



# 第一章

# 程式邏輯訓練導論

# 本章學習目標



1. 讓讀者瞭解「程式邏輯及演算法」定義及設計原則。
2. 讓讀者瞭解「演算法與程式的差異」。

# 本章內容



**1-1 何謂程式邏輯？**

**1-2 撰寫演算法的原則**

**1-3 演算法的種類**

**1-4 程式設計概念**

**1-5 演算法與程式的差異？**

**1-6 為什麼要撰寫程式？**

**1-7 一個好程式需要滿足條件**



# 1-1 何謂程式邏輯？

## 【引言】

我們時常聽到有人說：「我數學不好」，所以，我就不會寫程式。其實答案是「不一定」的。因為數學必須要同時兼具「邏輯思考」及「運算」。但是，寫程式著重在「邏輯思考」，而「運算」部分就交給電腦的CPU來處理了，其中「邏輯思考」我們又可稱它為「程式邏輯」，而在「程式設計」課程中，它就是一種「演算法」。

## 【演算法的定義】

在韋氏辭典中定義為：「在有限步驟內解決數學問題的程序」。我們可以把演算法(Algorithm)定義成：「解決問題的方法」。



## 【特性】

1. 利用「邏輯方式」來描述「解決問題」的步驟。
2. 利用「文字、流程圖或虛擬碼」方式來撰寫流程。
3. 撰寫方式「與程式語言」的選擇「無關」。

【實例】請寫出「製作一個蛋糕」的演算法。

步驟一：準備三顆雞蛋、一包麵粉及一瓶鮮奶。

步驟二：將三顆雞蛋、一包麵粉及一瓶鮮奶等三項食材，先倒入鍋子中。  
然後，再進行攪拌。

步驟三：再將攪拌後的食材，從鍋子倒到模型器中。

步驟四：最後，放入烤箱，並且設定10分鐘的加熱時間，即可完成一個  
蛋糕。完成之後，如下圖所示。

# 【動畫圖解】

步驟一：準備雞蛋、麵粉及鮮奶	步驟二：攪拌
	
步驟三：倒入模型器中	步驟四：設定烤箱加熱時間，即可完成一個蛋糕。
	

# 1-2 撰寫演算法的原則

## 【引言】

演算法是由有限的步驟組成，因此，如果依照這些步驟執行，一定可以解決某一特定的問題。所以，撰寫演算法必須遵守五點原則。

## 【演算法五點原則】

1. 輸入(Input)：不一定要有輸入。可能沒有，也可能是多個資料輸入。

例如(1)：取得系統目前的時間，不須要輸入，只要寫一行now()函數，就可以輸出系統時間。

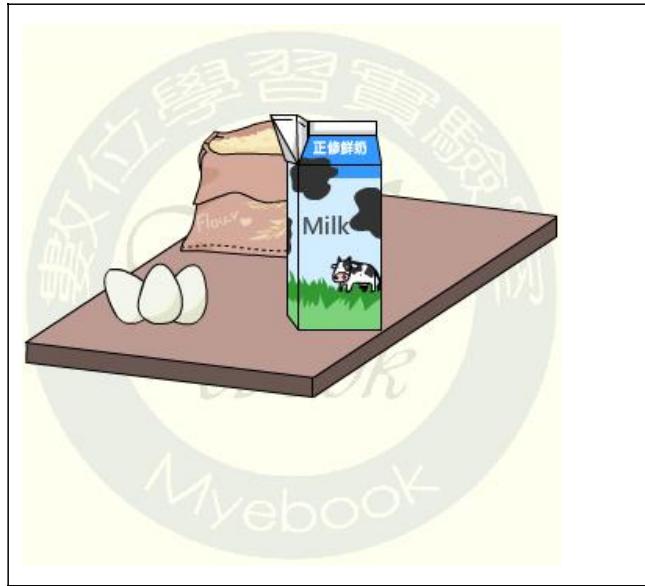
例如(2)：求某數為奇偶數時，則必須先要有一個整數輸入，才能進行判斷。

## 【實例】製作蛋糕的方法

在我們前面所介紹製作蛋糕的例子中，必須要輸入多項食材，例如，要放入「雞蛋、麵粉及鮮奶」等食材。如下圖所示。



## 【圖解說明】



2. 輸出(Output)：至少一個輸出。

例如：在電腦中，處理資料的基本過程有三個步驟：

輸入 → 處理 → 輸出

(原始資料) (程式) (有用的資訊)

