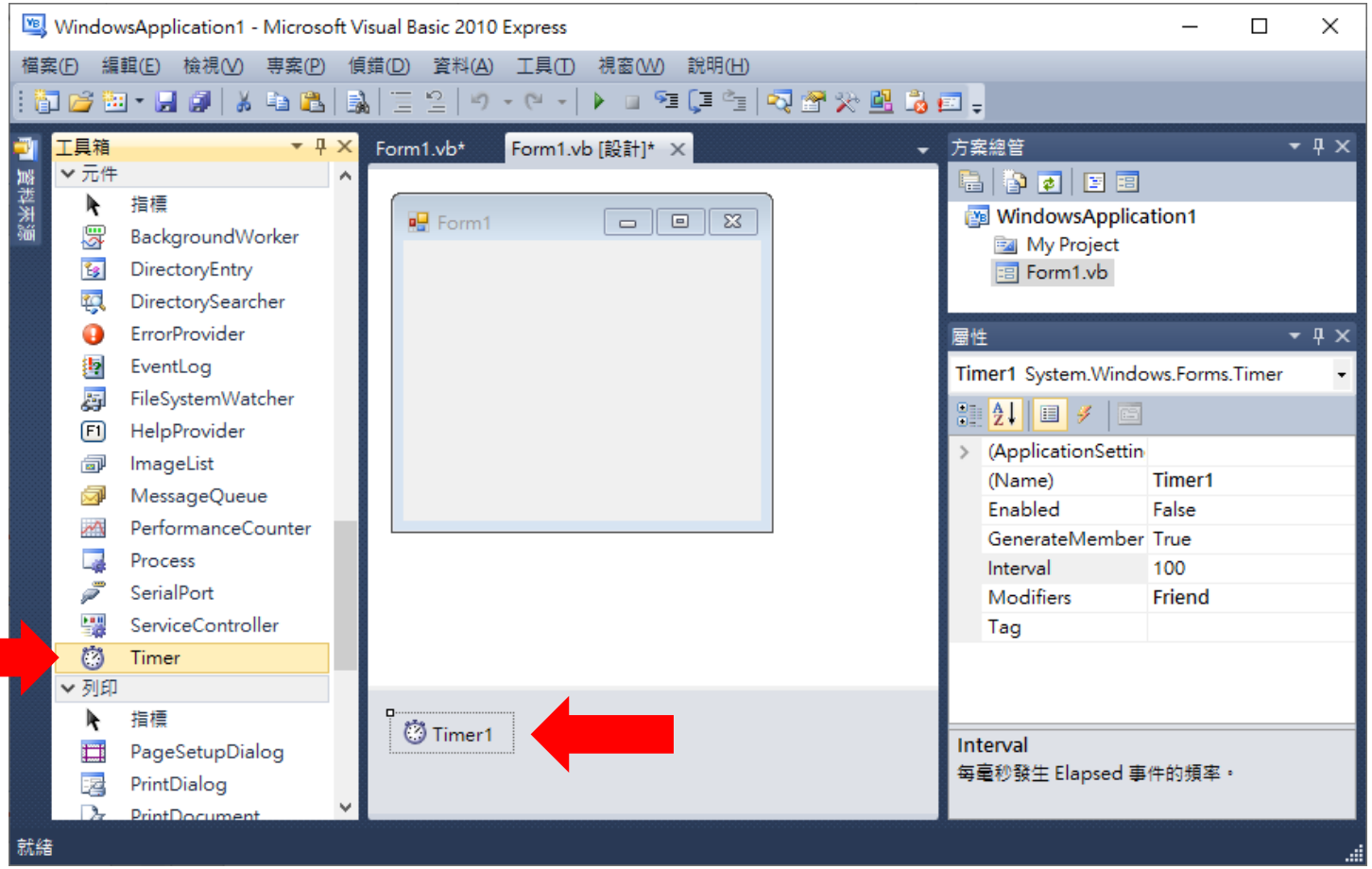
139

計時器(Timer)

- 用來作為與時間有關的功能，例如每隔多少時間自動執行一些事情。
- 常用屬性：
  - Interval：事件觸發的間隔時間(千分之一秒)
  - (若Interval = 1000，表示一秒鐘)
  - 注意：100以下時不一定非常精準
  - Enable：計時器停止或啟動
- 常用事件：
  - Trick：計時器的「滴答」聲，當指定的間隔時間到時觸發一次此事件。

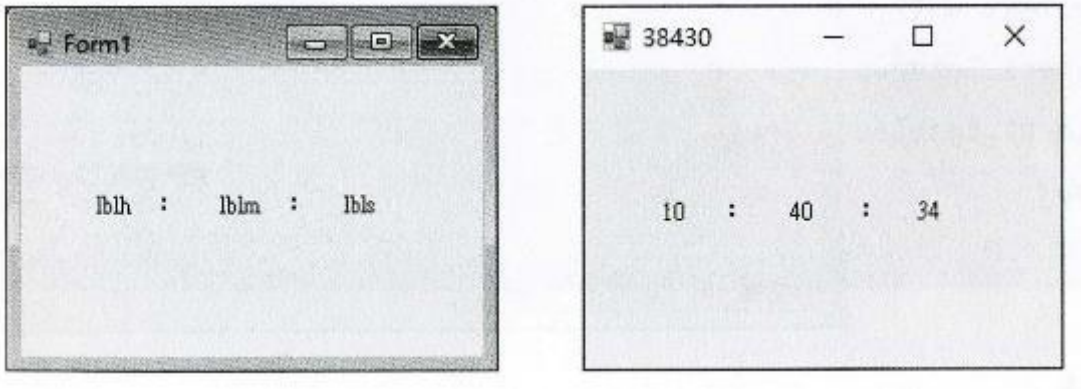
計時器(Timer)

- Timer不會出現在表單上而是在下方



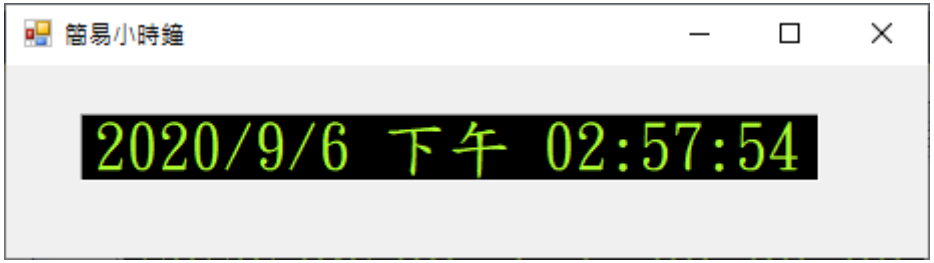
計時器(Timer)

- 練習做一個簡易小時鐘(一)



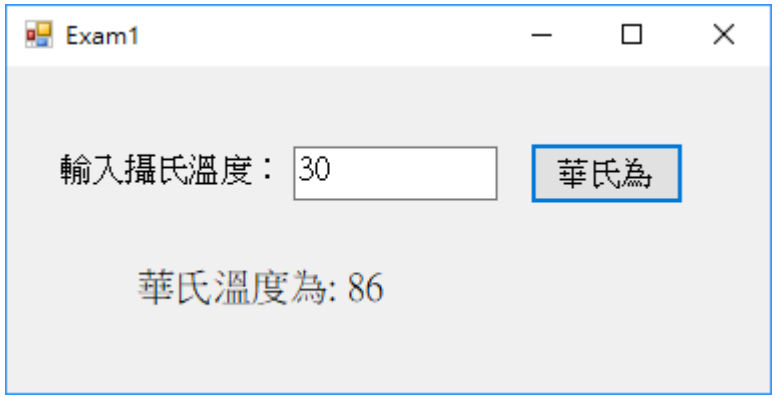
- 練習做一個簡易小時鐘(二)
  - 只需要一行程式碼。

```
Label1.Text = DateTime.Now.ToString()
```



練習

- 設計如下之程式，輸入攝氏溫度，轉換為華氏溫度
- (公式：F = (9.0 * C) /5.0 + 32)



參考作法

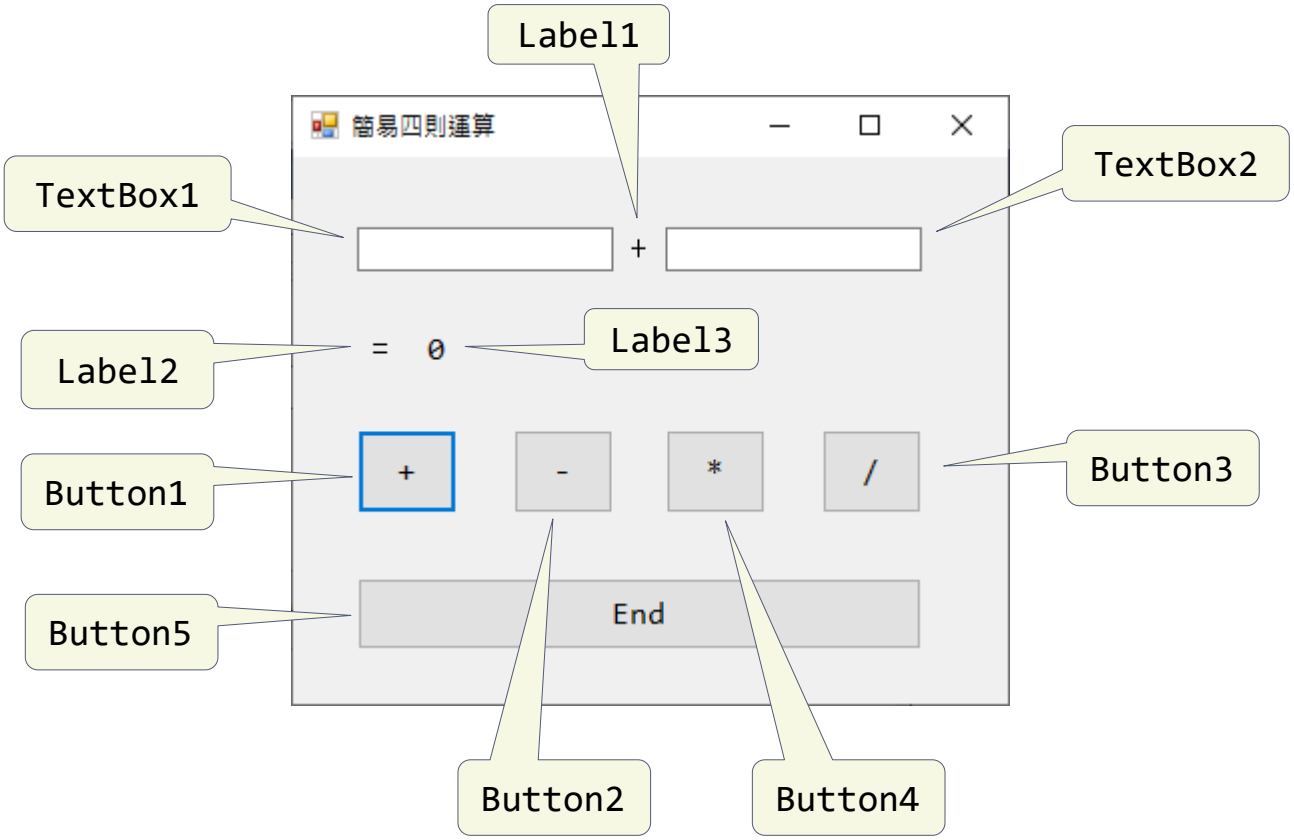
- 程式碼：

```
1   Public Class Form1
        0 個參考
2       Private Sub Button1_Click(sender As Object, e As EventArgs) Handles Button1.Click
3           Dim F As Double
4           F = (9.0 * CSng(TextBox1.Text)) / 5.0 + 32
5           Label2.Text = "華氏溫度為: " & F
6       End Sub
7   End Class
8
```

- TextBox輸入的皆視為字串，要做運算需做適當轉換
- & 運算子串接不同格式的資料格式成為字串
- CSng( )函數是將字元轉換成單精度數值

練習

- 一個簡單的計算機(Page 44)：



練習

- 一個簡單的計算機(Page 44)：

```
Public Class Form1
    Private Sub Button1_Click(sender As Object, e As EventArgs) Handles Button1.Click
        Label1.Text = "+"
        Label3.Text = Str(Val(TextBox1.Text) + Val(TextBox2.Text))
    End Sub

    Private Sub Button2_Click(sender As Object, e As EventArgs) Handles Button2.Click
        Label1.Text = "-"
        Label3.Text = Str(Val(TextBox1.Text) - Val(TextBox2.Text))
    End Sub

    Private Sub Button4_Click(sender As Object, e As EventArgs) Handles Button4.Click
        Label1.Text = "*"
        Label3.Text = Str(Val(TextBox1.Text) * Val(TextBox2.Text))
    End Sub

    Private Sub Button3_Click(sender As Object, e As EventArgs) Handles Button3.Click
        Label1.Text = "/"
        Label3.Text = Str(Val(TextBox1.Text) / Val(TextBox2.Text))
    End Sub

    Private Sub Button5_Click(sender As Object, e As EventArgs) Handles Button5.Click
        End
    End Sub
End Class
```

亂數

- 讓電腦隨機產生一個數值，而且不可預期，如果產生一連串數值，那看起來必須是沒有任何規律的數列
- 是電腦遊戲不可或缺的功能
- 需使用Random類別：
- 先宣告一個變數r：`Dim r As New Random()`
- 然後用下列方式取的亂數：
  - `r.next()` →取得 0~2,147,483,647 之間任一數
  - `r.next(4)` →取得 0~3 之間任一數
  - `r.next(1,42)` →取得 1~41 之間任一數
  - `r.next(-5,6)` →取得 -5~+5 之間任一數

亂數

- 範例：讓電腦產生6個1~49之間的亂數

```
Module Module1
    Sub Main()
        Dim r As New Random()
        Console.WriteLine(r.Next(1, 50))
        Console.WriteLine(r.Next(1, 50))
        Console.WriteLine(r.Next(1, 50))
        Console.WriteLine(r.Next(1, 50))
        Console.WriteLine(r.Next(1, 50))
        Console.WriteLine(r.Next(1, 50))

        Console.Read()  //暫停畫面
    End Sub
End Module
```

亂數

- 亂數是寫遊戲程式重要的功能，我們介紹完基本語法之後再來好好用它。

休息一下

- 超級程式設計師~

基本輸出入

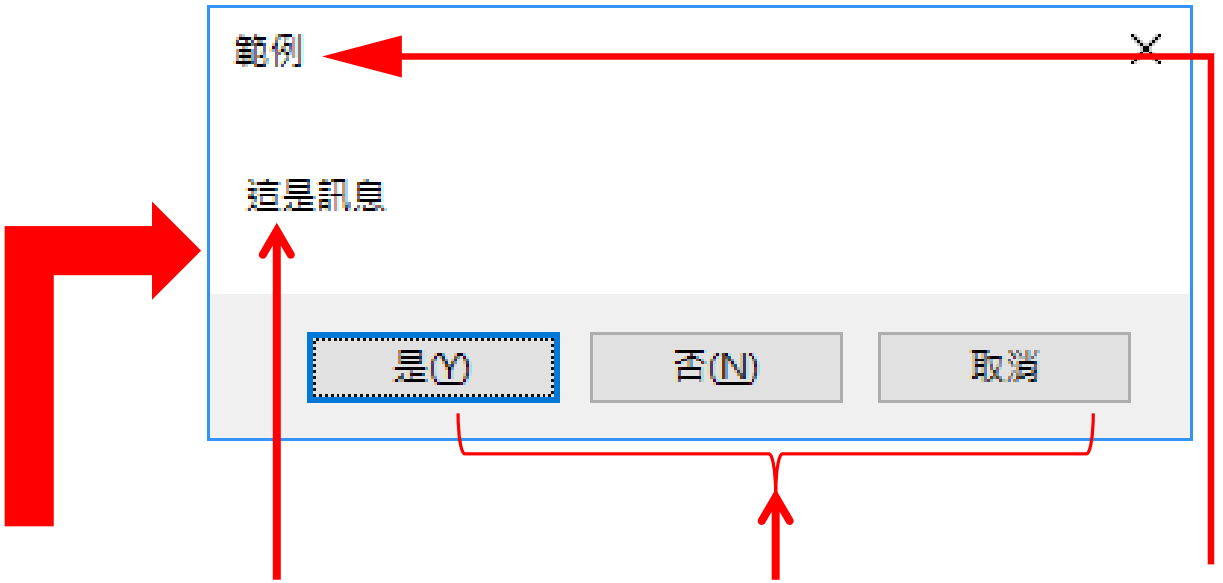
- 程式常需要與使用者互動，有下列基本指令：
- 視窗模式：
- MsgBox( )　自VB6就有的指令，簡單易用
- MessageBox( )　VB2010增加了這個，跟MsgBox( )一樣的功能，但更強一些
- InputBox( )　請使用者輸入訊息

- 命令列模式(前面介紹過了)：
- Console.Write( )
- Console.Read( )

MsgBox( )

- 語法：

MsgBox(“訊息”,樣式,”視窗標題”)



- 例：MsgBox("這是訊息", vbYesNoCancel, "範例")

MsgBox( )

- 關於樣式：

MsgBox(“訊息”,樣式,”視窗標題”)

代碼	常數符號	回復按鈕
0	vbOKOnly	確定
1	vbOKCancel	確定 取消
2	vbAbortRetryIgnore	中止(A) 重試(R) 略過(I)
3	vbYesNoCancel	是(Y) 否(N) 取消
4	vbYesNO	是(Y) 否(N)
5	vbRetryCancel	重試(R) 取消

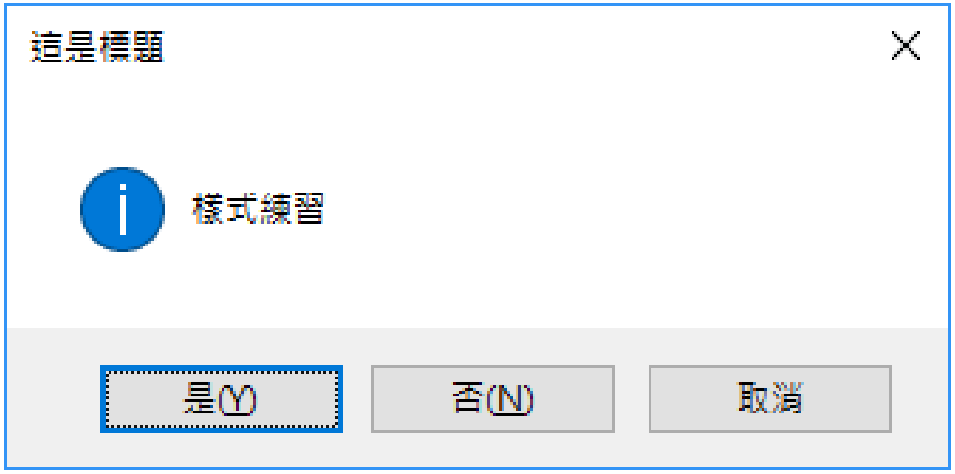
+

Table 10.3

Value	Named Constant	Icon
16	vbCritical	
32	vbQuestion	
48	vbExclamation	
64	vbInformation	

