第5章 程式之流程控制(二)──迴圈結構

5-1程式運作模式

- 日常生活中，常常有一段時間會重複做一些固定的事，過了這段時間就換做別的事
  - 每星期六5：00PM時，電視台就會播放卡通節目海賊王，直到電視台與製作片商簽約到期
- 當程式重複執行某些特定的敘述，直到違反條件才停止重複執行，這種架構稱為迴圈結構
  - 當一個問題，涉及重複執行完全相同的敘述或敘述相同但資料不同時，不管是否有的公式，都可利用迴圈結構來處理

C#語言有下列三種運作模式：

- 循序結構：(參考第4 章)
- 選擇結構：(參考第4 章)
- 重複結構：為一種迴圈結構，內部包含一組條件。當程式執行到此結構時，是否重複執行迴圈內部的程式敘述，是由條件的結果來決定
  - 若條件的結果為「true」，則會執行迴圈結構內部的程式敘述；
  - 若條件的結果為「false」，則不會進入迴圈結構內部

5-2 迴圈結構

- 根據條件(這些條件通常是由算術運算式、關係算術運算式及邏輯運算式組合而成)撰寫的位置來區分，迴圈結構分為前測式迴圈及後測式迴圈兩種類型：
  - 前測式迴圈
  - 後測式迴圈

- 前測式迴圈結構：條件寫在迴圈結構開端的迴圈。當執行到迴圈結構開端時，會先檢查條件
  - 若條件的結果為「true」(真)，則會執行迴圈內部的程式敘述，之後會再回到迴圈結構的開端，再檢查條件；
  - 否則不會進入迴圈內部，而是直接跳到迴圈結構外的第一列程式敘述