所以，使用電腦來為我們處理資料時，有可能是系統自動接收到一個訊號，來當作輸入資料，但是系統至少會輸出一項讓使用者參考的有用資訊。

## 【實例】製作蛋糕的方法

在輸入多項食材之後，最後一定至少會「輸出」一個蛋糕。

### 【圖解說明】



### 3.明確性(Definiteness)：每一行指令都必須明確，不可模稜兩可。

例如(1)：判斷某一數值是否為偶數。

首先我們試著用下列文字來加以描述：



- ①輸入一個正整數。
- ②做「餘除運算」是否為 0。
- ③為 0 即為偶數。

以上描述看來似乎正確，但是從演算法觀點來看，其中的第2點並不符合「明確性」，因它並未說明「餘除運算」是如何運算，容易造成混淆與不解。我們應該改寫為：

- ①輸入一個正整數 N。
- ②如果 N 除以 2，其餘數為 0。
- ③則其 N 為偶數。

例如(2)：「用功的學生才能領獎學金」就不具有明確性，因為每個人對用功的定義可能不盡相同，而如果改為「成績90以上的學生才能領獎學金」就是具有明確性，因為90分是一個比較客觀的定義。

### 【實例】製作蛋糕的方法

製作蛋糕時，要加入多少的麵粉與雞蛋及要加熱多久，必須明確，不可模稜兩可。

### 【圖解說明】



4.有限性(Finiteness)：演算法不能有無窮迴路，必須能終止執行，亦即必須在有限的步驟內完成。

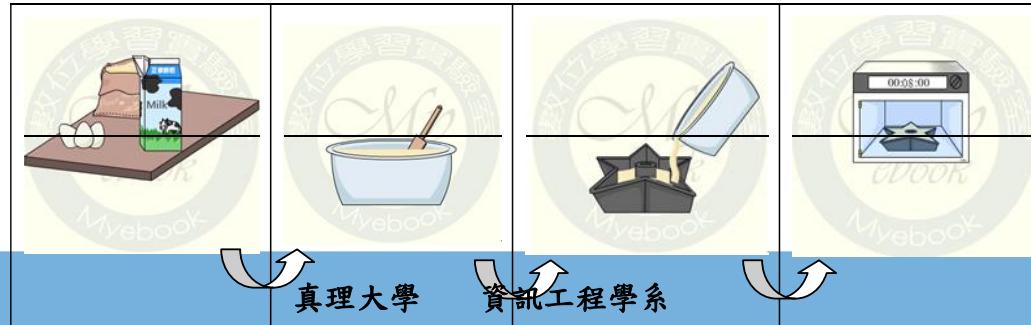
由於演算法並非是真正可以執行的程式，而是設計者所推演出解決問題的步驟，因此，必須在有限的步驟內要完成解決問題的程序，例如上述判斷某數為奇偶數的演算法。但是，真正的程式是可以有無窮迴路的動作。