MsgBox( )練習

- 請顯示下列訊息框




```
MsgBox("樣式練習", vbYesNoCancel + vbInformation, "這是標題")
```

MsgBox( )傳回值

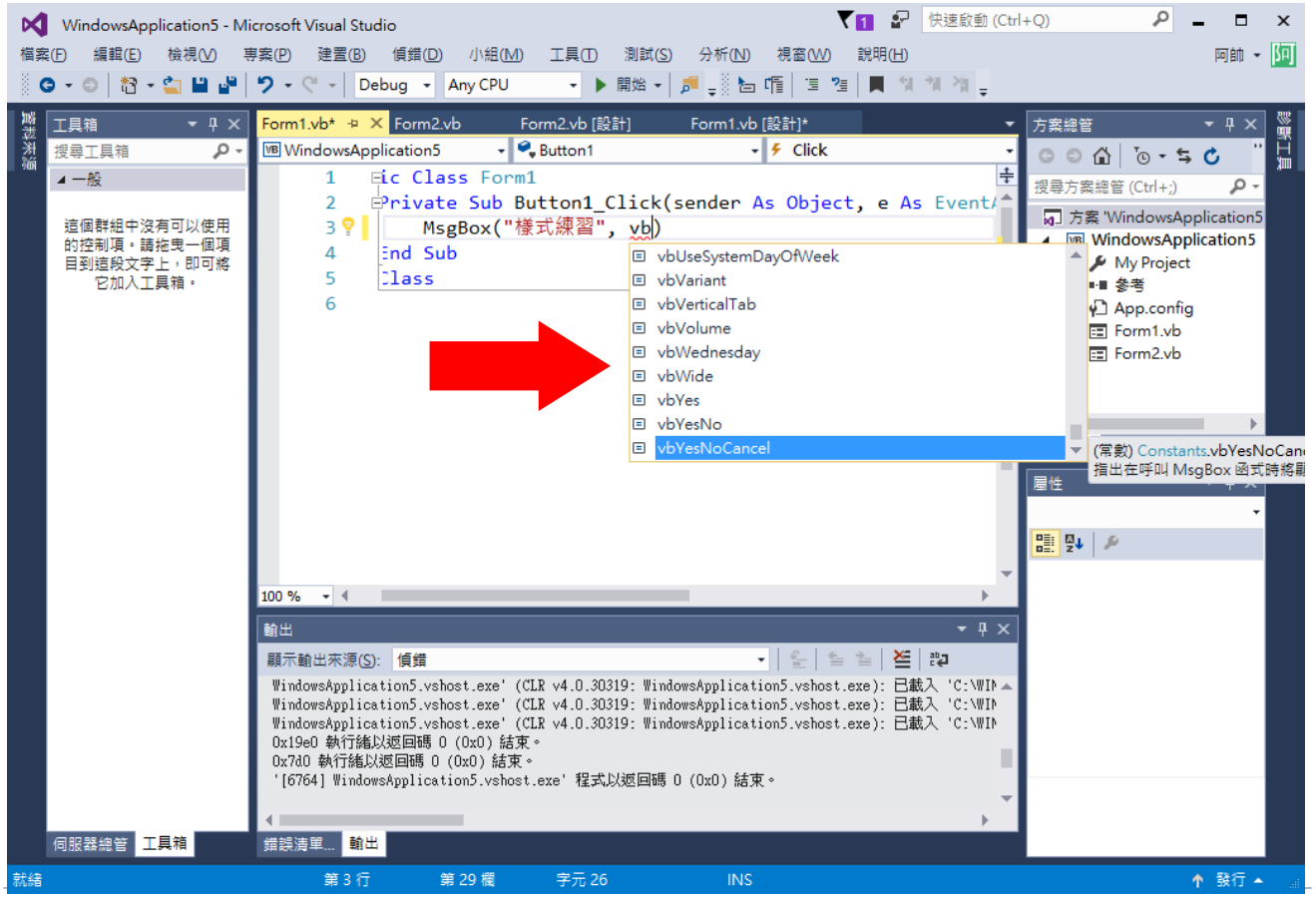
- 那怎麼知道使用者是按了哪個件呢?
- MsgBox( )會傳回一個數值來表示使用者按了哪個鍵

Value	Named Constant	Button Clicked
1	vbOk	Ok button
2	vbCancel	Cancel button
3	vbAbort	Abort button
4	vbRetry	Retry button
5	vbIgnore	Ignore button
6	vbYes	Yes button
7	vbNo	No button

- 例如：如果傳回 1 或 vbOk，就表示使用者按了OK這個按鈕

貼心的VB環境

- 系統在幫得上忙的地方就會自動跳出視窗提示你可用的選項



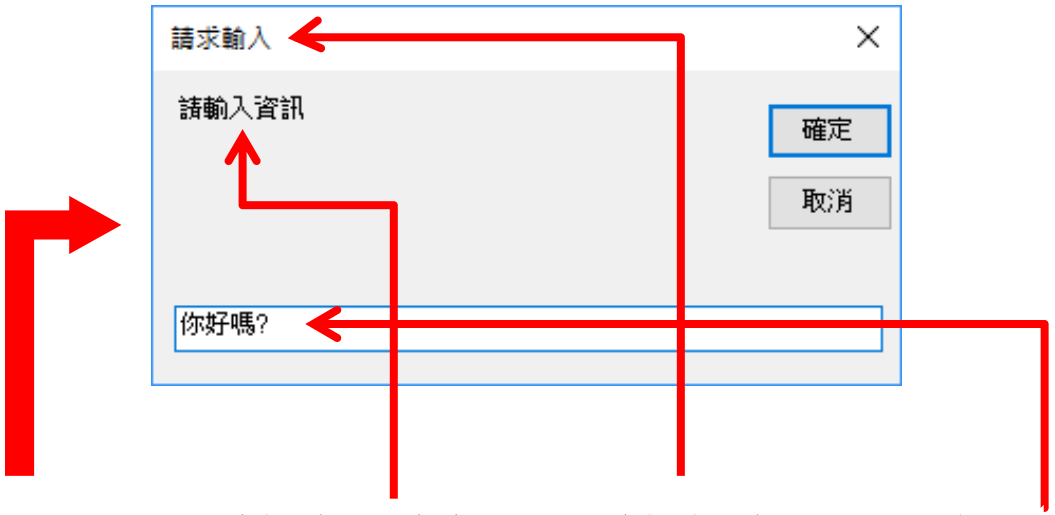
MessageBox( )

- 與MsgBox( )相同，請參考課本
- 略過

InputBox( )

- 請使用者輸入訊息，語法：

```
InputBox(“提示”, ”標題”, 預設值)
```



- 例： `inStr = InputBox("請輸入資訊", "請求輸入", "你好嗎?")`
- 由一個變數來接收使用者的輸入

InputBox( )

- 從InputBox( )輸入的資料一律為“字串”，所以如果是要求輸入數值的話要自己轉換一下

- 很方便的一個指令，但也可以直接在表單上設計好，讓使用者經由TextBox元件輸入資訊

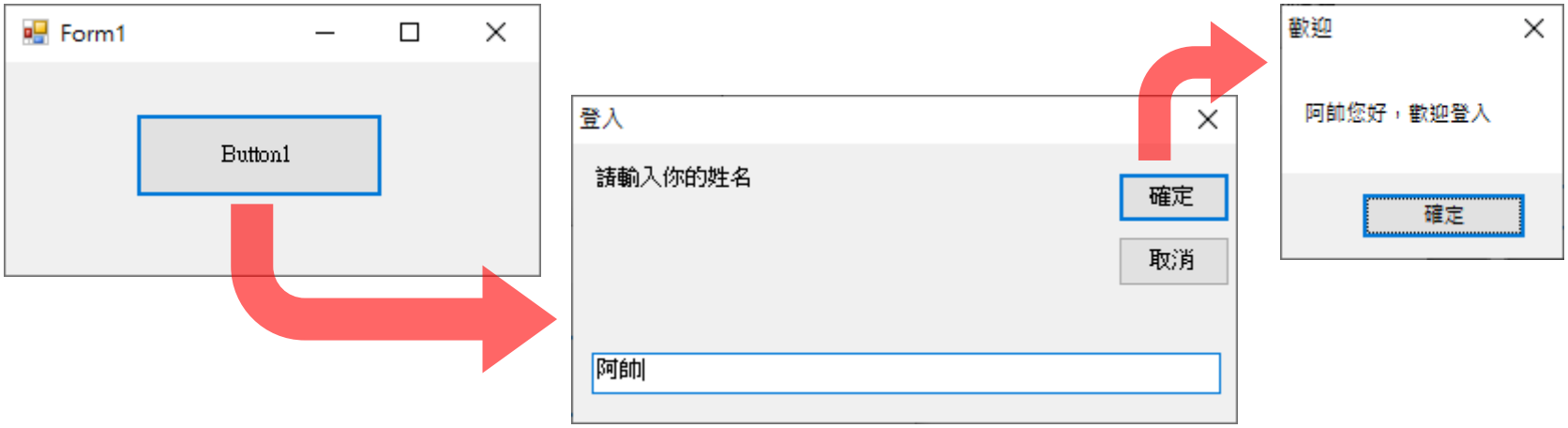
練習

- 以InputBox( )指令請使用者輸入姓名，以MsgBox( )顯示輸入的姓名加上”你好，” 。

- (程式的各項提示及與使用者良好的溝通是很重要的，適時提供訊息讓使用者知道下一步要做甚麼)

練習

- 參考程式：

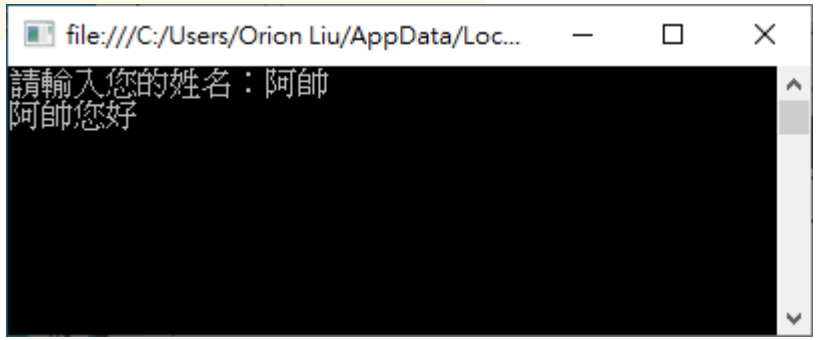



```
Public Class Form1
    Private Sub Button1_Click(ByVal sender As System.Object, ByVal e As . . .
        Dim name As String
        name = InputBox("請輸入你的姓名", "登入", "None")
        MsgBox(name & "您好，歡迎登入", vbOKOnly, "歡迎")
    End Sub
End Class
```

練習

- 以Console.readline( )指令請使用者輸入姓名，Console.Writeline( )顯示輸入的姓名加上”你好”。
- 參考程式：

```
Module Module1
    Sub Main()
        Dim name As String
        Console.Write("請輸入您的姓名：")
        name = Console.ReadLine()
        Console.WriteLine(name & "您好")
        Console.ReadLine()  '暫停畫面，不然會一閃而逝
    End Sub
End Module
```



程式語言的三種控制結構

- 1.循序結構
  - 由上往下，依序執行
- 2.選擇結構
  - 根據條件成立與否來選擇要執行的路徑
- 3.重複結構
  - 利用迴圈重複執行一個區塊

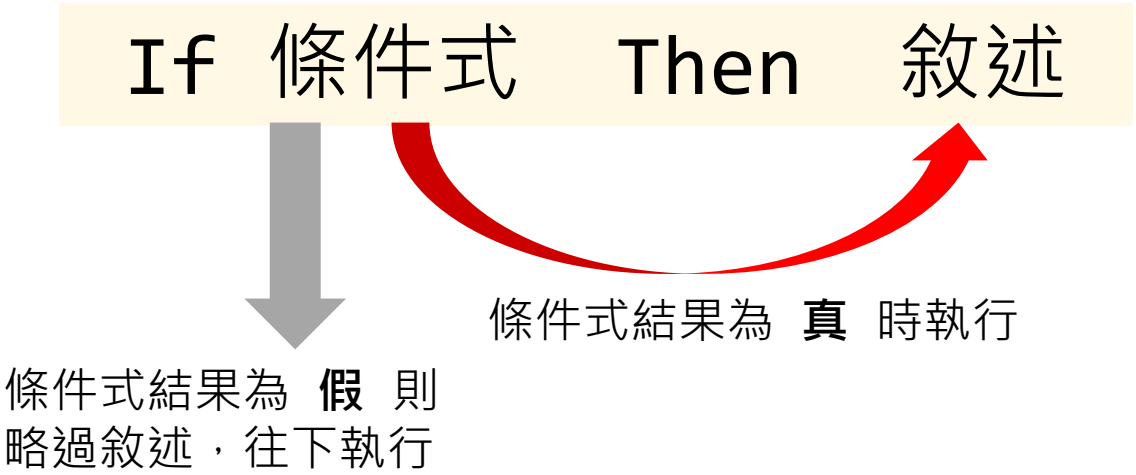
- 沒有具備這三種結構的就不能稱為程式語言(HTML就不算是正式的程式語言，因為它沒有選擇和重複的結構指令)

選擇結構

- 如果.....

If... Then指令

- 如果... 那就...
- 語法：



- 例： If A > B Then A = 10

If... Then指令

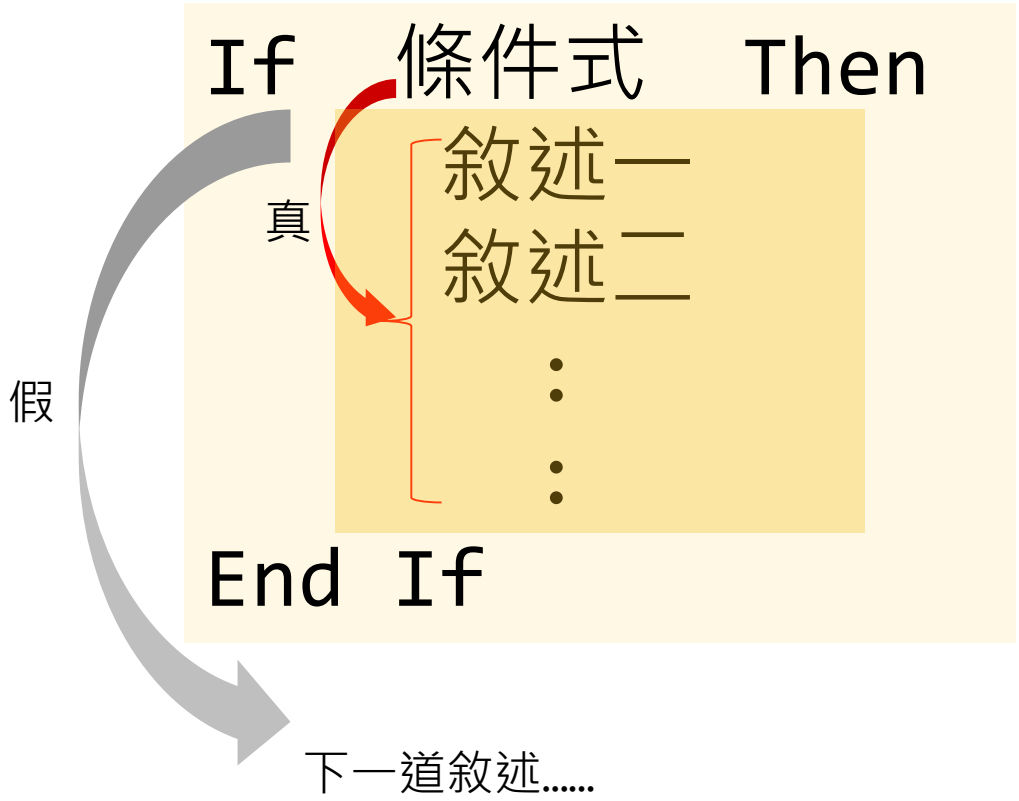
- 例：

```
Dim A As Integer = 300
Dim B As Integer = 150
If A > 200 Then A = A * 0.8
If B > 200 Then B = B * 0.8
```

- 執行結果：
  - A 為 240
  - B 為 150
  - (為什麼?)

If... Then... End If 指令

- 如果條件成立需要執行多行敘述，則可以寫成：



If... Then... End If 指令

- 例：

```
Dim A, B, C As Intege
Dim X As Integer = 10
A = 5 : B = 5 : C = 5
If X >= 10 Then
    A = A + 1
    B = B + 1
    C = C + 1
End If
```

- 執行結果：A, B, C 的值各被加 1

If... Then... End If 指令

- 例：

```
Dim A, B, C As Intege
Dim X As Integer = 10
A = 5 : B = 5 : C = 5
If X > 10 Then
    A = A + 1
    B = B + 1
    C = C + 1
End If
```

- 執行結果：A, B, C 維持原值（Why?）

If... Then... Else 指令

- 如果... 那就... 否則...
- 條件式成立或不成立都有敘述要做
- 語法：

If... Then... Else 指令

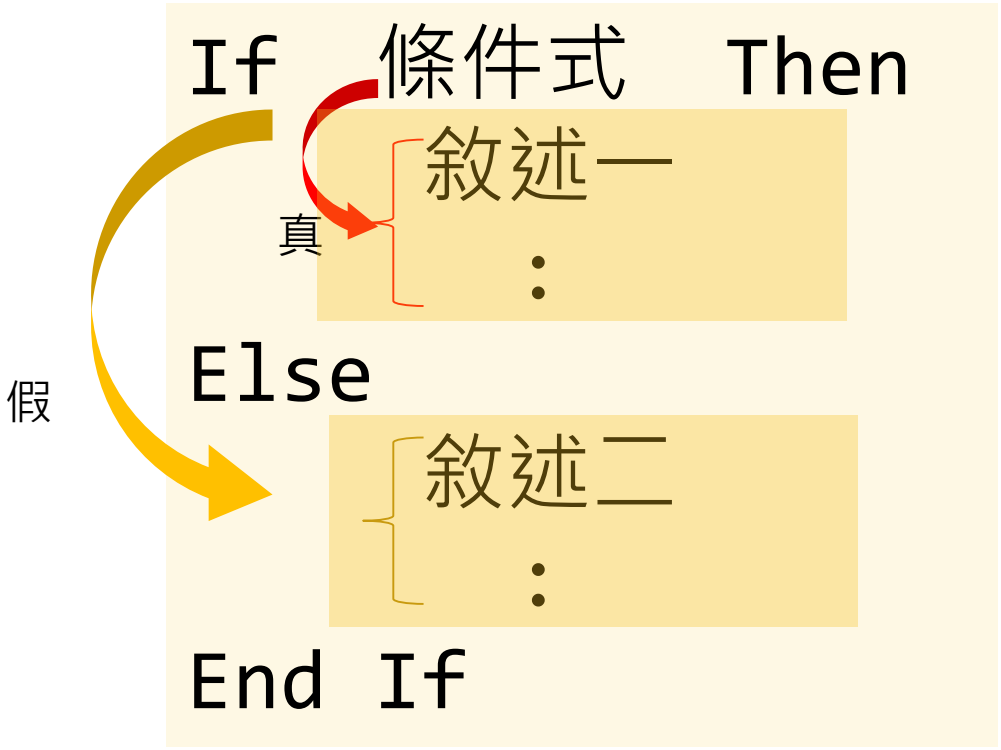
- 例：

```
Dim A As Integer = 300
If A>500 Then A = A*0.8 Else A = A*0.9
```