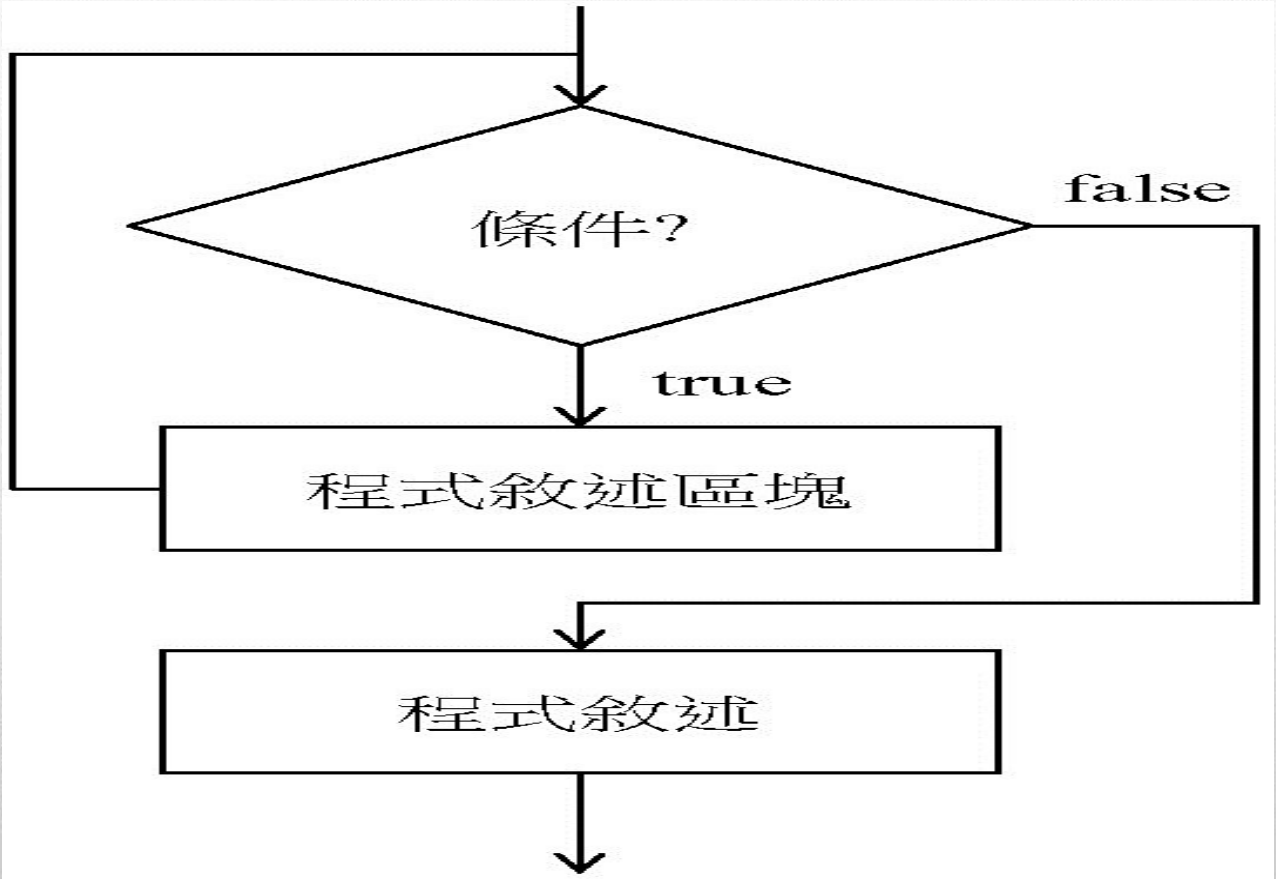


圖5-1　前測式迴圈結構流程圖

- 例：正常的狀況下，在上課時間內學生必須在教室內學習知識，否則可以下課休息的

- 後測式迴圈結構：條件寫在迴圈結構尾端的迴圈。當執行到迴圈結構時，是直接執行迴圈內部的程式敘述，並在迴圈結構尾端檢查條件
  - 若條件的結果為「true」(真)，則會從迴圈結構的開端，再執行一次
  - 否則執行迴圈結構外的第一列程式敘述