例如：Windows 作業系統除非系統關機或當機，否則它會永遠執行一個「等待迴圈」，來等待使用者從鍵盤或其他的輸入設備輸入。

### 【實例】製作蛋糕的方法

製作蛋糕時，必須在有限的步驟內完成。

### 【圖解說明】



## 5.正確性(Correctness)：既然演算法是解決問題的方法，因此，正確性是最基本的要求。

例如：以下判斷某數為奇偶數的演算法，雖然符合「明確性」，但是「不正確」，因為N 除以2，其餘數為0，則N應該為「偶數」，而非「奇數」。

- ①輸入一個正整數 N。
- ②如果 N 除以 2，其餘數為 0。
- ③則其 N 為奇數。→應該改為「偶數」

## 【實例】製作蛋糕的方法

例如製作蛋糕時，製作出來的蛋糕，必須要符合使用者的需求。

## 【動畫圖解】





# 1-3 演算法的種類

基本上，我們在撰寫演算法時，有三種方法可以使用。

**第一種：利用文字敘述**

**第二種：利用流程圖(在程式設計課程中，最常被使用)**

**第三種：利用虛擬碼**

**一、文字敘述**

**【定義】**是指利用文字來加以描述解決問題的步驟，但是會比較不精確，因此，一般較不常用。

**【撰寫方式】**採用口語化的文字敘述來加以描述。

**【缺點】**在於冗長且較不精確，在撰寫、閱讀、會意時可能會有誤差。

**【例如】**請利用「文字敘述」來描述，使用者登入帳號與密碼時，系統檢查的過程。



## 【解答】

步驟一：輸入使用者帳號與密碼

步驟二：系統自動檢查是否正確

## (二)流程圖 ( Flowchart )

【定義】利用圖形方式來表達欲解決問題的步驟。

### 【優點】

1. 協助程式設計者設計出周詳的程式。
2. 可增加程式的可讀性。
3. 對於初學者而言可幫助奠定良好的程式設計基礎。

### 【繪圖的思維】

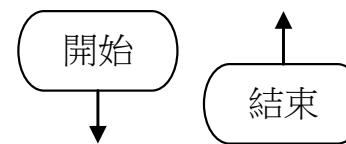
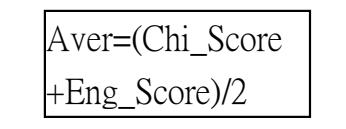
1. 分析要「輸入」那些原始資料。
2. 將輸入的資料加以「處理」。
3. 分析要「輸出」那些資訊報表。

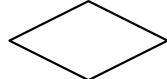
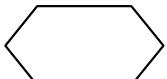
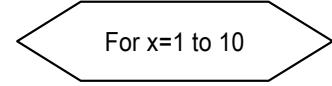
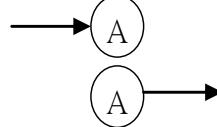
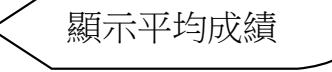
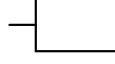
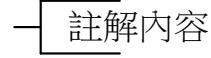


## 【繪圖的原則】

1. 流程圖必須使用標準符號，便於閱讀和分析。如下表【流程圖常用的符號表】所示。
2. 流程圖中的文字力求簡潔、扼要，而且明確可行。
3. 繪製方向應由上而下，由左至右。
4. 流程線條避免太長或交叉，可多用連接符號。

## 【流程圖常用的符號表】

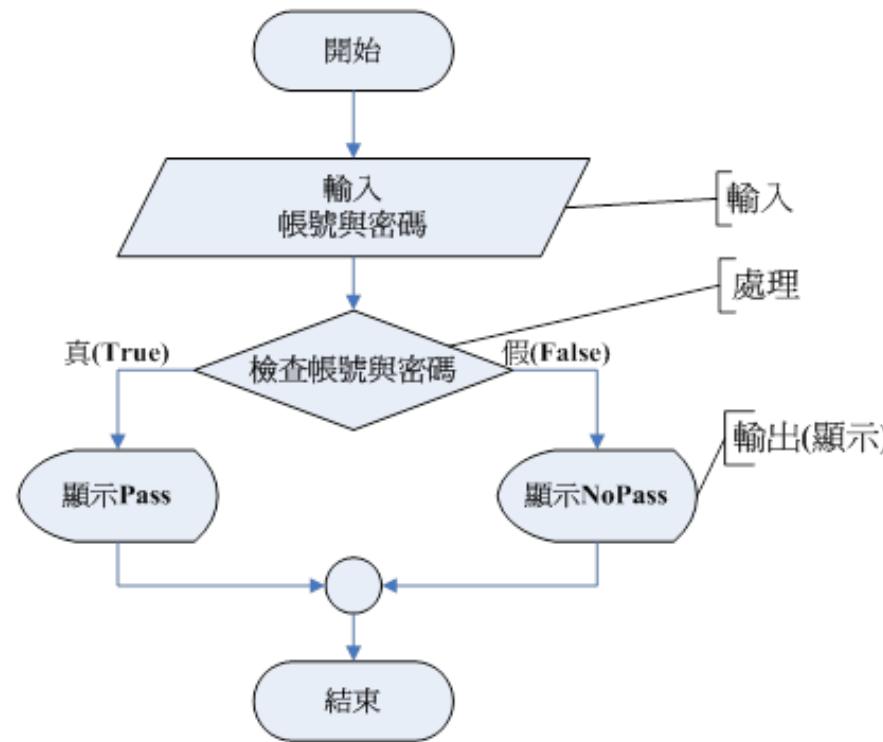
符號	名稱	意義	實例說明
	開始 / 結束符號	表示流程圖的起點或終點，每一個流程圖只有一個起點，但可以有一個以上的終點。	
	輸入 / 輸出符號	表示資料的輸入或輸出。	
	處理符號	表示程式正在執行。	