- 執行結果：270值（Why?）

If... Then... Else 指令

- 多行式寫法：

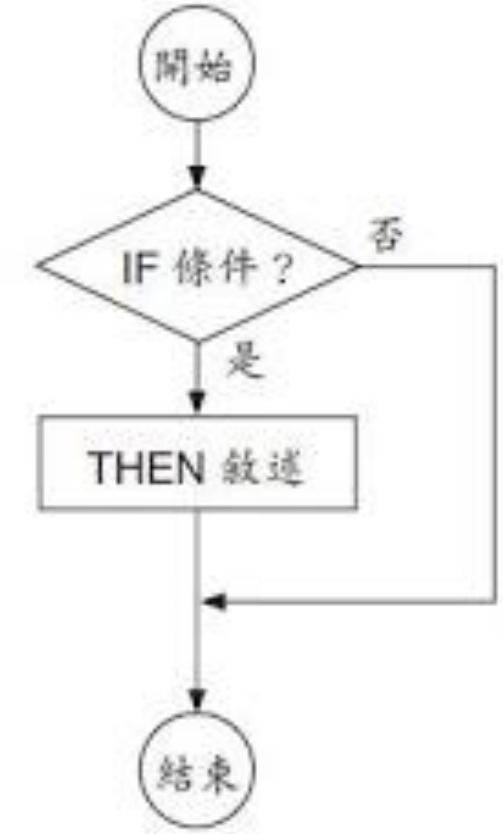


If... Then... Else 指令

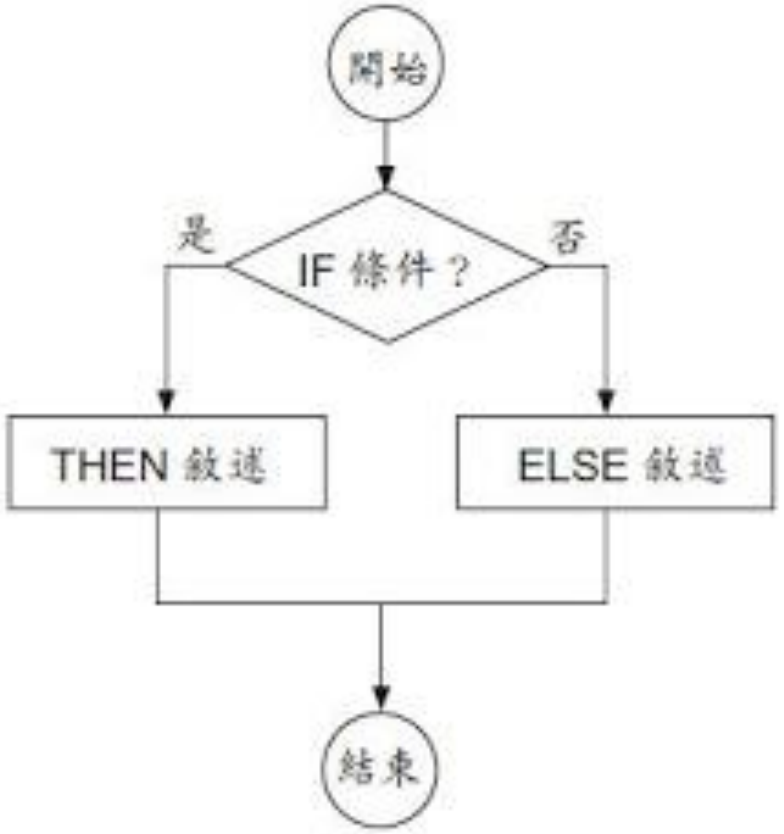
- 多行式寫法例：

```
Dim A As Integer = 300
If A>500 Then
    A = A*0.8
Else
    A = A*0.9
End If
```

If 指令流程圖



IF...THEN 敘述



IF...THEN...ELSE 敘述

If 練習

- 1.輸入一數值,若大於等於60則印出”及格”,否則印出”不及格”
- 2.輸入兩數值,印出較大者
- 3.輸入三數值,印出最大者
- 4.判斷PH酸鹼值,若輸入小於7則印出”酸性”,若等於7則印出”中性”,若大於7則印出”鹼性”

參考寫法

- 題目1(命令列模式)：

```
Dim x As Integer
Console.Write("請輸入數值: ")
x = Val(Console.ReadLine())
If x >= 60 Then
    Console.WriteLine("及格 “)
Else
    Console.WriteLine("不及格 “)
End If
Console.Read()
```

參考寫法

- 題目1(視窗模式)：

```
Dim x As Integer
x = Val(InputBox("請輸入數值: "))
If x >= 60 Then
    MsgBox("及格")
Else
    MsgBox("不及格")
End If
```

參考寫法

- 題目2(命令列模式)：

```
Dim a, b As Integer
Console.Write("輸入第一個數值: ")
a = Val(Console.ReadLine())
Console.Write("輸入第二個數值: ")
b = Val(Console.ReadLine())
If a > b Then
    Console.WriteLine(a)
Else
    Console.WriteLine(b)
End If
Console.Read()
```

參考寫法

- 題目2(視窗模式)：

```
Dim a, b As Integer
a = Val(InputBox("輸入第一個數值: "))
b = Val(InputBox("輸入第二個數值: "))
If a > b Then
    MsgBox(a & "比較大")
Else
    MsgBox(b & "比較大")
End If
```

參考寫法

- 題目3(命令列模式)：

```
Dim A, B, C, MAX As Integer
Console.Write("請輸入第一個數值: ")
A = Val(Console.ReadLine())
Console.Write("請輸入第二個數值: ")
B = Val(Console.ReadLine())
Console.Write("請輸入第三個數值: ")
C = Val(Console.ReadLine())
If A > B Then Max = A Else Max = B
If MAX > C Then
    Console.WriteLine(MAX)
Else
    Console.WriteLine(C)
End If
Console.Read()
```

參考寫法

- 題目3(視窗模式)：

```
Dim A, B, C, MAX As Integer
A = Val(InputBox("請輸入第一個數值: "))
B = Val(InputBox("請輸入第二個數值: "))
C = Val(InputBox("請輸入第三個數值: "))
If A > B Then MAX = A Else MAX = B
If MAX > C Then
    MsgBox(MAX & "是最大值")
Else
    MsgBox(C & "是最大值")
End If
```

參考寫法

- 題目4(命令列模式)：

```
Dim PH As Single
Console.Write("請輸入酸鹼值: ")
PH = Val(Console.ReadLine())
If PH > 7 Then Console.WriteLine("鹼性")
If PH = 7 Then Console.WriteLine("中性")
If PH < 7 Then Console.WriteLine("酸性")
Console.Read()
```

參考寫法

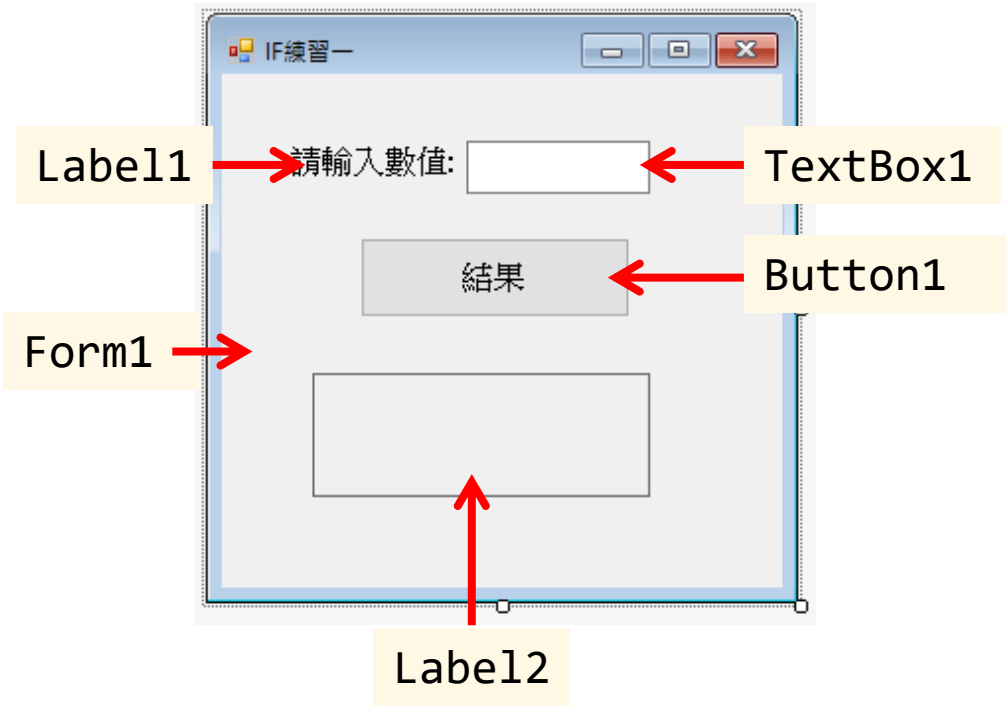
- 題目4(視窗模式)：

```
Dim PH As Single
PH = Val(InputBox("請輸入酸鹼值: "))
If PH > 7 Then MsgBox("鹼性")
If PH = 7 Then MsgBox("中性")
If PH < 7 Then MsgBox("酸性")
```

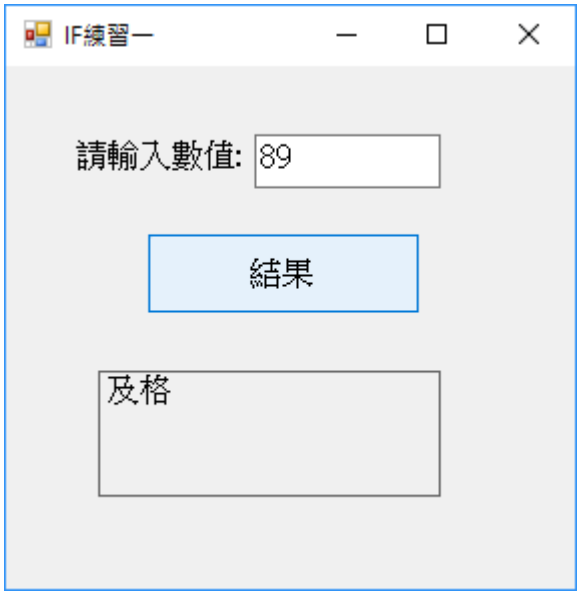
參考寫法

- 題目1(視窗模式)：

設計



執行結果



參考寫法

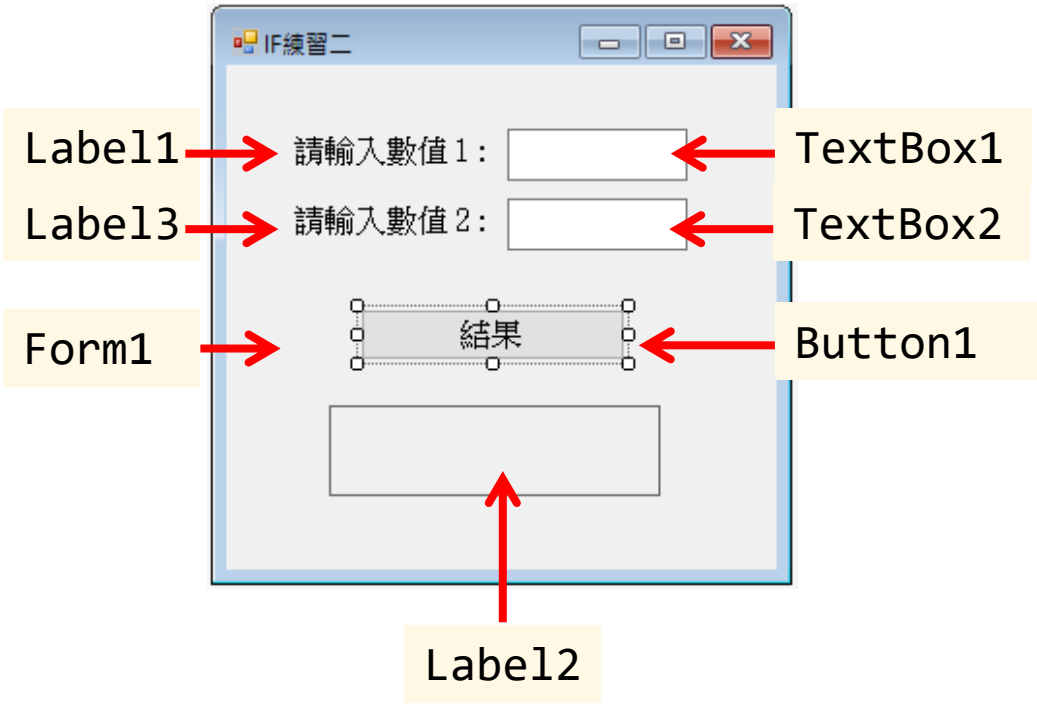
- 題目1(視窗模式)：

```
Private Sub Button1_Click(sender As Object ……
    Dim x As Integer
    x = Val(TextBox1.Text)
    If x >= 60 Then
        Label2.Text = "及格 "
    Else
        Label2.Text = "不及格 "
    End If
End Sub
```

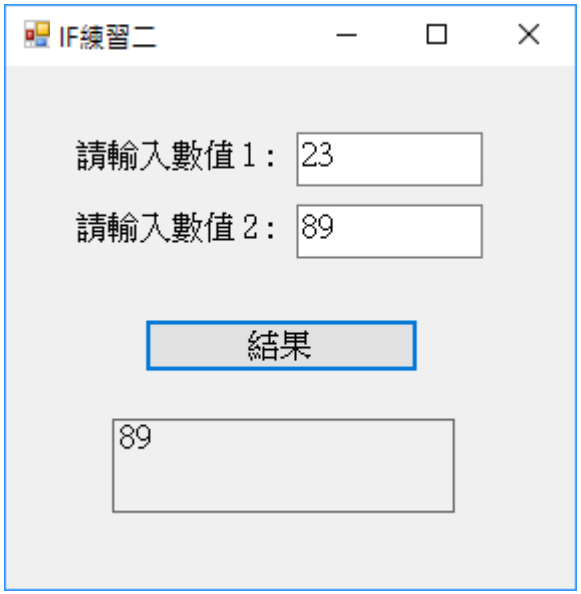
參考寫法

- 題目2(視窗模式)：

設計



執行結果



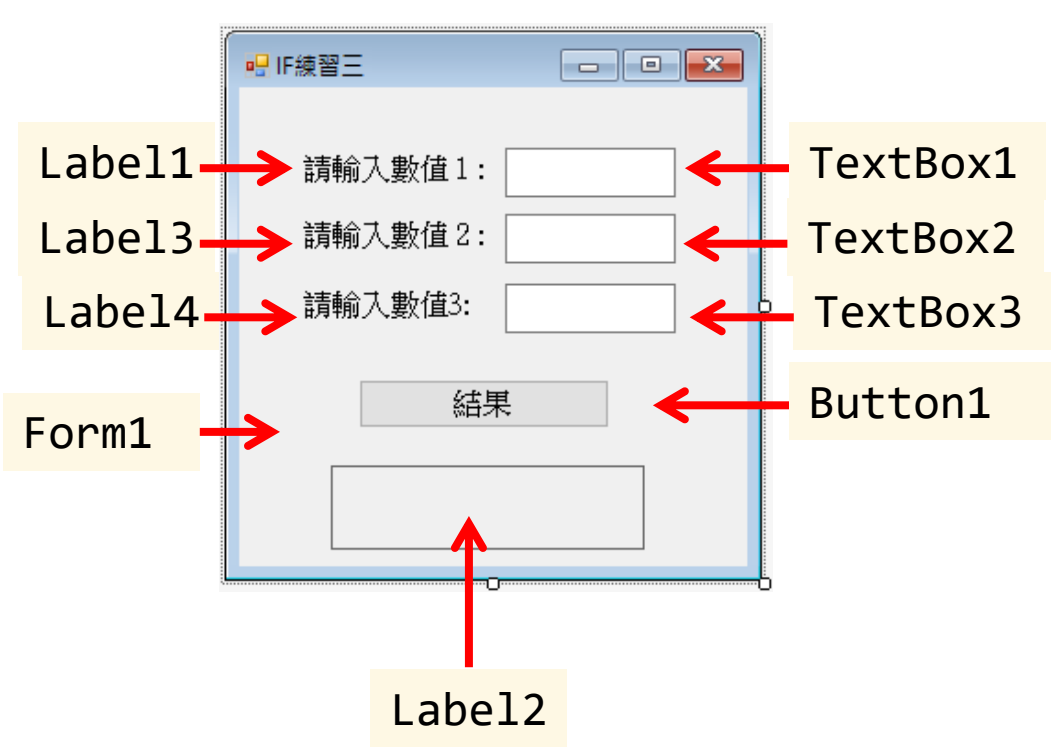
參考寫法

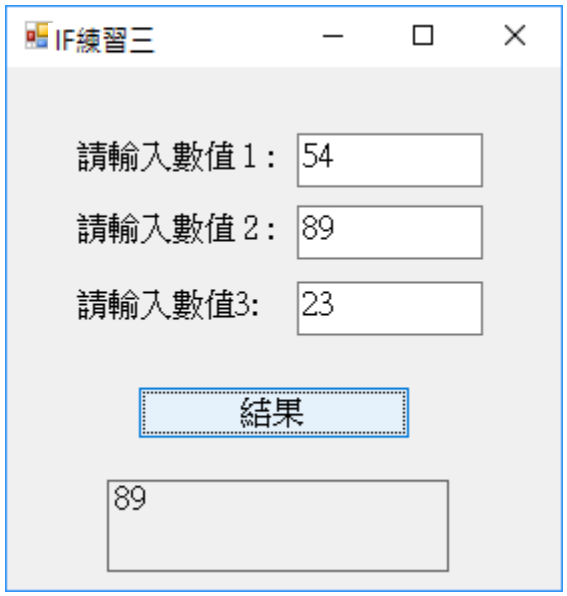
- 題目2(視窗模式)：

```
Private Sub Button1_Click(sender As Object…
    If Val(TextBox1.Text) >= Val(TextBox2.Text) Then
        Label2.Text = TextBox1.Text
    Else
        Label2.Text = TextBox2.Text
    End If
End Sub
```