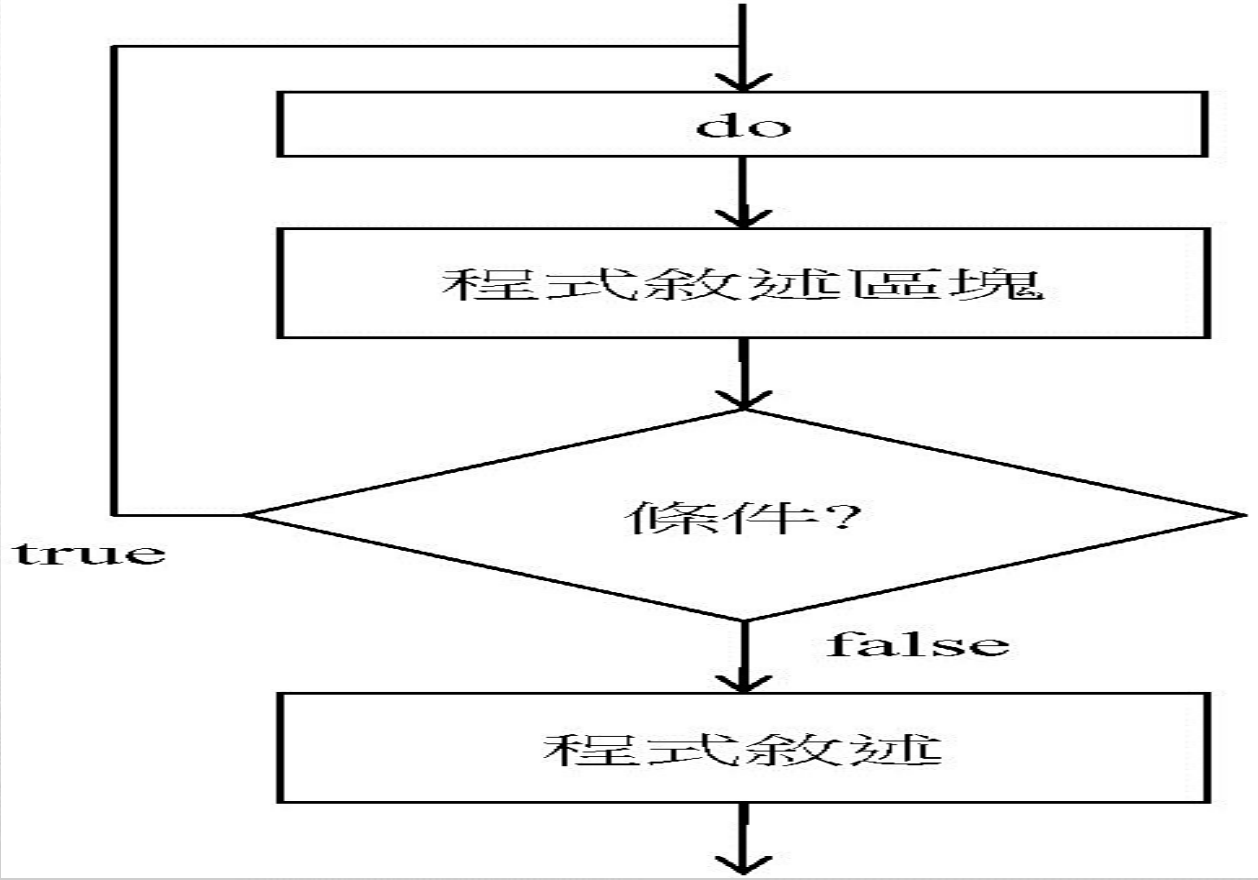


圖5-2　後測式迴圈結構流程圖

- 例：一位大學生是否能畢業，必須視該系之規定。若沒符合該系規定，則必須繼續修課

5-2-1前測式迴圈結構

- 在C#語言提供的前測式迴圈結構，有「for」及「while」兩種迴圈
- 迴圈結構「for」的語法如下：

```
for (迴圈變數初值設定;進入迴圈的條件;迴圈變數增減量)
  {
       程式敘述
       …
  }
```