	決策符號	表示條件式是否成立與否。	
	迴圈符號	固定次數或前測試迴路及後測試迴路。	
	連結符號	表示流程圖的出口或入口的連接點。	
	流向符號	表示程式所執行的方向。	
	顯示符號	表示顯示輸出的結果到螢幕	
	文件符號	表示輸出結果到文件中	
	預定處理符號	表示已定義的副程式	
	註解符號	此符號可對程式加一些說明	

## 【題目】

請利用「流程圖」來描述使用者登入帳號與密碼時，系統檢查的過程。

## 【解答】





### (三)虛擬碼 ( Pseudo Code )

【定義】利用文字中摻雜程式語言，來描述解題步驟與方法。

【優點】兼具「文字描述」及「流程圖」的優點。

【例子1】請利用「虛擬碼 ( Pseudo Code ) 」敘述使用者登入帳號與密碼時，系統檢查的過程。

【解答】

```
(1)Input: UserName, Password  
(2)IF (UserName And Password) ALL True  
    Output: You Can Pass!  
Else  
    Output: You Can not Pass!
```

[註]虛擬碼是無法被執行的指令，它只是用來說明程式處理的流程。

## 【例子2】請撰寫「虛擬碼」來描述 $1+2+3+\dots+10$ 的計算過程。

### 【解答】

- (1) 設 Count=1, Total=0;
- (2) Total=Total+Count;
- (3) Count=Count+1;
- (4) 若 Count  $\leq 10$  則回步驟(2)
- (5) 印出 Total

## 【例子3】請撰寫「虛擬碼」來描述 $10! = 1 \times 2 \times 3 \times \dots \times 10$ 的計算過程。

### 【解答】

- (1) 設 i=1, Result=1;
- (2) Result = Result \* i;
- (3) i=i+1;
- (4) 若 i  $\leq 10$  則回步驟(2)
- (5) 印出 Result

【注意】Result的初值設定為1，否則會產生錯誤的結果。

真理大學 · 資訊工程學系

## 【延伸學習】

基本上，我們在撰寫演算法時，除了上述探討的三種方法之外，我們也可以利用「數學式表示法」。因此，數學式在轉換成程式語言中的運算式時，極為相近。

例如：

### 1. 計算圓面積與周長

數學式	程式語言中的運算式
圓面積= $\pi R^2$	$A=3.14*R^2$
圓周長= $2\pi R$	$A=2*3.14*R$

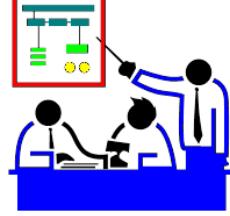
### 2. 轉換攝氏(C)為華氏(F)。

數學式	程式語言中的運算式
$F=9/5C+32$	$F = 9 / 5 * C + 32$

# 1-4 程式設計的流程

基本上，我們在開發一個「資訊系統」時，並非直接撰寫程式，而是必須要先經過一連串的步驟，而「撰寫程式」其實只是其中一個步驟。因此，我們要開始程式設計時，一定要進行下面五個步驟。