參考寫法

- 題目3(視窗模式)：
- 設計　　執行結果





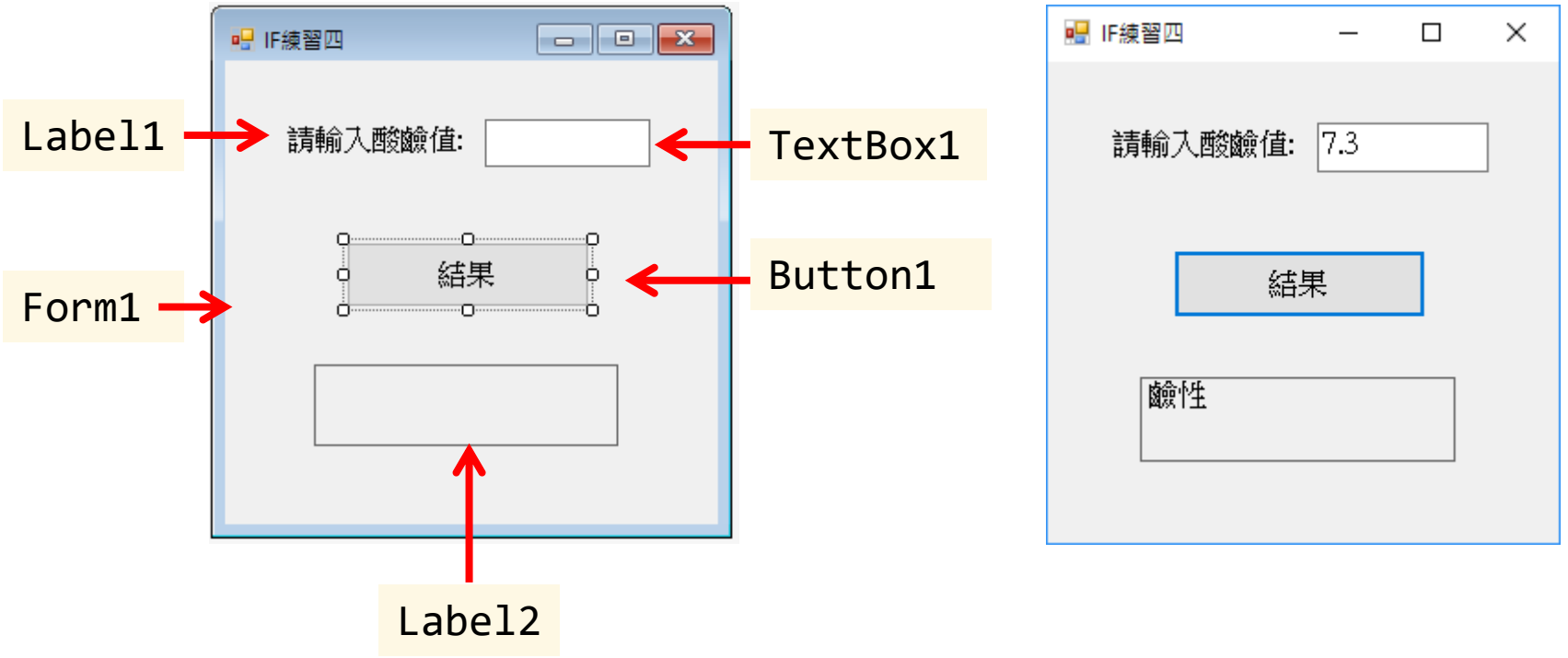
參考寫法

- 題目3(視窗模式)：

```
Private Sub Button1_Click(sender As Object…
    Dim a, b, c, max As Integer
    a = Val(TextBox1.Text)
    b = Val(TextBox2.Text)
    c = Val(TextBox3.Text)
    If a > b Then max = a Else max = b
    If max < c Then max = c
    Label2.Text = max
End Sub
```

參考寫法

- 題目4(視窗模式)：
- 設計　　執行結果



參考寫法

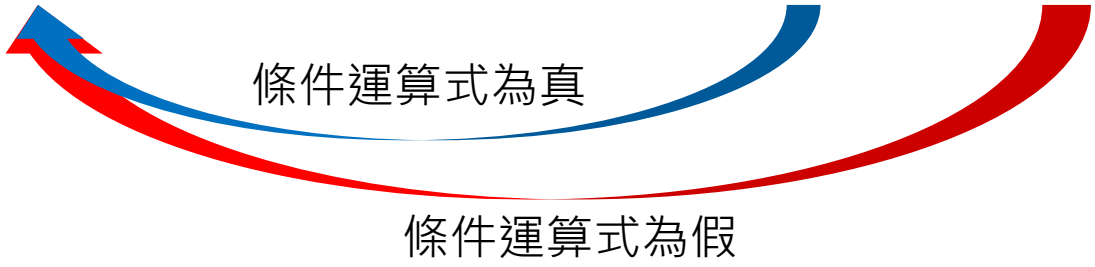
- 題目4(視窗模式)：

```
Private Sub Button1_Click(sender As Object…
    Dim PH As Single
    PH = Val(TextBox1.Text)
    If PH > 7 Then Label2.Text = "鹼性"
    If PH = 7 Then Label2.Text = "中性"
    If PH < 7 Then Label2.Text = "酸性"
End Sub
```

IIF( )條件函數

- 當IF的判斷很單純時，可以使用此方式
- 語法：

```
變數 = IIF(條件運算式，設定值1，設定值2)
```



IIF( )條件函數

- 簡化三數比大小的寫法：

```
Private Sub Button1_Click(sender As
Object…
    Dim a, b, c, max As Integer
    a = Val(TextBox1.Text)
    b = Val(TextBox2.Text)
    c = Val(TextBox3.Text)
    max = IIf( a > b, a, b )
    Label2.Text = IIf( max > c, max, c )
End Sub
```

If...Then...ElseIf...

- 當有多重的判斷時，可以這樣用

```
If … Then
    敘述一
Else
    If … Then
        敘述二
    Else
        If … Then
            敘述三
        Else
            敘述四
        End If
    End If
End If
```

```
If … Then
    敘述一
ElseIf … Then
    敘述二
ElseIf … Then
    敘述三
Else
    敘述四
End If
```

有沒有簡化很多呢?

練習

- 1. 寫一程式，讀入一個數字，若是負數，就將它轉為正數。

- 2. 寫一程式，輸入x和y，如果x和y都是正數，則令z = 1，如果兩者均為負數，則令z = -1，否則令z = 0

參考寫法

- 題目1：

```
Sub Main()
    Dim A As Integer
    Console.Write("請輸入數字：")
    A = Val(Console.ReadLine())
    Console.WriteLine(IIf(A > 0, A, A * -1))

    Console.ReadLine()
End Sub
```

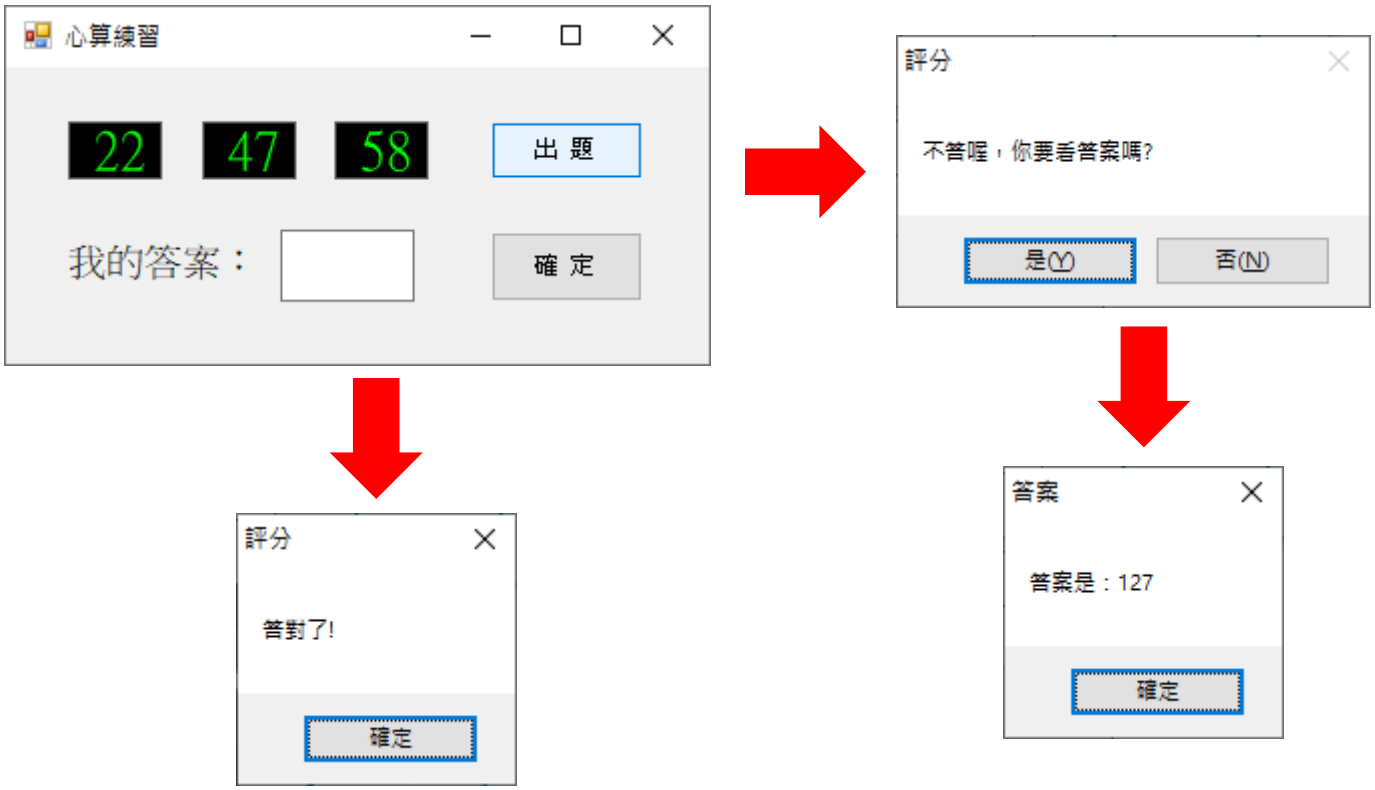
參考寫法

- 題目2：

```
Sub Main()
    Dim A, B, Z As Integer
    Console.Write("請輸入數字：")
    A = Val(Console.ReadLine())
    Console.Write("請輸入數字：")
    B = Val(Console.ReadLine())
    If A >= 0 And B >= 0 Then
        Z = 1
    ElseIf A < 0 And B < 0 Then
        Z = -1
    Else
        Z = 0
    End If
    Console.WriteLine(Z)
    Console.ReadLine()
End Sub
```

練習

- 線上心算練習：請設計一個程式，可以自動產生三個兩位數，讓使用者練習心算加法，電腦評定是否正確。



參考寫法

```
Public Class Form1
    Dim ans As Integer  ‘因為大家都要用到ans，所以要在此宣告

    Private Sub Button1_Click(ByVal sender As System.Object, ByVal e As System.EventArgs) ...
        Dim r As New Random
        Label1.Text = Str(r.Next(100))  ‘產生三個亂數
        Label2.Text = Str(r.Next(100))
        Label3.Text = Str(r.Next(100))
        ans = Val(Label1.Text) + Val(Label2.Text) + Val(Label3.Text)
    End Sub

    Private Sub Button2_Click(ByVal sender As System.Object, ByVal e As System.EventArgs) ...
        Dim user_ans, Y As Integer
        user_ans = Val(TextBox1.Text)
        If user_ans = ans Then
            MsgBox("答對了!", vbOKOnly, "評分")
        Else
            Y = MsgBox("不答喔，你要看答案嗎?", vbYesNo, "評分")
            If Y = vbYes Then
                MsgBox("答案是：" & ans, vbOKOnly, "答案")
            End If
        End If
    End Sub
End Class
```

- 試試看如果再加上計時限制呢?

選擇結構

- 多重選擇：

Select Case 判斷式

- 當需要多重判斷時，還有比If ... ElseIf ... 更簡潔的方式，就是用Select Case，語法：

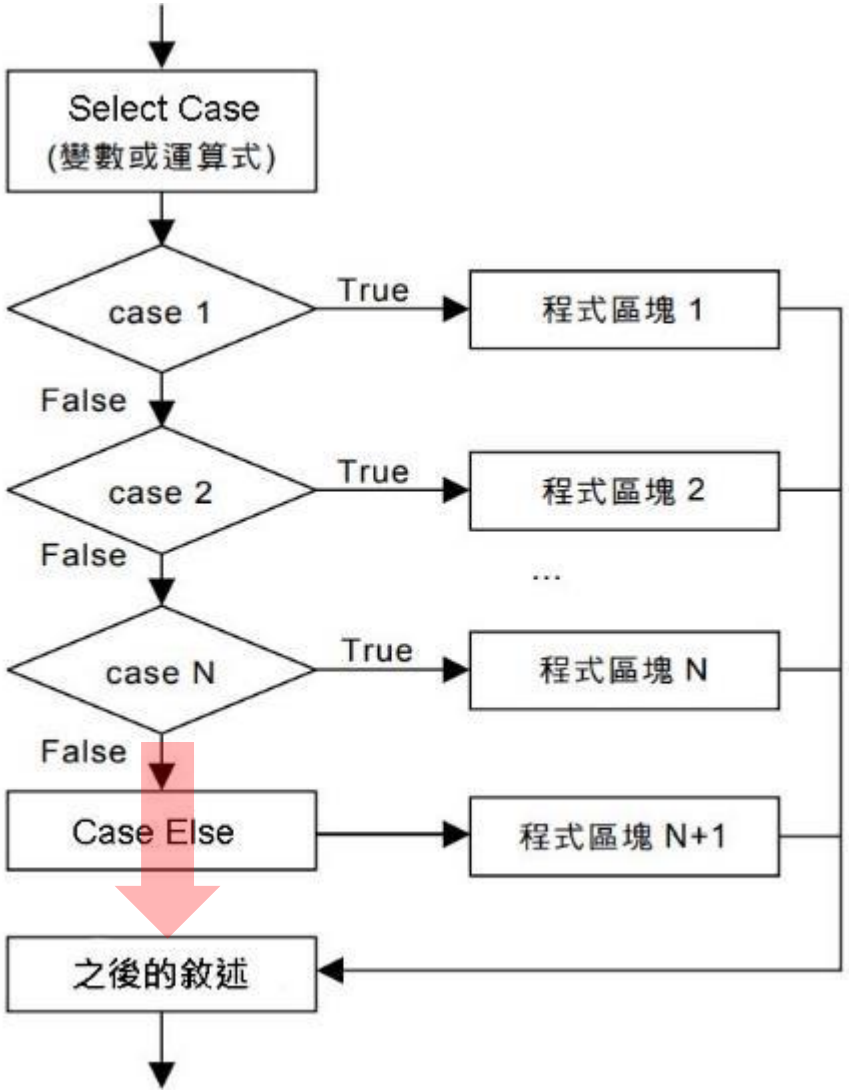
```
Select Case 變數
    Case 條件式一
        敘述一
    Case 條件式二
        敘述二
        :
    Case Else
        敘述
End Select
```

可以有很多Case
第一個符合條件的
Case 會被執行，之
後的都忽略

Case Else可省略

Select Case 判斷式

流程圖



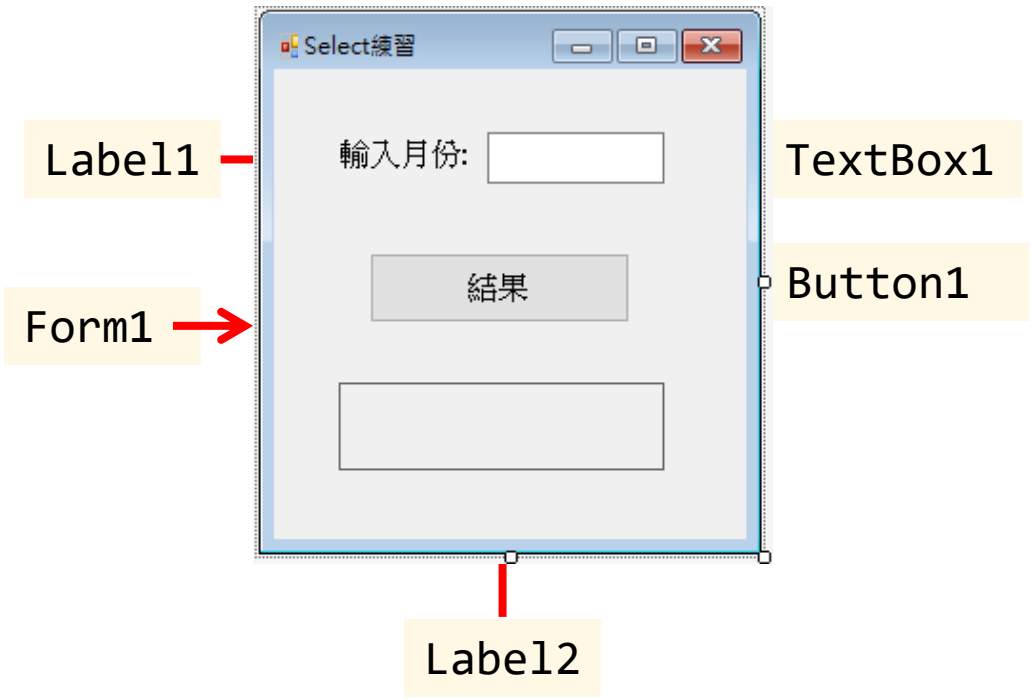
只有一個程式區塊會被執行到

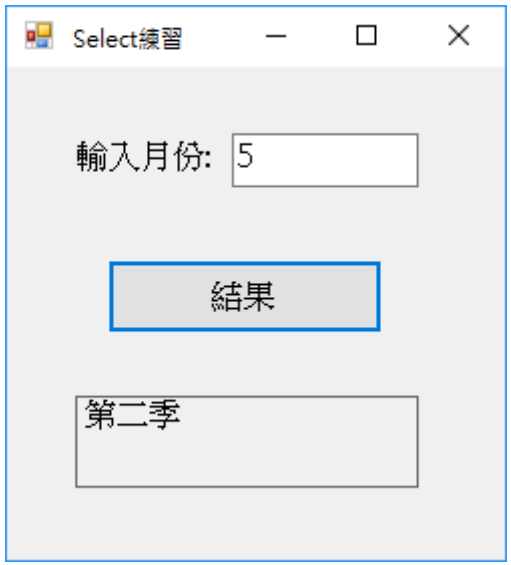
若所有Case皆不成立，也沒有Case Else，則直接離開，沒有任何程式區塊會被執行

若所有Case皆不成立，則執行Case Else的區塊

Select Case 判斷式

- 輸入月份，顯示是第幾季：
- 設計　　執行結果





Select Case 判斷式

- 參考寫法：

```
Dim M As Integer
M = TextBox1.Text
Select Case M
    Case 1, 2, 3
        Label2.Text = "第一季"
    Case 4 To 6
        Label2.Text = "第二季"
    Case 7
        Label2.Text = "第三季"
    Case 8, 9
        Label2.Text = "第三季"
    Case 10 To 12
        Label2.Text = "第四季"
    Case Is > 12
        Label2.Text = "一年只有12個月"
    Case Else
        Label2.Text = "你輸入了什麼?"
End Select
```

Select Case 判斷式

- Case可接受的運算式：

Case運算式	說明
`Case 100`	直接寫出數值
`Case “ABC”`	直接寫出字元或字串
`Case 1 To 5`	是否介於某一範圍
`Case 1, 2, 3`	個別列舉
`Case Is > 100`	是否大或小於某一值

Select Case練習

- 1.依據氣象局規定：風速每小時低於62公里時為熱帶低氣壓,達62~117公里為輕度颱風，118~183公里為中度颱風，184~220公里為強烈颱風，超過220公里則為超級颱風,寫一程式，依輸入風速來判定為何種等級的台風。
-
- 2.寫一程式判斷成績等第(90~100分為優等，80~89分為甲等，70~79為乙等，60~69分為丙等，60分以下為丁等)