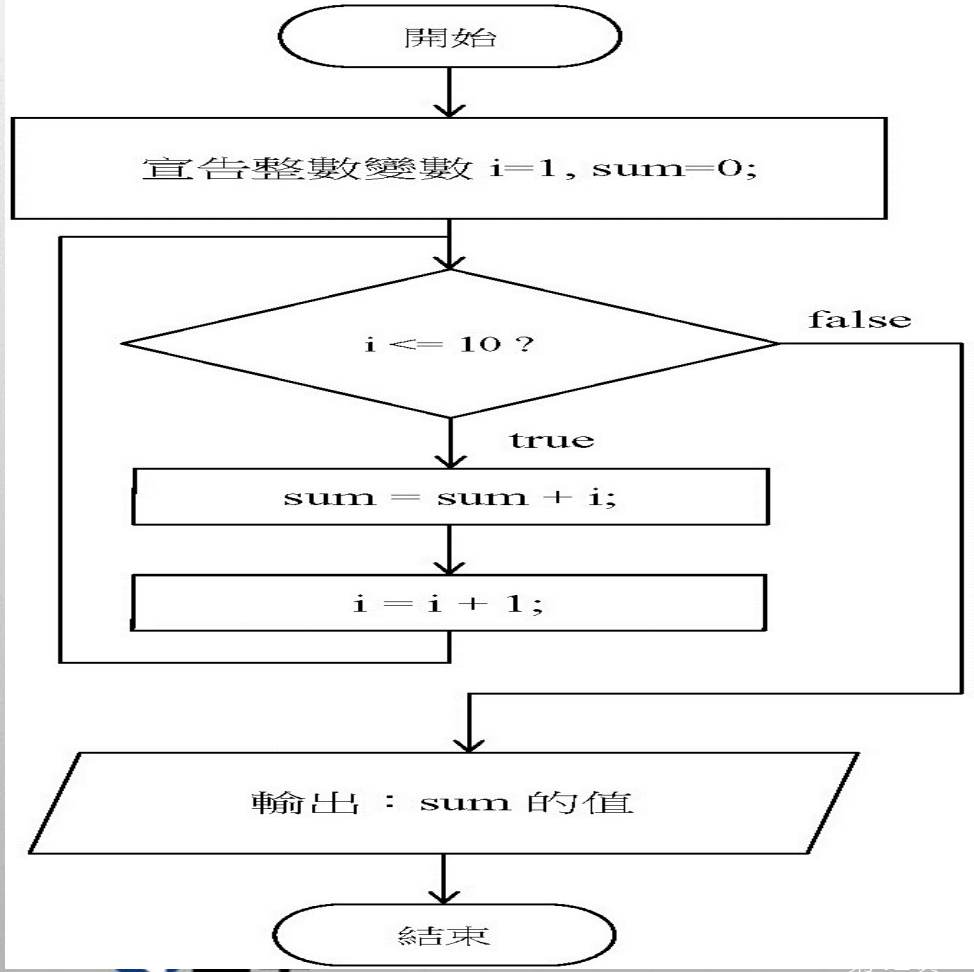
- 程式執行到迴圈結構「for」時，程式執行的步驟如下：
  1. 設定迴圈變數的初值
  2. 檢查進入迴圈結構「for」的條件結果是否為「true」？若為「true」，則執行步驟3；否則跳到迴圈結構「for」外的第一列敘述。
  3. 執行迴圈結構for「 { } 」內的程式敘述
  4. 增加(或減少) 迴圈變數的值，然後回到步驟2

範例 1-1：寫一程式，輸出1+2+…+10的結果。

(開啟D:\C#\ch05\Ex1_1\Ex1_1.csproj)

執行結果	1+2+…+10=55

範例1-2：寫一程式，使用迴圈結構「for」，輸出1+2+…+10的結果。(開啟D:\C#\ch05\Ex1_2\Ex1_2.csproj)



執行結果	1+2+…+10=55

■ 程式說明

1. 由迴圈結構for的「( )」中，知道迴圈變數「i」的初值=1，進入迴圈的條件為「i<=10」，及迴圈變數增量=1(因i=i+1)
   - 利用這三個資訊，知道迴圈結構for「{ }」內的程式敘述，總共會執行10(=(10-1)/1+1)次，即執行了1 + 2+…+10的計算。直到i=11 時，才違反迴圈條件而跳離迴圈結構「for」
2. 因for的「{ }」內只有一列敘述，故「{ }」被省略
3. 若改成輸出1+2+…+100，程式只需將「i<=10」改成「i<=100」