## 【圖解】

需求分析	繪製流程圖	撰寫程式	上機測試	撰寫說明書
				

## 【說明】

### 步驟1. 分析所要解決的問題(需求)

- (1)先了解使用者的問題及需求。
- (2)確定要「輸入」那些資料。
- (3)確定要「輸出」那些資訊報表。

## 【動畫圖解】



## 步驟2. 設計解題的步驟(流程圖)

根據使用者的需求，著手撰寫演算法以解決問題，它可以利用文字敘述、流程圖或虛擬碼來表示解決問題的步驟。

### 【動畫圖解】



## 步驟3. 編寫程式 (程式碼)

選擇適當的程式語言，將演算法的步驟寫成一個完整的程式。

【動畫圖解】

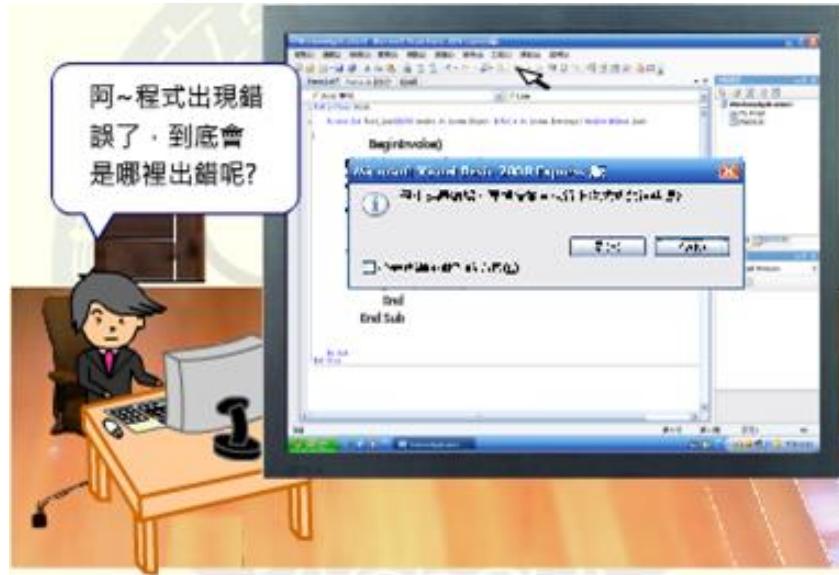


## 步驟4. 上機測試、偵測錯誤 (測試)

一個有用性、易用性的程式，必須要經過多次的測試，若有錯誤，立即更正，直到正確無誤為止。



### 【動畫圖解】



## 步驟5. 編寫程式說明書(可執行)

一個功能強而完整的程式，使用者就會願意使用，因此必須有使用說明書，以便於別人使用或日後的維護；一般而言，在程式書面資料中，一般包括有下列三項：

- (1) 程式的功能、輸入需求及輸出格式。
- (2) 演算法或程式流程圖。
- (3) 測試結果或數據。

### 【動畫圖解】



〔實例〕計算國文與英文的平均成績，並依照平均成績來求顯示「及格」與「不及格」。



### 【解答】

#### 步驟1. 分析所要解決的問題(需求)

依照題意，我們將分成兩種不同的等級

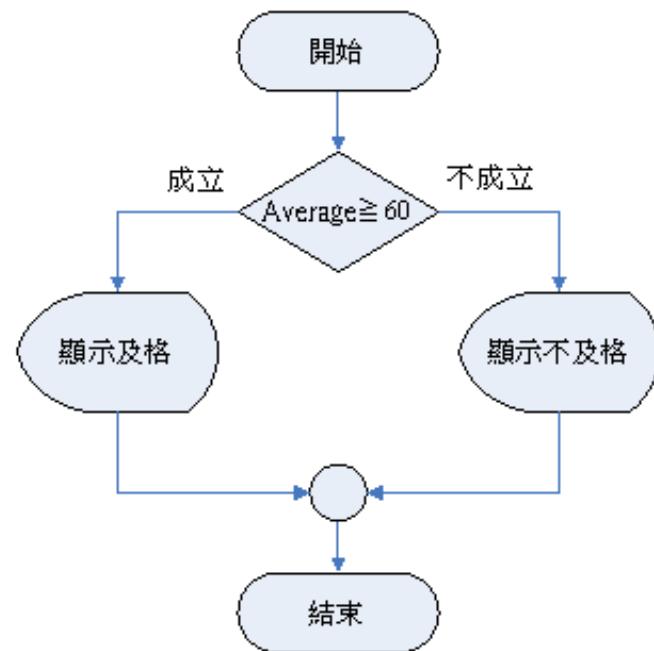
(1)及格：60(含)以上。

(2)不及格：60以下。

#### 步驟2. 設計解題的步驟(演算法)

畫出整合問題的「流程圖」及「虛擬碼」。

# (1) 流程圖



## (2) 虛擬碼



```
01 Procedure Method()
02 Begin
03     int C_Score, E_Score, Average;
04     C_Score=60;
05     E_Score=70;
06     Average = (C_Score + E_Score) / 2;
07     if (Average >= 60)
08         printf("及格");
09     else
10         printf ("不及格");
11 End
12 End Procedure
```