Select Case練習

- 題1.參考寫法：

```
W = Val(TextBox1.Text)
Select Case W
    Case Is < 62
        Label2.Text="熱帶低氣壓"
    Case 62 To 117
        Label2.Text="輕度颱風"
    Case 118 To 183
        Label2.Text="中度颱風"
    Case 184 To 220
        Label2.Text="強烈颱風"
    Case Is > 220
        Label2.Text="超級颱風"
End Select
```

Select Case練習

- 題2.參考寫法：

```
S = Val(TextBox1.Text)
Select Case S
    Case 90 To 100
        Label2.Text="優等"
    Case 80 To 89
        Label2.Text="甲等"
    Case 70 To 79
        Label2.Text="乙等"
    Case 60 To 69
        Label2.Text="丙等"
    Case Is < 60
        Label2.Text="丁等"
End Select
```

Select Case注意事項

- 如果S輸入100，雖然所有Case都符合條件，但只有第一個符合的會被執行，其他敘述會被忽略

```
S = Val(TextBox1.Text)
Select Case S
    Case 100
        Label2.Text="優等"
    Case 90 To 100
        Label2.Text="甲等"
    Case Is > 60
        Label2.Text="乙等"
End Select
```

重複結構

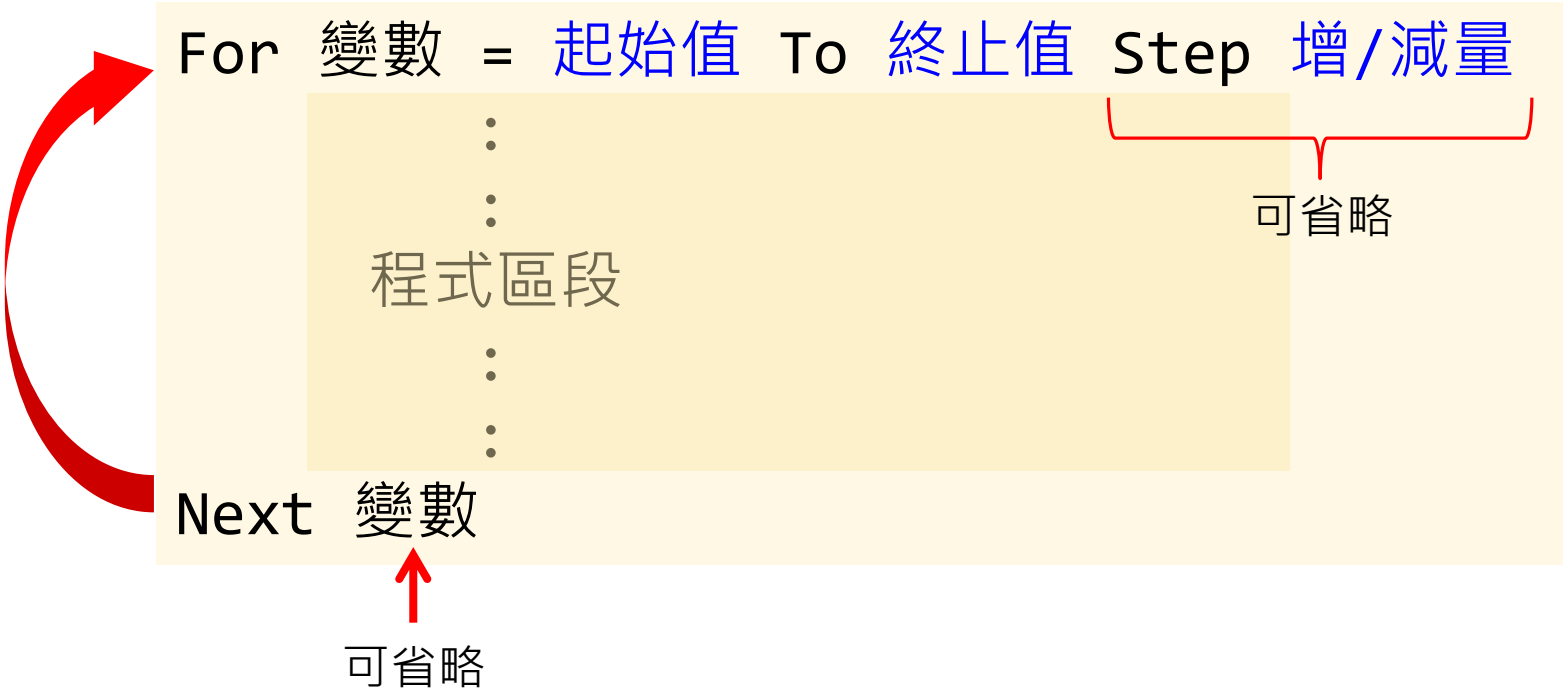
- 重複執行某個區段，又稱迴圈，一共有四種：
- For
- While
- Do loop
- For each(於陣列時介紹)

重複結構

For

For … Next敘述

- 讓程式區段依照給定的起始值及終止值，重複執行多次，語法：

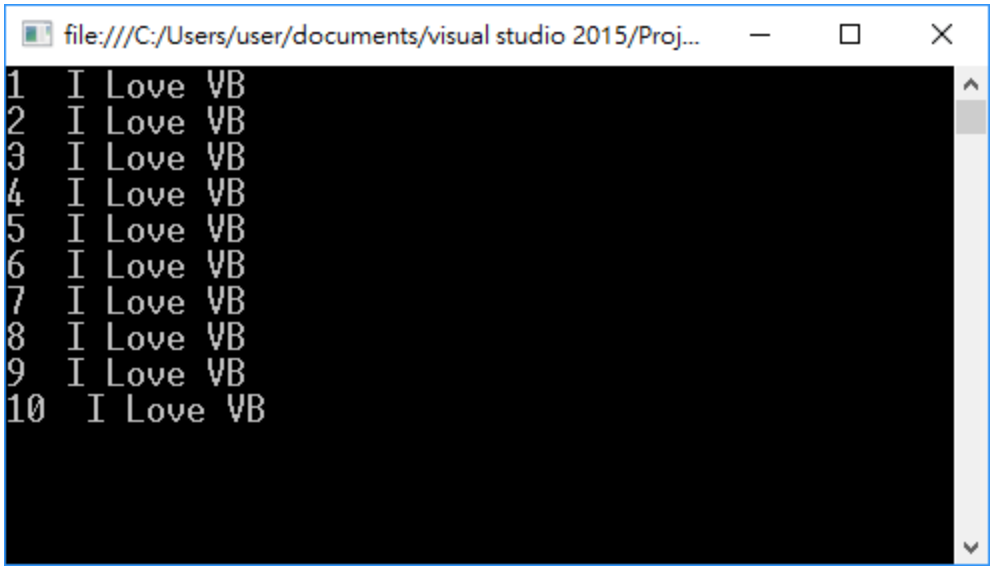


For … Next敘述

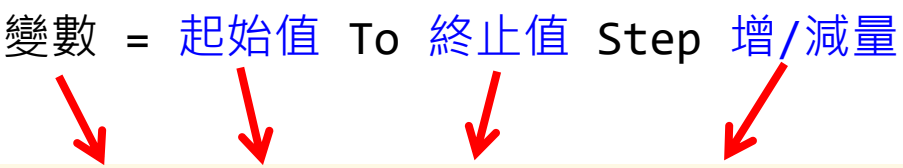
- 範例練習，印出10次”I Love VB” ：

```
For I = 1 To 10 Step 1
    Console.WriteLine(I & “I Love VB”)
Next I
```

- 執行結果：



For … Next敘述




```
For I = 1 To 10 Step 1
    Console.WriteLine(I & “ I Love VB”)
Next I
```

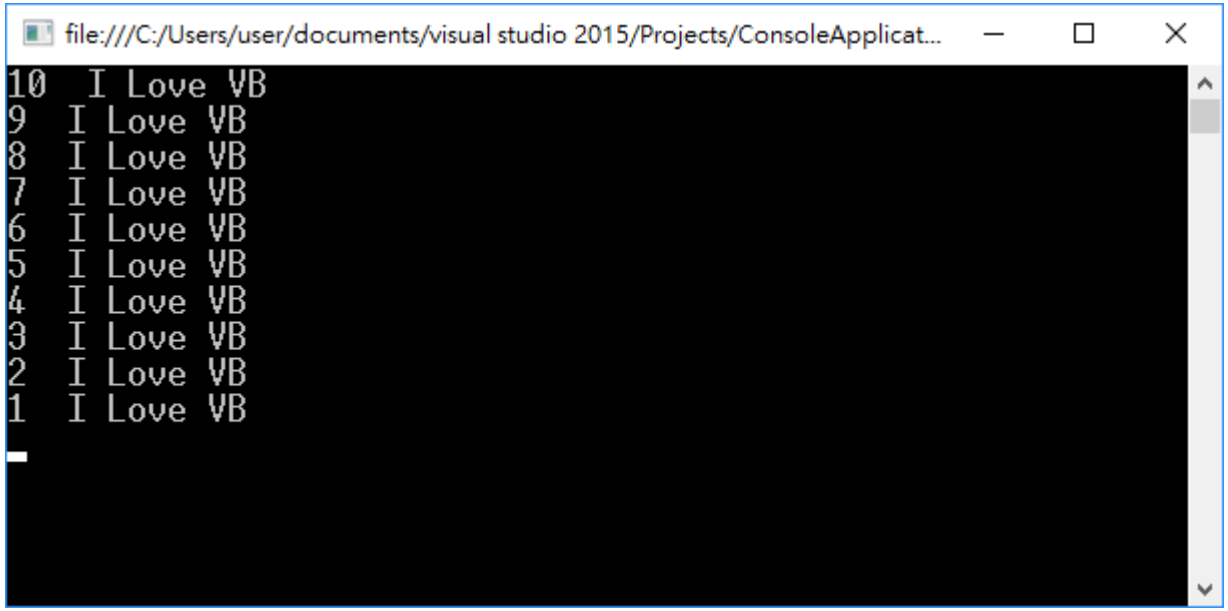
- I = 1時，印一行 I Love VB
- I = 2時，印一行 I Love VB
- ：
- I = 10時，印一行 I Love VB
- 一共印出10行

For … Next敘述

- 也可以用倒數的：

```
For I = 10 To 1 Step -1
    Console.WriteLine(I & “ I Love VB”)
Next I
```

- 執行結果：



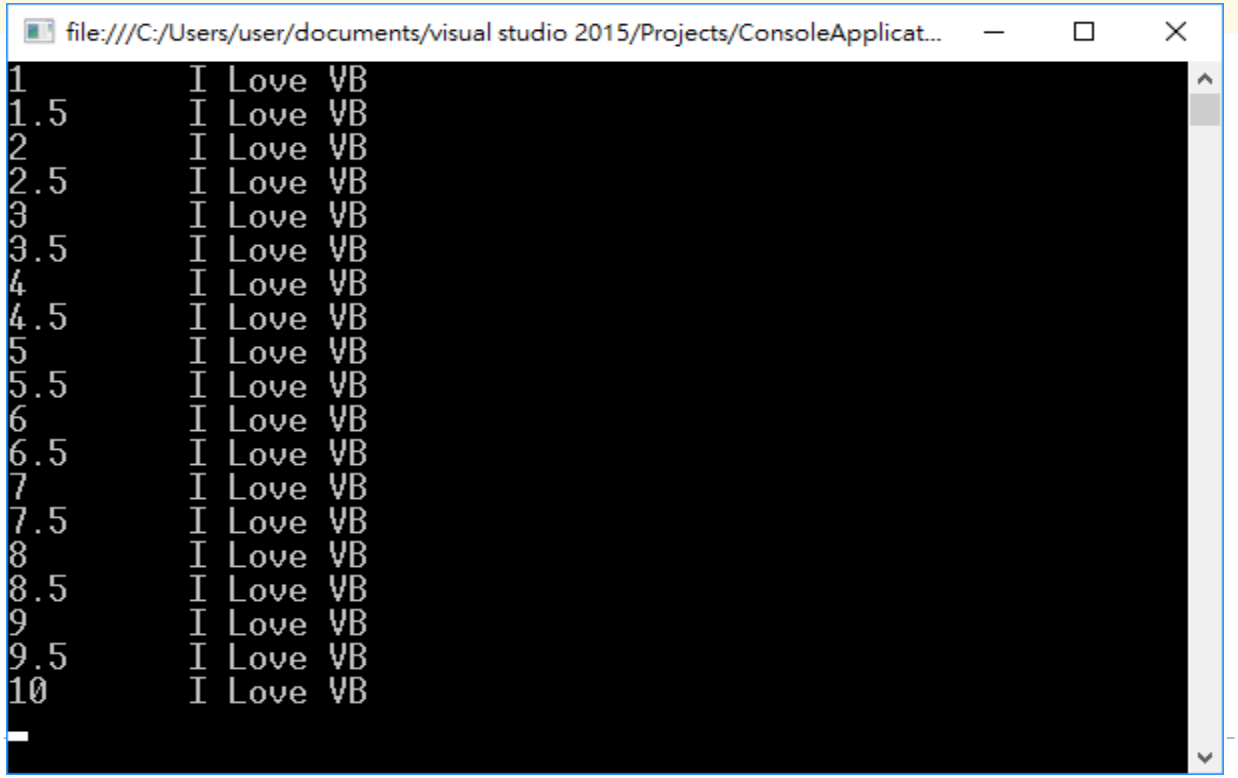
For … Next敘述

- 增/減量也可以不是整數：

插入一個跳格(Tab)

```
For I = 1 To 10 Step 0.5
    Console.WriteLine(I & vbTab & “ I Love VB”)
Next I
```

- 執行結果：(20次)



For … Next敘述

- 增/減量是+1時可省略，Next後面的變數可省略：

```
For I = 1 To 10 Step 1
    Console.WriteLine(I & “ I Love VB”)
Next   I
```

```
For I = 1 To 10  (省略)
    Console.WriteLine(I & “ I Love VB”)
Next (省略)
```

在多重迴圈中若需要辨識，
還是可以寫出來

For … Next敘述

- 錯誤的用法：(一次都不會執行)

```
For I = 10 To 1 Step 1
    Console.WriteLine(I & “ I Love VB”)
Next I
```

- 或

```
For I = 1 To 10 Step -1
    Console.WriteLine(I & “ I Love VB”)
Next I
```

- 原因：增/減量無法讓初始值趨向終止值

For練習一

- 1. 計算1+2+3...+1000的值
- 2. 計算1+3+5...+99的值
- 3. 印出1~100之間可被7整除的數
- 4. 印出1~1000之間可被7和13整除的數

For練習一 參考寫法

- 1. 計算1+2+3…+1000的值

```
Dim Sum As Integer = 0
For i = 1 To 1000
    Sum = Sum + i
Next
Console.WriteLine(Sum) ’印出結果
Console.Read()  ’暫停螢幕
```

For練習一 參考寫法

- 2. 計算1+3+5…+99的值

```
Dim Sum As Integer = 0
For i = 1 To 99 Step 2
    Sum += i
Next
Console.WriteLine(Sum) ’印出結果
Console.Read()  ’暫停螢幕
```

For練習一 參考寫法

- 印出1~100之間可被7整除的數

```
For i = 1 To 100
    If i Mod 7 = 0 Then
        Console.WriteLine(i)
    End If
Next
Console.Read()
```

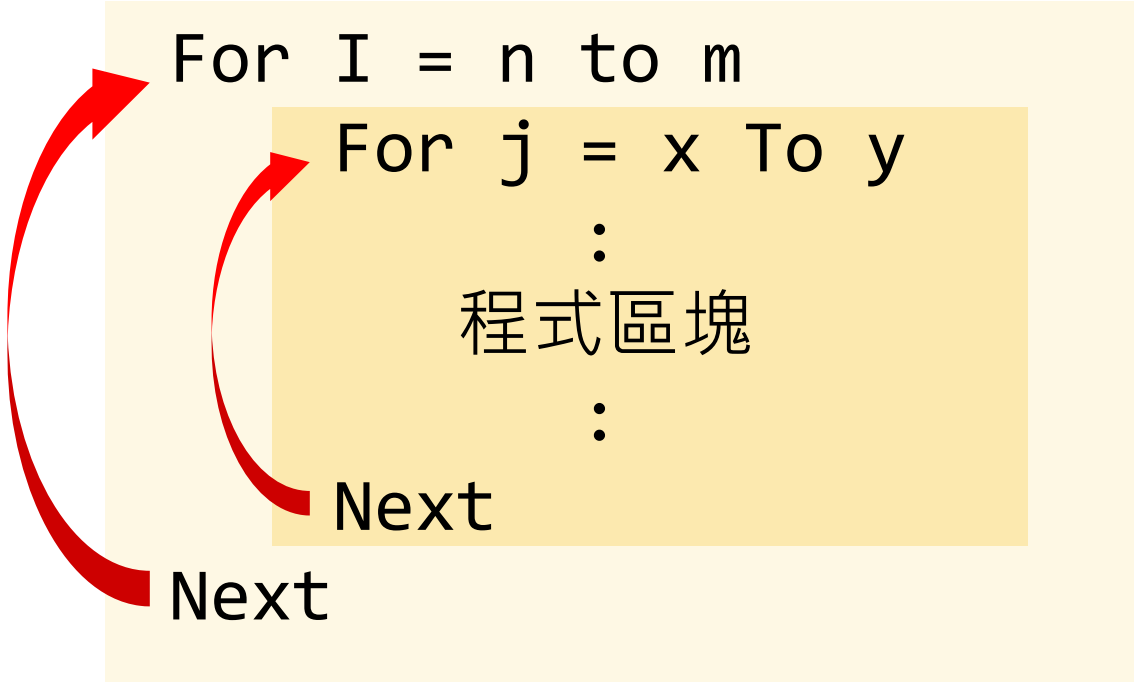
For練習一 參考寫法

- 印出1~1000之間可被7和13整除的數

```
For i = 1 To 100
    If i Mod 7 = 0 And I Mod 13 = 0 Then
        Console.WriteLine(i)
    End If
Next
Console.Read()
```

For巢狀迴圈

- 就是迴圈裏面還有迴圈



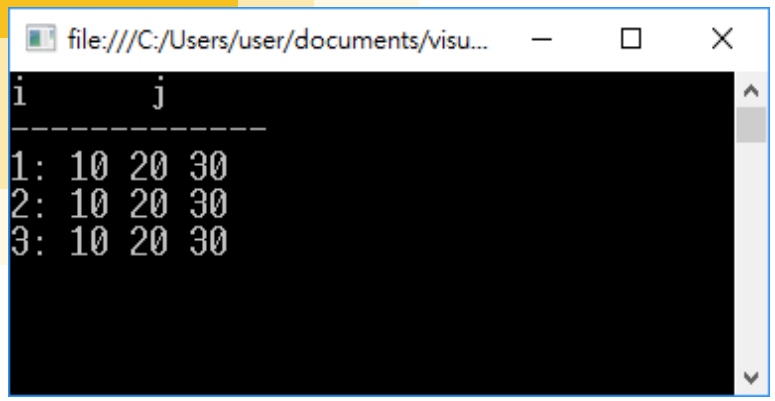
- 注意：套得越多層，效率越差

For巢狀迴圈

- 範例說明：

```
Console.WriteLine("i      j")
Console.WriteLine("-------------")
For i = 1 To 3
    Console.Write(i & ": ")
    For j = 10 To 30 Step 10
        Console.Write(j & " ")
    Next
    Console.WriteLine()
Next
Console.Read()
```