### 3. 編寫程式 (程式碼)

C 語言

```
01 main()
02 {
03     int C_Score, E_Score, Average;
04     C_Score=60;
05     E_Score=70;
06     Average = (C_Score + E_Score) / 2;
07     if (Average >= 60)
08         printf("及格");
09     else
10         printf ("不及格");
11     system("PAUSE");
12     return 0;
13 }
```



## 【程式解析】

- ①行號03~05：宣告3個科目變數，並設定初值
- ②行號06：計算2科的「平均成績」
- ③行號07~10：判斷「平均成績」是否大於60分，如果是的話，則顯示「及格」，否則，顯示「不及格」。

### 步驟4. 上機測試、偵測錯誤 (偵錯)

對每一個程式模組進行測試及除錯，直到沒有錯誤為止。

當使用者輸入國文為60分，英文為61分時，是否可以計算出平均成績為60.5，如果沒有則必須要進行除錯，亦即要將Average的資料型態改為float(浮點數)

## 1-5 演算法與程式的差異？

### 一、演算法

1. 以「人」為主，亦即「任何人都可以閱讀的程式碼」。
2. 強調「可讀性」。

### 二、程式

1. 以「電腦」為主。
2. 強調執行結果的「正確性」、「執行效率」及「可維護性」。
3. 程式「不一定要滿足」演算法中的「有限性」之要求。

例如：電腦主機上的作業系統，除非當機，否則會永遠在等待迴路。

所以「程式」違反了「演算法」應遵守五大原則之「有限性」。

### 【程式的特性】

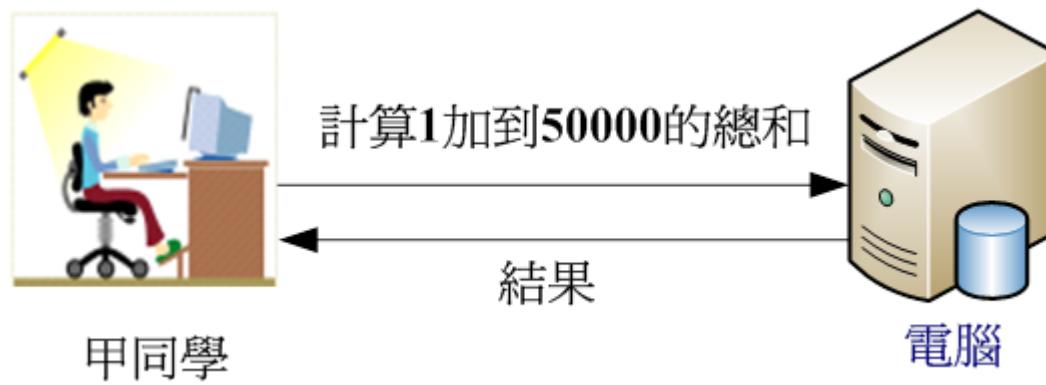
1. 可以實際被執行。
2. 利用程式語言的規範，來真實的解決問題。
3. 可以利用不同程式語言來解決相同的問題。

## 1-6 為什麼要撰寫程式？

您知道，我們為什麼要花那麼多時間來撰寫程式呢？

你、我可能都不是非常的了解。接下來，我們就來說明「為什麼要撰寫程式」，其實在資訊科學的領域中，撰寫程式的主要目的就是快速來幫助人類解決「複雜的問題」。

### 【概念圖】



因此，我們可以從以下兩個例子的說明。

### 【例子1】

小華說：嗨！小明，請你幫我計算1加到10的總和。

小明說：1加到10，太簡單了，大家都會！

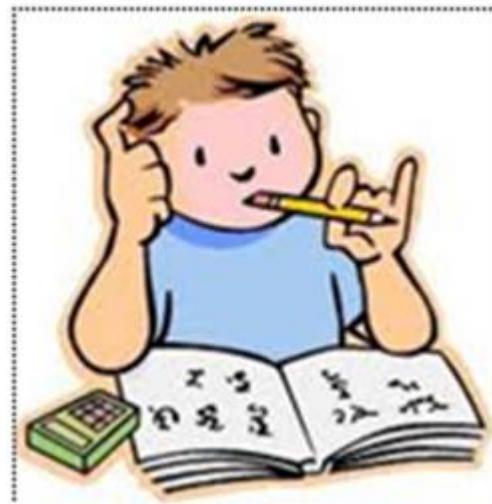
在這個例子中，或許你會認為這簡單的問題，你我都會算，何必寫程式呢？



## 【例子2】

小華說：那小明請你再幫我計算1加到50000。

此時小明說：這太困難了，我無法馬上計算出結果。



但是，我可以「撰寫程式」來計算。

因此，我們可以從以上兩個例子中，清楚得知，「程式語言」是用來幫忙人類「解決複雜的問題」。

基本上，一個好程式需要滿足的條件有以下三點：

**第一點：正確性**

**第二點：效率性**

**第三點：可維護性**

**一、正確性(Correctness)：**

**【定義】正確性是一個好程式最基本的要求。**

**【示意圖】**



# 【例如】設計一個判斷某一數值是否為「偶數」的程式

①輸入：一個正整數 N。

②處理：如果 N 除以 2，其餘數為 0，則 N 就是奇數。→改為「偶數」

③輸出：N 為奇數。→改為「偶數」

說明：上面的程式處理過程中，由於程式不正確，所以產生錯誤的結果。

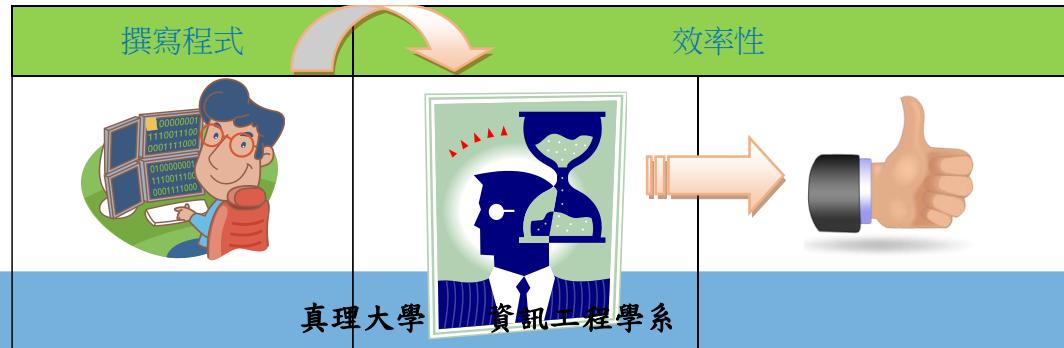
## 二、效率性(Performance)

### 【引言】

當我們撰寫一個可以正確地執行程式之後，接下來就是要再考慮到程式的執行效率，也就是程式真正執行時所必須要花費的時間。

【定義】是指程式真正執行時所必須要花費的時間。

### 【示意圖】



## 【直覺的作法】

程式執行時間 = 「結束時間」減掉「開始時間」

但是，實務上有可能因為程式編譯的過程或是電腦設備的差異使得效率分析會因電腦的軟硬體不同而有不同的結果。

【定義】是指程式真正執行時所必須要花費的時間。

## 【一般評估執行時間的方法】

是依程式碼所被執行的「總次數」來計算。亦即所謂的「頻率次數」，當「頻率次數愈高」時，代表所需的「執行時間愈長」。

【例如】請計算下列程式中變數Count被執行的次數為何？

I . 單一敘述	II . 單層迴圈敘述	III . 雙層迴圈敘述
Count=Count+1;	for (i=1; i <= n; i++) Count=Count+1;	for (i=1; i <= n; i++) for(j=1; j<=n; j++) Count=Count+1;
1 次	n 次	$n^2$ 次

說明：(1)在上圖 I 中， $\text{Count}=\text{Count}+1$ ; 敘述被執行1次。  
(2)在上圖II中， $\text{Count}=\text{Count}+1$ ; 敘述被執行n次。  
(3)在上圖III中， $\text{Count}=\text{Count}+1$ ; 敘述被執行 $n^2$ 次。  
若n=10時，則敘述 $\text{Count}=\text{Count}+1$ ;之執行次數分別是：  
1,10,100之級數增加。

### 三、可維護性(Maintainable)

#### 【引言】

一個好的程式，不只需要有效率地被正確地執行之外，也必須要考慮程式的可讀性、及未來修改和擴充性，這屬於程式設計方法和風格的問題，例如：使用模組化來設計程式和加上完整程式註解的說明。