- 執行結果：



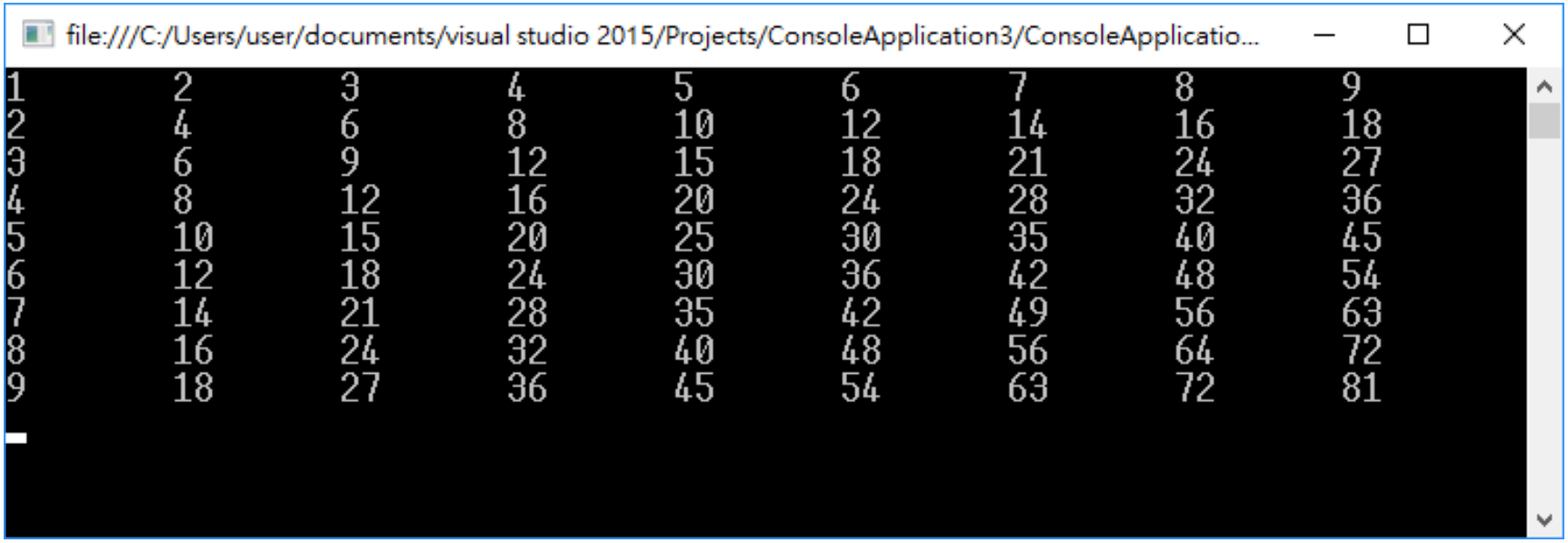
For巢狀迴圈

- 印出一個9*9乘法表，參考寫法：

```
For i = 1 To 9
    For j = 1 To 9
        Console.Write(i * j & vbTab)
    Next
    Console.WriteLine()
Next
Console.Read()
```

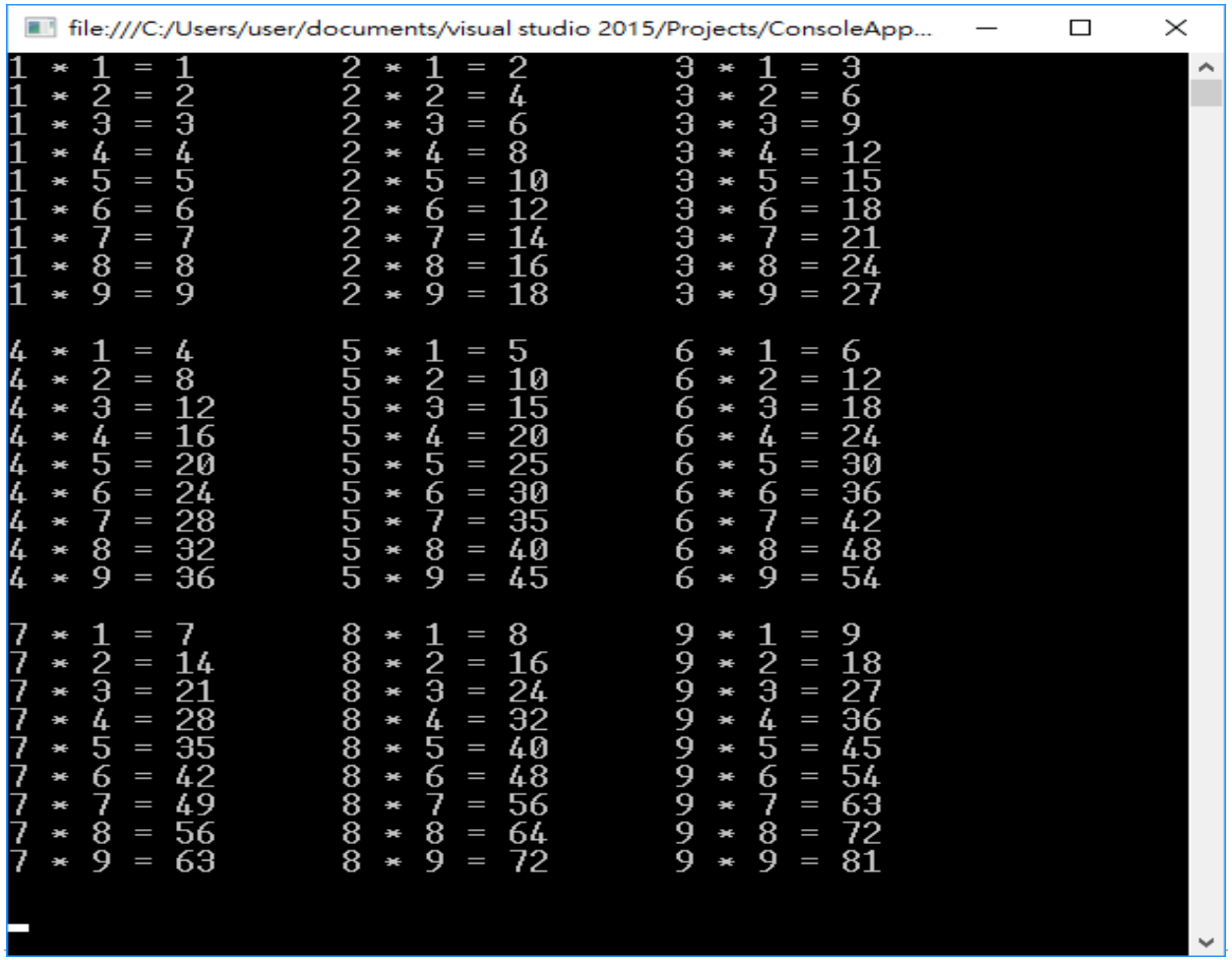
For巢狀迴圈

- 印出一個9*9乘法表執行結果：



For巢狀迴圈

- 練習做出下面的9*9乘法表



For巢狀迴圈

- 9*9乘法表 參考寫法：

```
For i = 1 To 9 Step 3
  For j = 1 To 9
    Console.WriteLine(
           i & " * " & j & " = " & i * j & vbTab &
           i + 1 & " * " & j & " = " & (i + 1) * j & vbTab &
           i + 2 & “ * ” & j & “ = ” & (i + 2) * j & vbTab )
  Next
  Console.WriteLine()
Next
Console.Read()
```

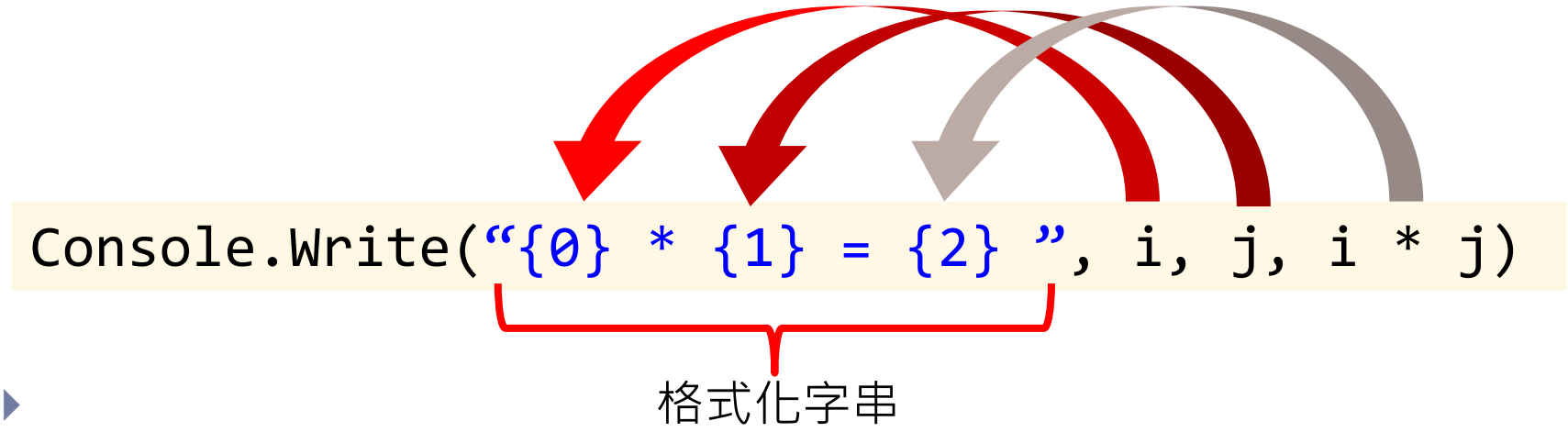
For巢狀迴圈

- 9*9乘法表 另一種寫法：

```
For i = 1 To 9 Step 3
  For j = 1 To 9
    Console.Write(“{0} * {1} = {2} ”, i, j, i * j & vbTab)
    Console.Write("{0} * {1} = {2} ", i + 1, j, (i + 1) * j & vbTab)
    Console.WriteLine("{0} * {1} = {2} ", i + 2, j, (i + 2) * j & vbTab)
  Next
  Console.WriteLine()
Next
Console.Read()
```

補充Consloe.Write( )

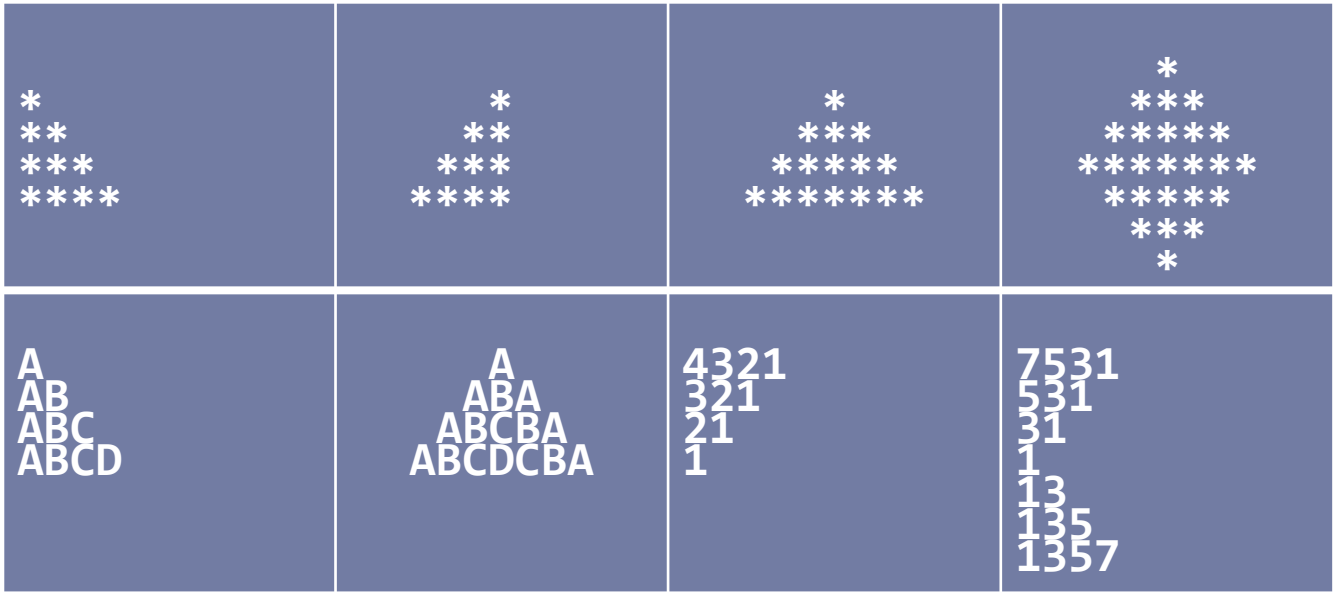
- 可以對輸出做格式化：



- 變數內容依序交給格式化字串裡的 {n} 位置
- 雖然現在都是視窗程式了，但有時還是會用到命令列模式輸出

For練習

- 印出下列圖形：



- 提示：Space(n)函數可以印出n個空白
- 提示：Chr(n)函數可以轉換數值為字元

函數說明

- Space函數可印出指定數目的空白：

```
Console.Write(Space(10) & “*”)
```

會印出10個空白和一個*

- Chr( )與Asc( )函數：

```
Console.Write(Chr(65))
```

- 印出” A” (會印出數值65相對的ASCII字碼(65=A))

```
Console.Write(Asc(“A”))
```

- 印出 65 (會印出字母A相對的ASCII值(A=65))

For練習參考寫法

- 1.

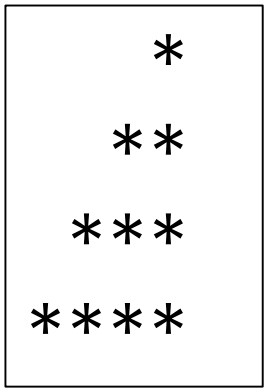
```
For i = 1 To 4
    For j = 1 To i
        Console.Write("*")
    Next
    Console.WriteLine()
Next
Console.Read()
```

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

For練習參考寫法

- 2.

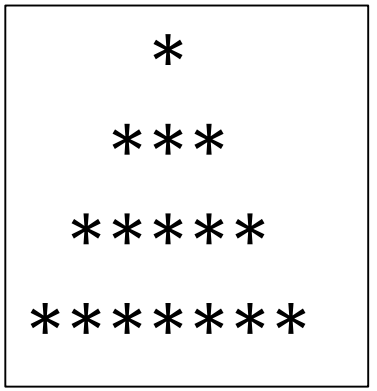
```
For i = 1 To 4
  Console.Write(Space(4 - i))
  For j = 1 To i
    Console.Write("*")
  Next
  Console.WriteLine()
Next
Console.Read()
```



For練習參考寫法

- 3.

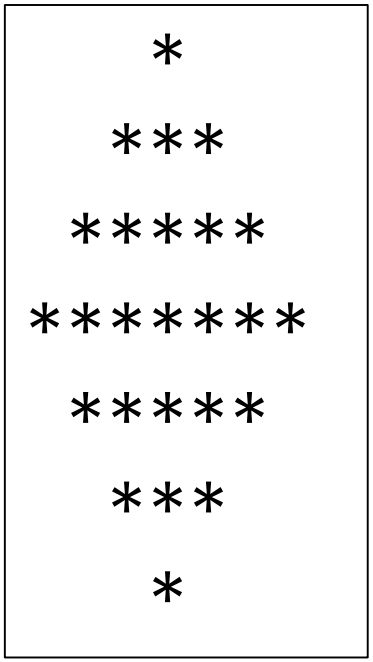
```
Dim X As Integer = 3
For i = 1 To 7 Step 2
  Console.Write(Space(X))
  X = X - 1
  For j = 1 To i
    Console.Write("*")
  Next
  Console.WriteLine()
Next
Console.Read()
```



For練習參考寫法

4.

```
Dim X As Integer = 3
For i = 1 To 7 Step 2 '畫上半部
  Console.Write(Space(X))
  X = X - 1
  For j = 1 To i
    Console.Write("*")
  Next
  Console.WriteLine()
Next
X = 1
For i = 5 To 1 Step -2 '畫下半部
  Console.Write(Space(X))
  X = X + 1
  For j = 1 To i
    Console.Write("*")
  Next
  Console.WriteLine()
Next
Console.Read()
```



For練習參考寫法

- 5. 提示：數值65是A的ASCII值

```
For i = 0 To 3
  For j = 65 To 65 + i
    Console.Write(Chr(j))
  Next
  Console.WriteLine()
Next
Console.Read()
```

```
A
AB
ABC
ABCD
```

For練習參考寫法

- 6.

```
Dim S As Integer = 3
For i = 0 To 3
  Console.Write(Space(S))
  S = S - 1
  For j = 65 To 65 + i
    Console.Write(Chr(j))
  Next
  For j = 64 + i To 65 Step -1
    Console.Write(Chr(j))
  Next
  Console.WriteLine()
Next
Console.Read()
```

```
   A
  ABA
 ABCBA
ABCDCBA
```

For練習參考寫法

- 7.

```
For i = 4 To 1 Step -1
  For j = i To 1 Step -1
    Console.Write(j)
  Next
  Console.WriteLine()
Next
Console.Read()
```

```
4321
321
21
1
```

For練習參考寫法

8.

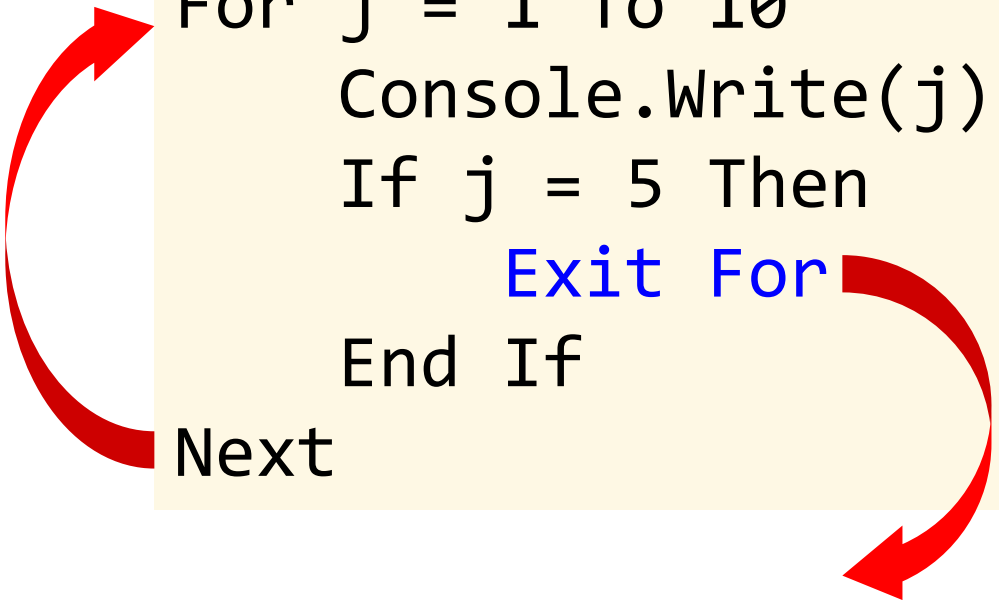
```
'畫上半部
For i = 7 To 1 Step -2
  For j = i To 1 Step -2
    Console.Write(j)
  Next
  Console.WriteLine()
Next
'畫下半部
For i = 3 To 7 Step 2
  For j = 1 To i Step 2
    Console.Write(j)
  Next
  Console.WriteLine()
Next
Console.Read()
```

```
7531
531
31
1
13
135
1357
```

Exit For敘述

- 此敘述可以強迫跳出所在的迴圈，不執行剩下的敘述。
- 範例：

```
For j = 1 To 10
    Console.Write(j)
    If j = 5 Then
        Exit For
    End If
Next
```



跳出迴圈，執行迴圈外的下一道敘述

Exit For敘述

- Exit For只能跳出所在的迴圈，不會一次全跳出來。

```
For i = 1 To 10
    For j = 1 To 10
        Console.Write(j)
        If j = 5 Then
            Exit For
        End If
    Next
    Console.WriteLine()
Next
```

重複結構

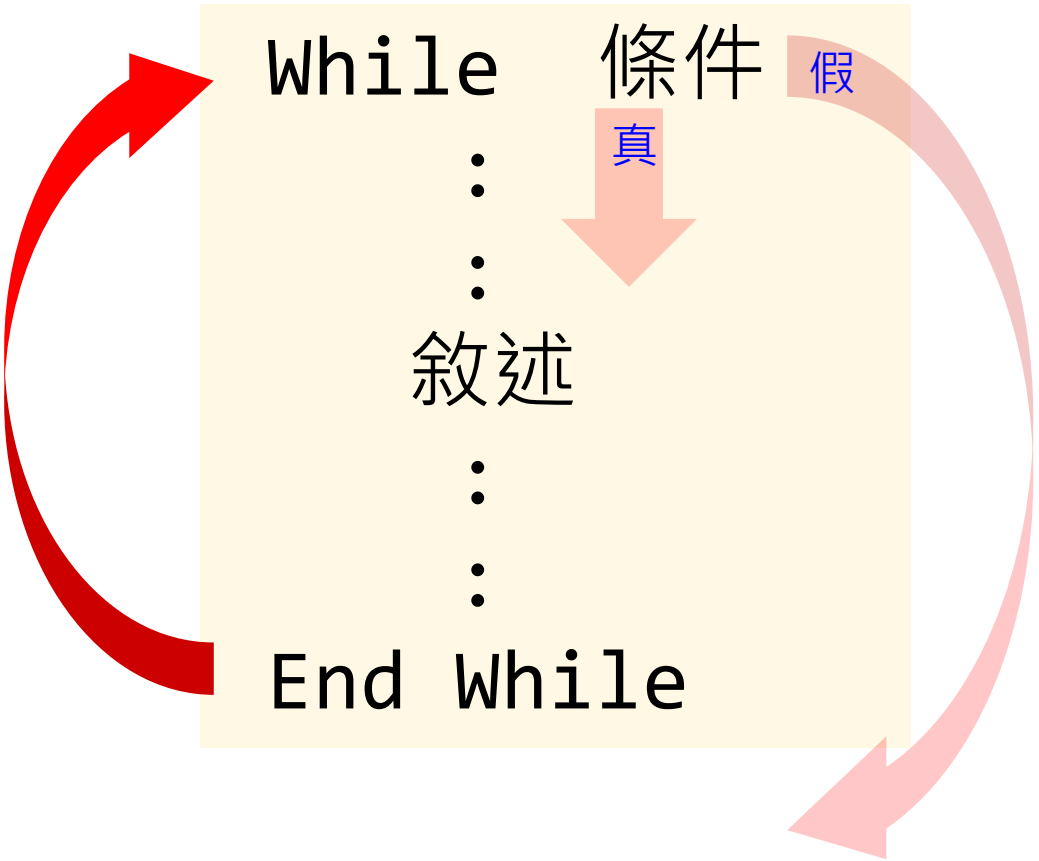
While

While … End While敘述

- 使用For迴圈時是依照起始值及終止值計算需重複多少次，那如果不確定要執行幾次呢? 例如賽車遊戲，如果在時間內繞完一圈，就再玩一次，否則就結束，所以高手和魯蛇玩的次數一定不同，那迴圈要執行幾次呢? 這是不能寫死的。
- 這種不確定要執行幾次的情況就用：

While ... End While敘述

- 當條件為真時執行迴圈，為假時結束迴圈



While ... End While敘述

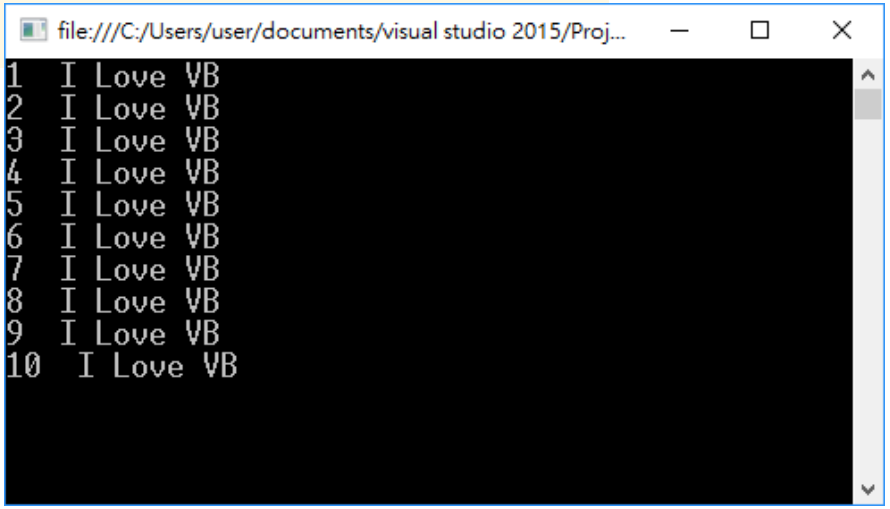
- 範例練習，印出10次”I Love VB” ：

```
Dim i As Integer
i = 1
While i <= 10
    Console.WriteLine(i & " I Love VB")
    i += 1
End While
Console.Read()
```

每次執行條件測試，看是否為真

增/減量

- 執行結果：



While ... End While敘述

- 無盡迴圈，因無意或故意，讓迴圈永遠不會結束
- 例：

```
While 1=1
    :
    :
End While
```

故意的，因為測試條件永遠是**真**

```
While 1
    :
    :
End While
```

```
i = 1
While i <= 10
   Console.WriteLine(i)
      ?
End While
```

無意的，因為忘了改變i值，使測試條件永遠是**真**

While ... End While敘述

- While當無盡迴圈用是常見的
- 不是的話記得要有敘述改變測試條件

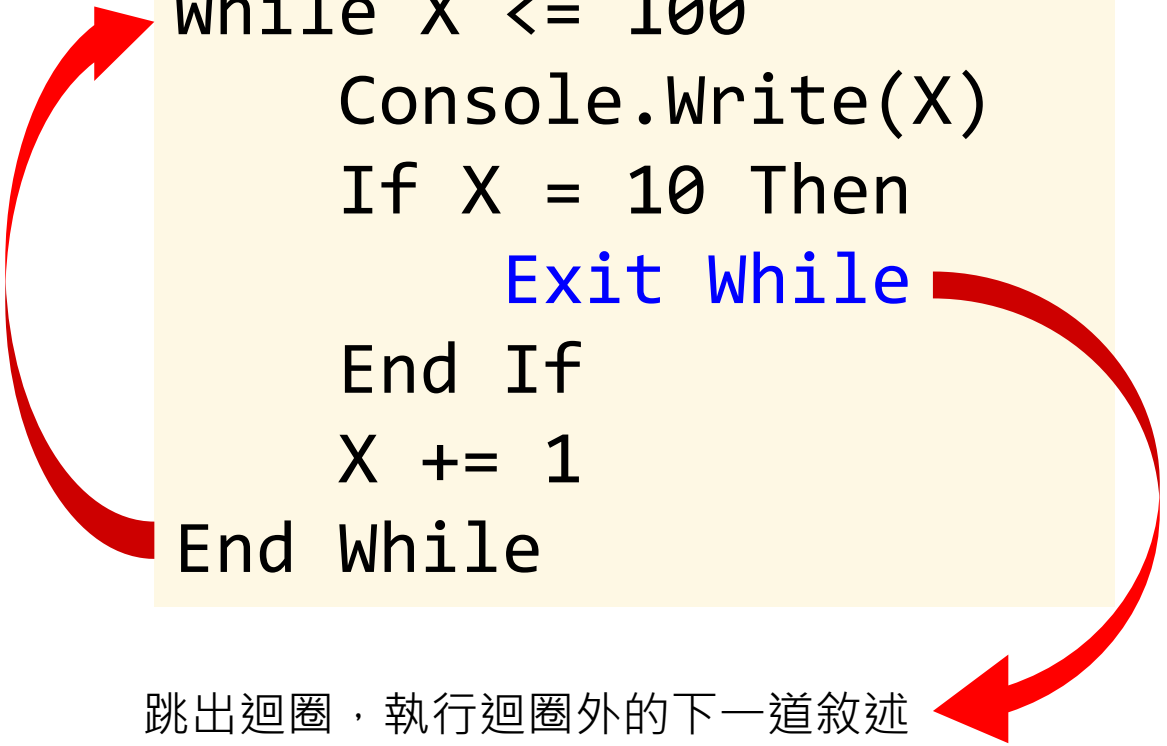
```
While 測試條件
      :
    敘述
      :
End While
```

控制迴圈要不要繼續的關鍵

Exit While敘述

- 此敘述可以強迫跳出所在的迴圈，不執行剩下的敘述，功能與Exit For相同。
- 範例：

```
While X <= 100
    Console.Write(X)
    If X = 10 Then
        Exit While
    End If
    X += 1
End While
```



While練習

- 1. 由鍵盤輸入10個數值並將其加總
- 2. 由鍵盤輸入n個數值，直到輸入999為止(999不算)，將輸入之數值加總印出
- 3. 承上題，增加印出最大值及最小值

While練習 參考寫法

- 1.由鍵盤輸入10個數值並將其加總

```
Dim i, x, sum As Integer
i = 0 : sum = 0
While i < 10
    x = Console.ReadLine()
    sum += x
    i += 1
End While
Console.WriteLine("Sum= " & sum)
Console.Read()
```

- 說明：冒號 : 是將多行敘述寫在同一行時當分隔符號

While練習 參考寫法

- 2. 由鍵盤輸入n個數值，直到輸入999為止(999不算)，將輸入之數值加總印出

```
Dim x, sum As Integer
x = 0 : sum = 0
While x <> 999
    sum += x
    x = Console.ReadLine()
End While
Console.WriteLine("Sum= " & sum)
Console.Read()
```

While練習 參考寫法

- 3. 承上題，增加印出最大值及最小值

```
Dim x, sum, max, min As Integer
x = 0 : sum = 0
max = Int32.MinValue : min = Int32.MaxValue
While x <> 999
    sum += x
    x = Console.ReadLine()
    If x <> 999 Then
        If x > max Then max = x
        If x < min Then min = x
    End If
End While
Console.WriteLine(“Max= ” & max & “, Min= " & min)
Console.WriteLine("Sum= " & sum)
Console.Read()
```

說明：
Int32.MinValue是取得整數可表示的最小值
Int32.MaxnValue是取得整數可表示的最大值

While練習 參考寫法

- 1. 利用While寫一程式，讀入10個數字，找出最大值。

- 2. 利用While寫一程式，讀入10個數字，找出大於13的數字，再求這些數字的和。

Do Loop

Do ... Loop敘述

- 區分為前測式與後測式迴圈：

前測式

先判斷條件再執行迴圈

```
Do While 條件
    :
    :
Loop
```

```
Do Until 條件
    :
    :
Loop
```

後測式

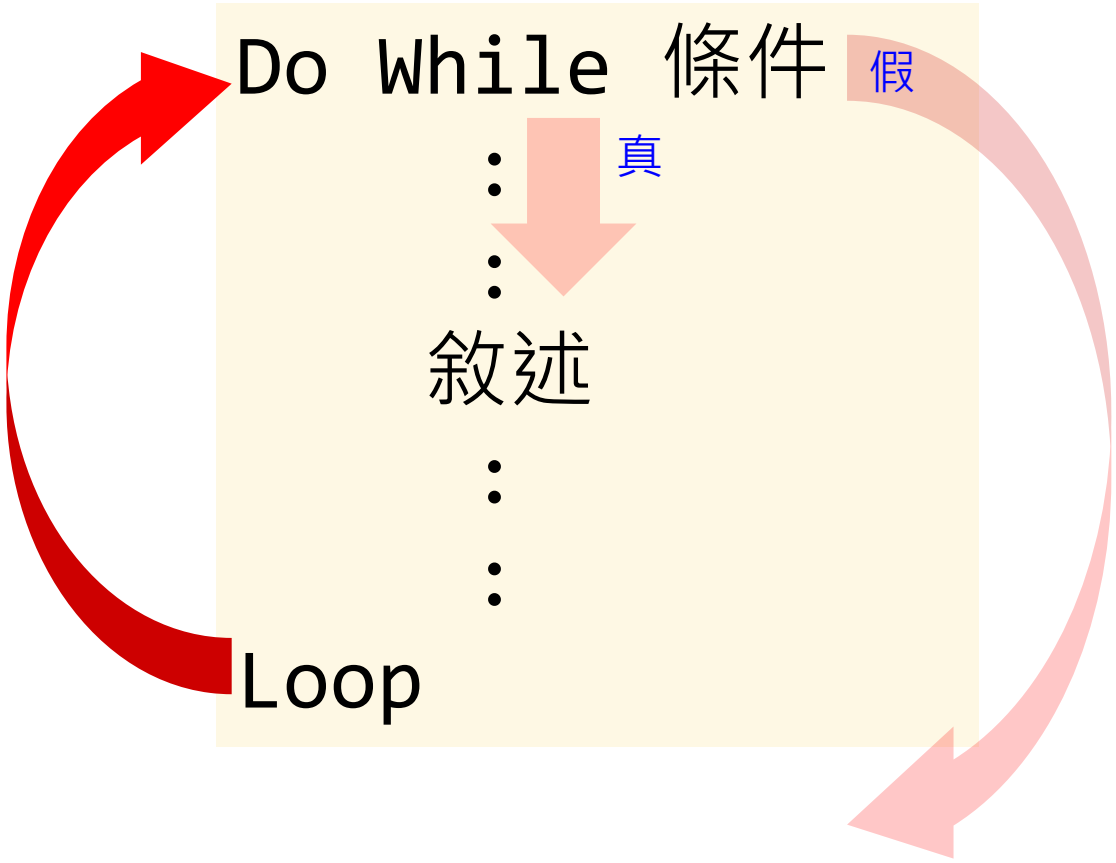
先執行迴圈再判斷條件

```
Do
    :
    :
Loop While 條件
```

```
Do
    :
    :
Loop Until 條件
```