【定義】是指在撰寫完成一套程式之後，它是可以很容易的讓自己或他人修改。

#### 【示意圖】





## 【三種技巧】

要如何讓程式具有可維護性呢？那就必須在撰寫程式時，使用以下三種技巧：

第一種技巧就是「縮排」。

第二種技巧就是加入「註解」。

第三種技巧就是「變數及函數名稱的命名」要有意義。

### (一)縮排技巧

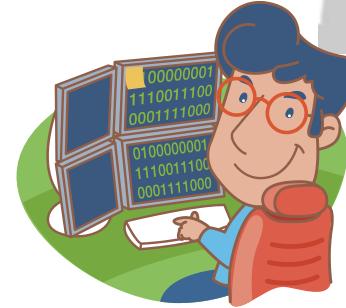
【定義】它是撰寫程式時，最基本撰寫技巧。

【目的】1.了解整個程式碼的邏輯性

2.了解區塊群組的概念。

【優點】易於閱讀及除錯。

## 【示意圖】

適度的縮排	易於閱讀及除錯
	

## 【比較「縮排」與「未縮排」的情況】

①使用「縮排」技巧	②未使用「縮排」技巧
<pre>int i, j; for (i = 1; i &lt;= 9; i++) {     for (j = 1; j &lt;= 9; j++)     {         //程式區塊     } }</pre>	<pre>int i, j; for (i = 1; i &lt;= 9; i++) {     for (j = 1; j &lt;= 9; j++)     {         //程式區塊     } }</pre>

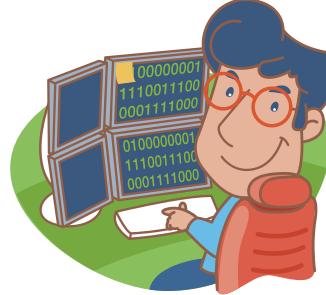
說明：有縮排的程式碼，易於閱讀及除錯。

## (二)註解技巧

**【定義】**它是一種「非執行的敘述」亦即是給人看的，而電腦不會去執行它。

**【功能】**是用來說明某一段程式碼的作用與目的。

**【示意圖】**

適度的註解	助於程式的維護
	



## 【註解的方式】

基本上，註解的方式有二種：

1. 使用雙敘線「//」的使用時機：可以寫在程式碼的後面或單獨一行註解。

```
double R, A, L; //宣告三個變數 R,A,L 為倍精準度
```

2. 使用「/\*.....\*/」的使用時機：註解的內容超過一行時。

```
/*題目：計算圓的面積與周長  
圓面積公式：PI*R2  
圓周長公式：2*PI*R  
*/
```

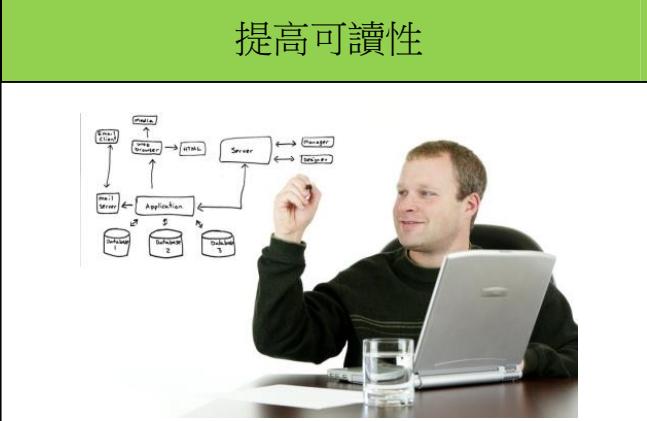
說明：/\*後面的文字內容被視為註解，一直到遇到 \*/為止。

註：在程式中加入「註解」時，註解的文字會自動變成綠色字。真理大學 資訊工程學系 Page : 46

### (三)有意義的變數命名

**【目的】**提高程式的可讀性，以利爾後的除錯。

**【示意圖】**

提高可讀性	利於除錯
	

### 【方法】

- 變數名稱的命名最好具有意義的，並且與該程式有關係的。  
例如：Stu\_Name(代表學生姓名)，Stu\_No(代表學生學號)
- 如果想不出變數名稱的英文字母時，最好在命名時，適時的在變數之後加以註解。

例如：Dim Start\_X AS Integer '宣告X的開始座標