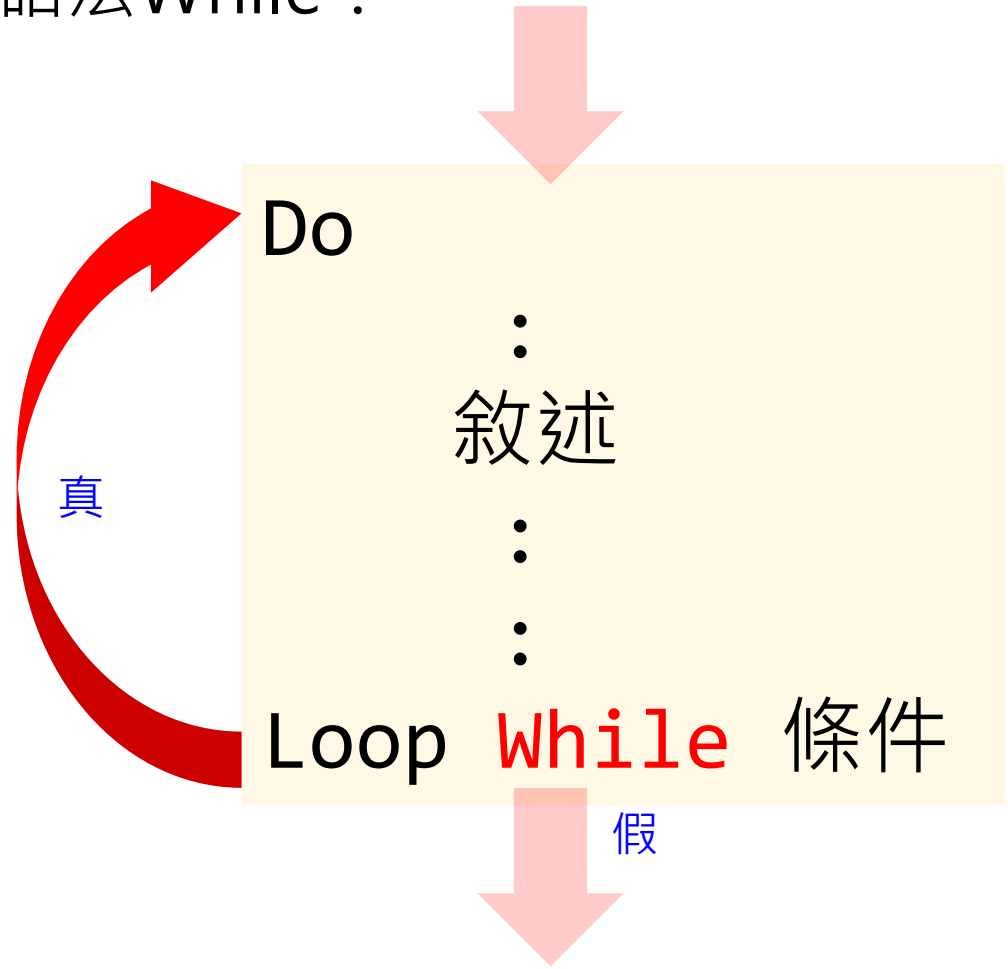
Do ... Loop敘述

- 前測式效果與While敘述相同，端看各位喜歡用哪種敘述來寫



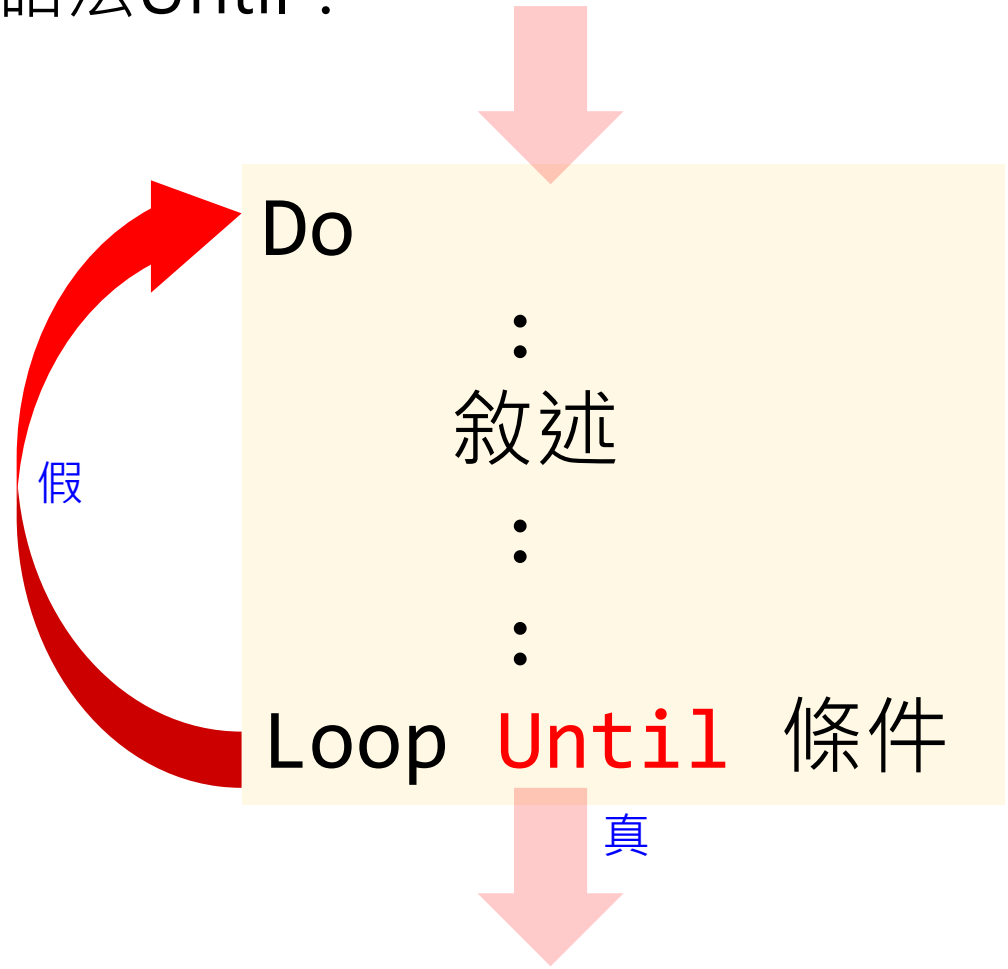
Do ... Loop敘述

- 後測式語法While：



Do ... Loop敘述

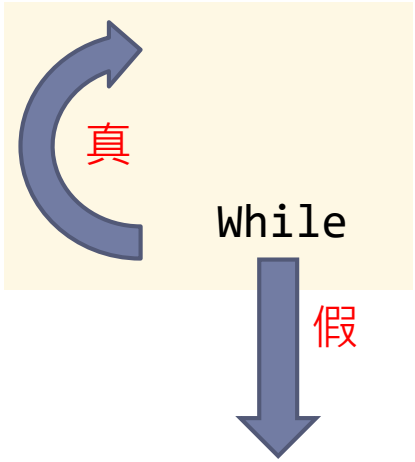
- 後測式語法Until：



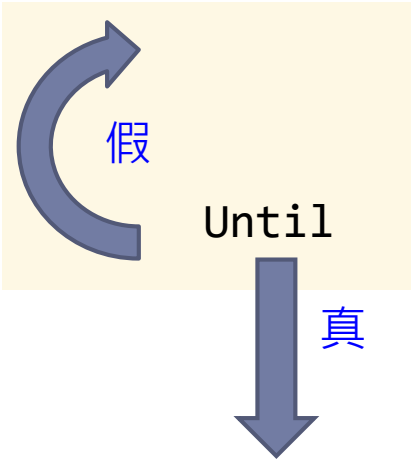
Do ... Loop敘述

- 記住：

While是條件
為真時要重複
為假時要離開

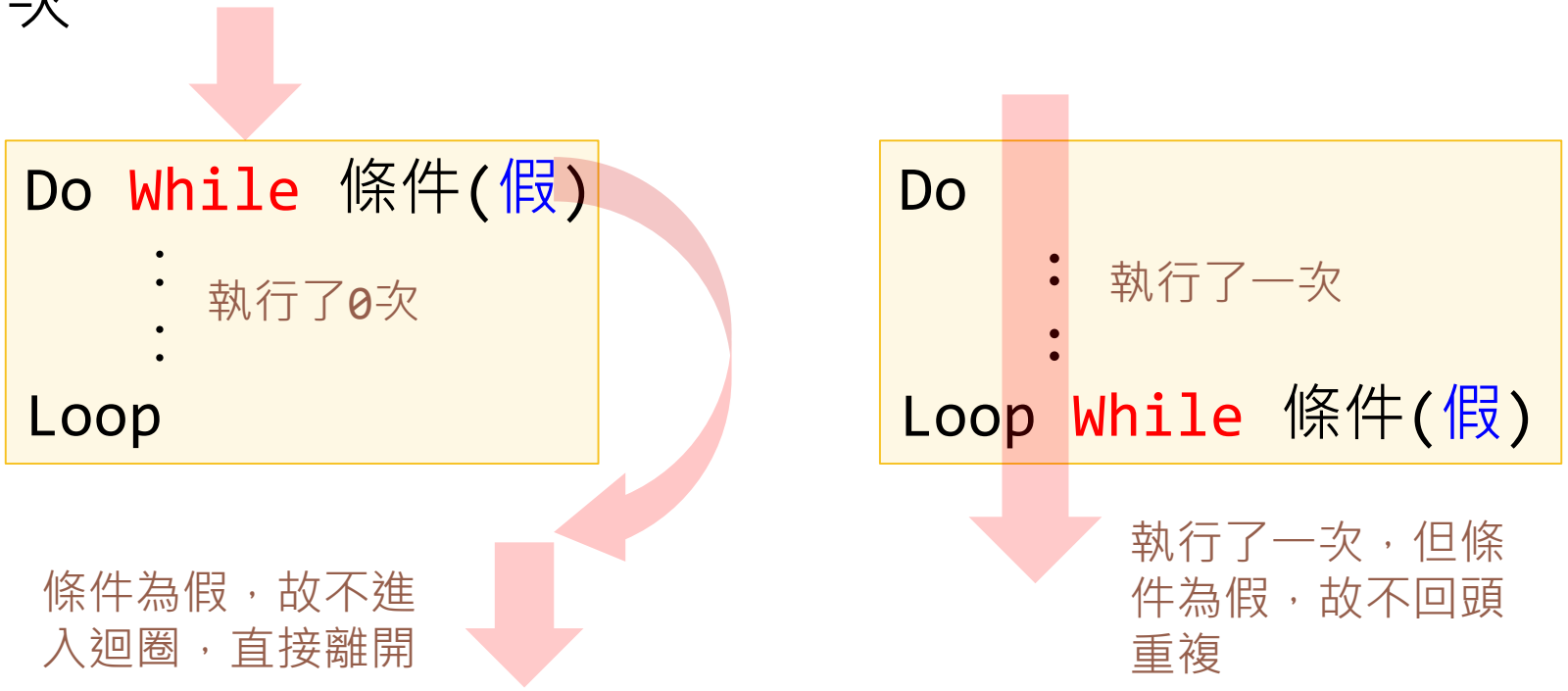


Until是條件
為假時要重複
為真時要離開



Do ... Loop敘述

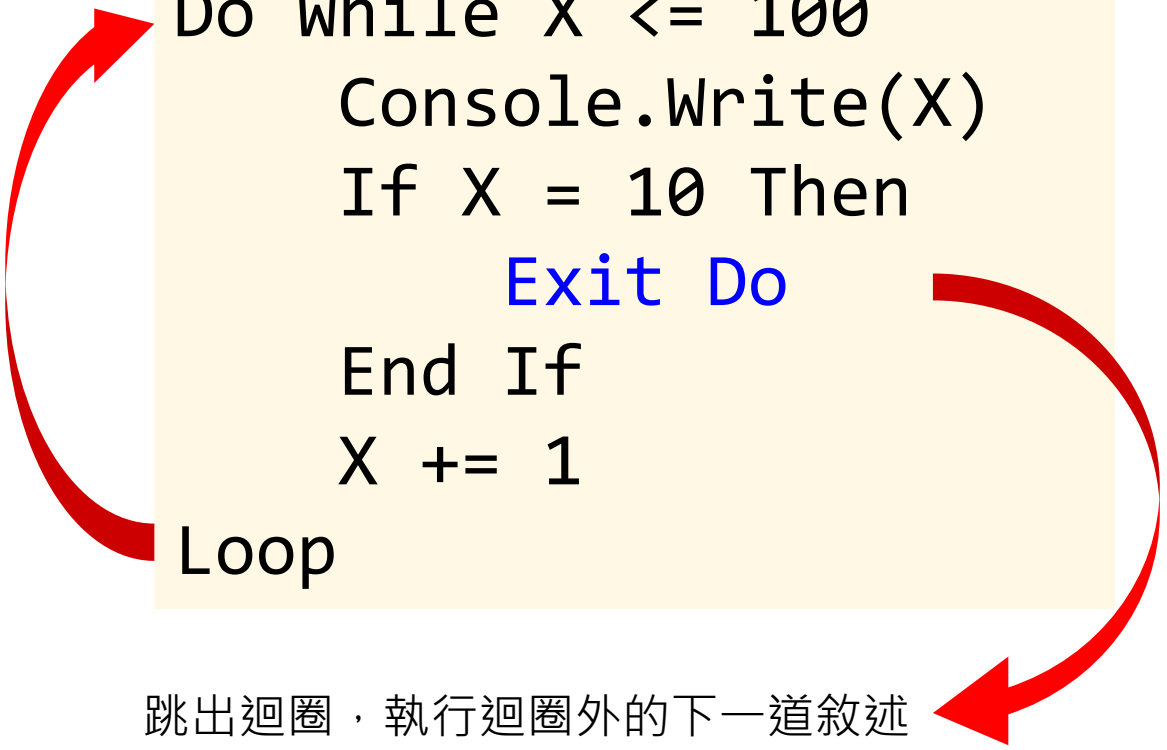
- 後測式是先執行迴圈內的敘述，再判斷要不要繼續重複執行
- 後測式最大的不同是迴圈內的敘述一定會執行至少一次



Exit Do敘述

- 此敘述可以強迫跳出所在的迴圈，不執行剩下的敘述，功能與Exit While相同。
- 範例：

```
Do While X <= 100
    Console.Write(X)
    If X = 10 Then
        Exit Do
    End If
    X += 1
Loop
```



Continue 指令

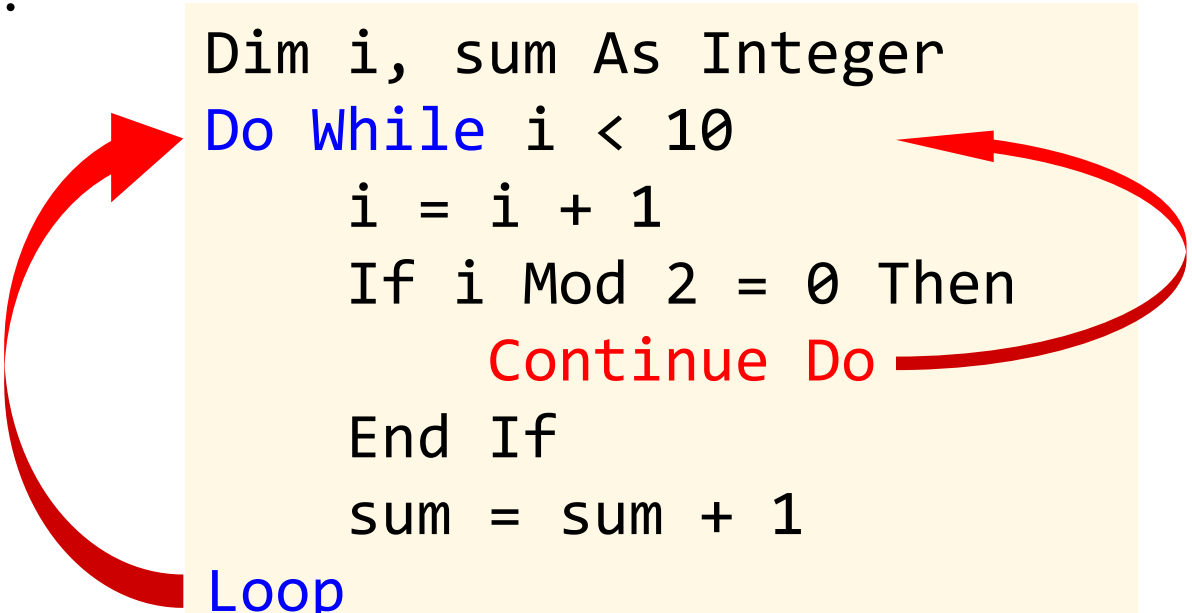
- Continue For：
  跳過For迴圈剩下的指令，回到For的開頭，執行下一個For/Next迴圈。

- Continue Do：
  忽略之後的指令，跳回Do去執行。

- 跟Exit指令相反

Continue 指令

- 範例：

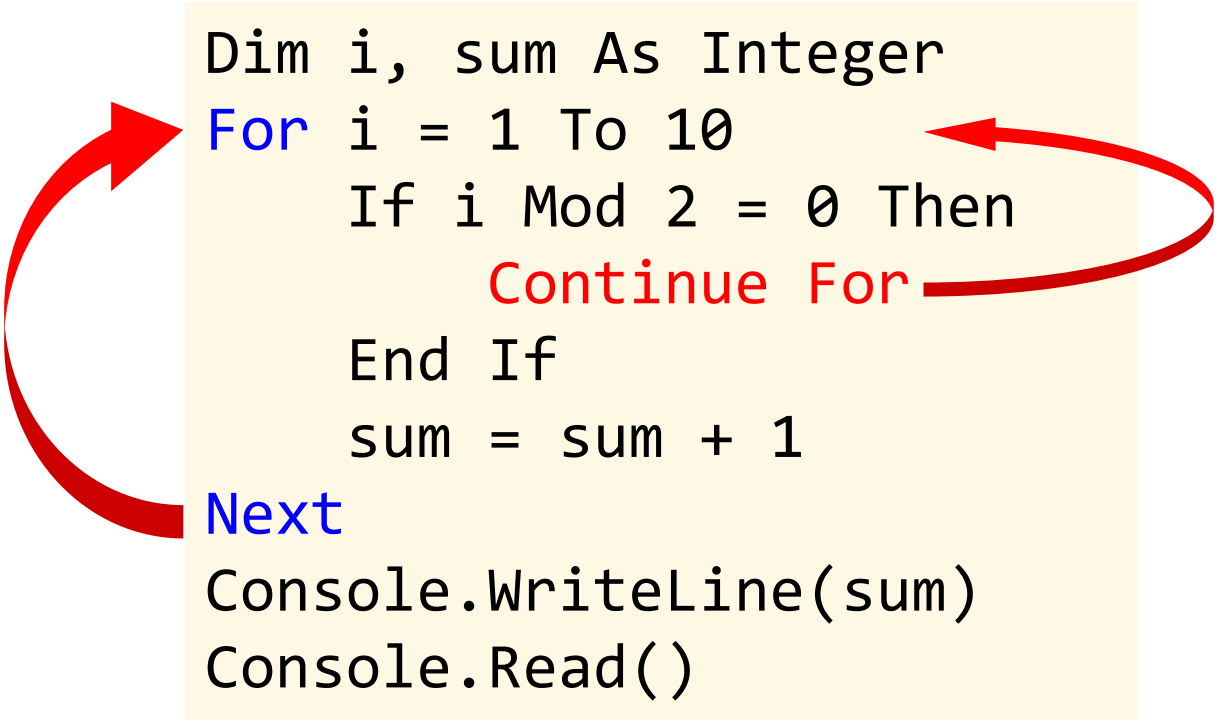


```
Dim i, sum As Integer
Do While i < 10
    i = i + 1
    If i Mod 2 = 0 Then
        Continue Do
    End If
    sum = sum + 1
Loop
Console.WriteLine(sum)
Console.Read()
```

- 執行結果： 5

Continue 指令

- 範例：



```
Dim i, sum As Integer
For i = 1 To 10
    If i Mod 2 = 0 Then
        Continue For
    End If
    sum = sum + 1
Next
Console.WriteLine(sum)
Console.Read()
```

- 執行結果： 5

Do ... Loop練習

- 將前面While的程式改寫為Do Loop型式

- 1. 由鍵盤輸入10個數值並將其加總

- 2. 由鍵盤輸入n個數值，直到輸入999為止(999不算)，將輸入之數值加總印出

Do Loop練習 參考寫法

- 1.由鍵盤輸入10個數值並將其加總(前置)

```
Dim i, x, sum As Integer
i = 0 : sum = 0
Do While i < 10
    x = Console.ReadLine()
    sum += x
    i += 1
Loop
Console.WriteLine("Sum= " & sum)
Console.Read()
```

- 說明：冒號 : 是將多行敘述寫在同一行時當分隔符號

Do Loop練習 參考寫法

- 1.由鍵盤輸入10個數值並將其加總(後置)

```
Dim i, x, sum As Integer
i = 0 : sum = 0
Do
    x = Console.ReadLine()
    sum += x
    i += 1
Loop While i < 10
Console.WriteLine("Sum= " & sum)
Console.Read()
```

- 說明：冒號 : 是將多行敘述寫在同一行時當分隔符號

Do Loop練習 參考寫法

- 1.由鍵盤輸入10個數值並將其加總(前置)

```
Dim i, x, sum As Integer
i = 0 : sum = 0
Do Until i >= 10
    x = Console.ReadLine()
    sum += x
    i += 1
Loop
Console.WriteLine("Sum= " & sum)
Console.Read()
```

- 說明：冒號 : 是將多行敘述寫在同一行時當分隔符號

Do Loop練習 參考寫法

- 1.由鍵盤輸入10個數值並將其加總(後置)

```
Dim i, x, sum As Integer
i = 0 : sum = 0
Do
    x = Console.ReadLine()
    sum += x
    i += 1
Loop Until i >= 10
Console.WriteLine("Sum= " & sum)
Console.Read()
```

- 說明：冒號 : 是將多行敘述寫在同一行時當分隔符號

Do Loop練習 參考寫法

- 2. 由鍵盤輸入n個數值，直到輸入999為止(999不算)，將輸入之數值加總印出(前置)

```
Dim x, sum As Integer
x = 0 : sum = 0
Do While x <> 999
    sum += x
    x = Console.ReadLine()
Loop
Console.WriteLine("Sum= " & sum)
Console.Read()
```

Do Loop練習 參考寫法

- 2. 由鍵盤輸入n個數值，直到輸入999為止(999不算)，將輸入之數值加總印出(後置)

```
Dim x, sum As Integer
x = 0 : sum = 0
Do
    sum += x
    x = Console.ReadLine()
Loop While x <> 999
Console.WriteLine("Sum= " & sum)
Console.Read()
```

Do Loop練習 參考寫法

- 2. 由鍵盤輸入n個數值，直到輸入999為止(999不算)，將輸入之數值加總印出(前置)

```
Dim x, sum As Integer
x = 0 : sum = 0
Do Until x = 999
    sum += x
    x = Console.ReadLine()
Loop
Console.WriteLine("Sum= " & sum)
Console.Read()
```

Do Loop練習 參考寫法

- 2. 由鍵盤輸入n個數值，直到輸入999為止(999不算)，將輸入之數值加總印出(後置)

```
Dim x, sum As Integer
x = 0 : sum = 0
Do
    sum += x
    x = Console.ReadLine()
Loop Until x = 999
Console.WriteLine("Sum= " & sum)
Console.Read()
```

休息一下~