■ 迴圈結構「while」的語法(一)如下：

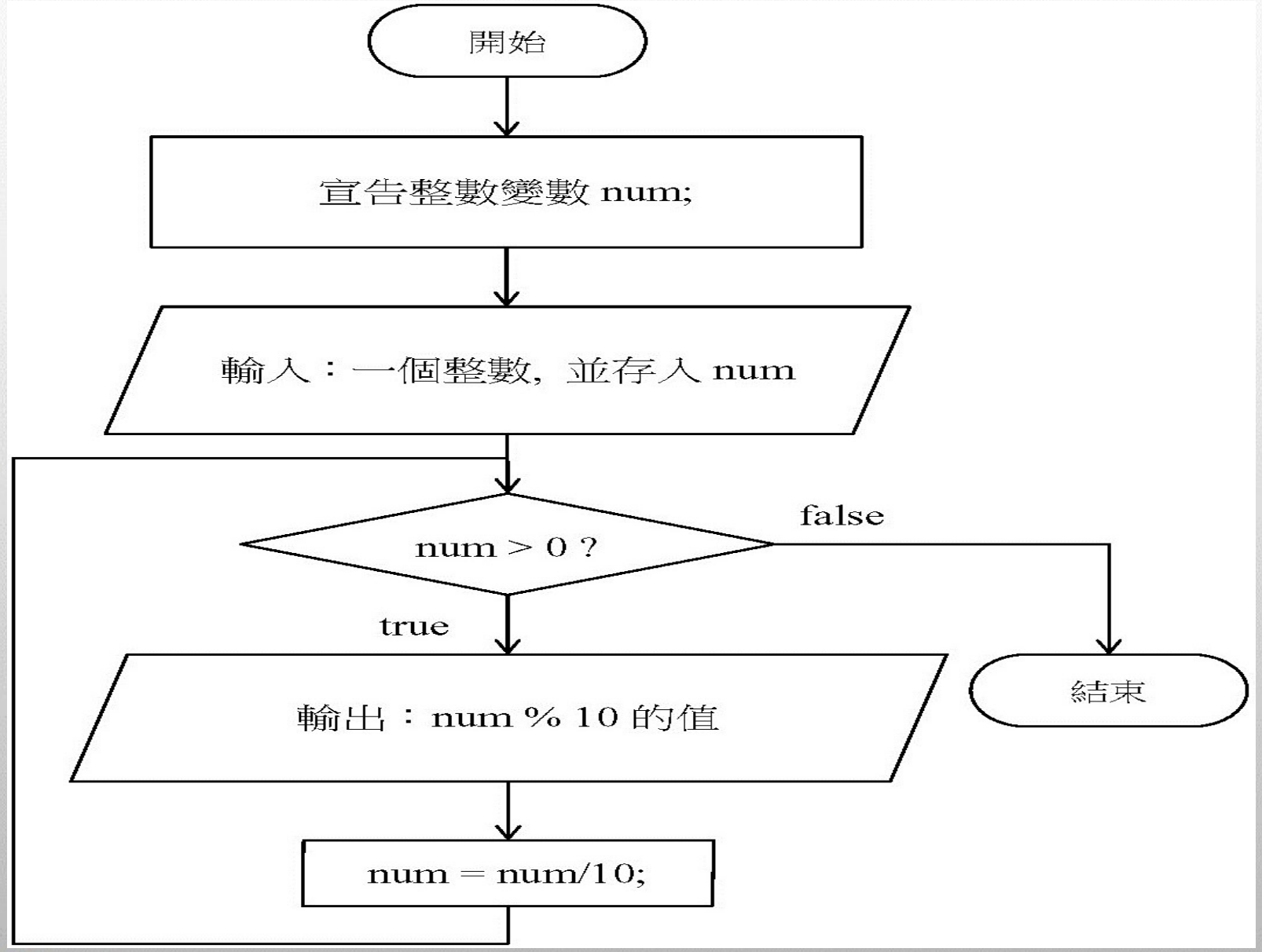
```
while (進入迴圈的條件)
  {
      程式敘述
      …
  }
```

■ 執行到迴圈結構「while」時，程式執行的步驟如下

1. 檢查進入迴圈結構「while」的條件結果是否為「true」？若為「true」，則執行步驟2；否則跳到迴圈結構「while」外的第一列敘述
2. 執行迴圈結構while「｛ ｝」內的程式敘述
3. 回到步驟1

範例3：寫一程式，輸入一正整數，然後將它倒過來輸出。(例：1234 → 4321)
(開啟D:\C#\ch05\Ex3\Ex3.csproj)

執行結果	輸入一正整數:1234 1234倒過來為4321


開始
宣告整數變數 num;
輸入：一個整數，並存入 num
num > 0 ?
false
true
輸出：num % 10 的值
結束
num = num/10;

■ 迴圈結構「while」的語法(二)如下：

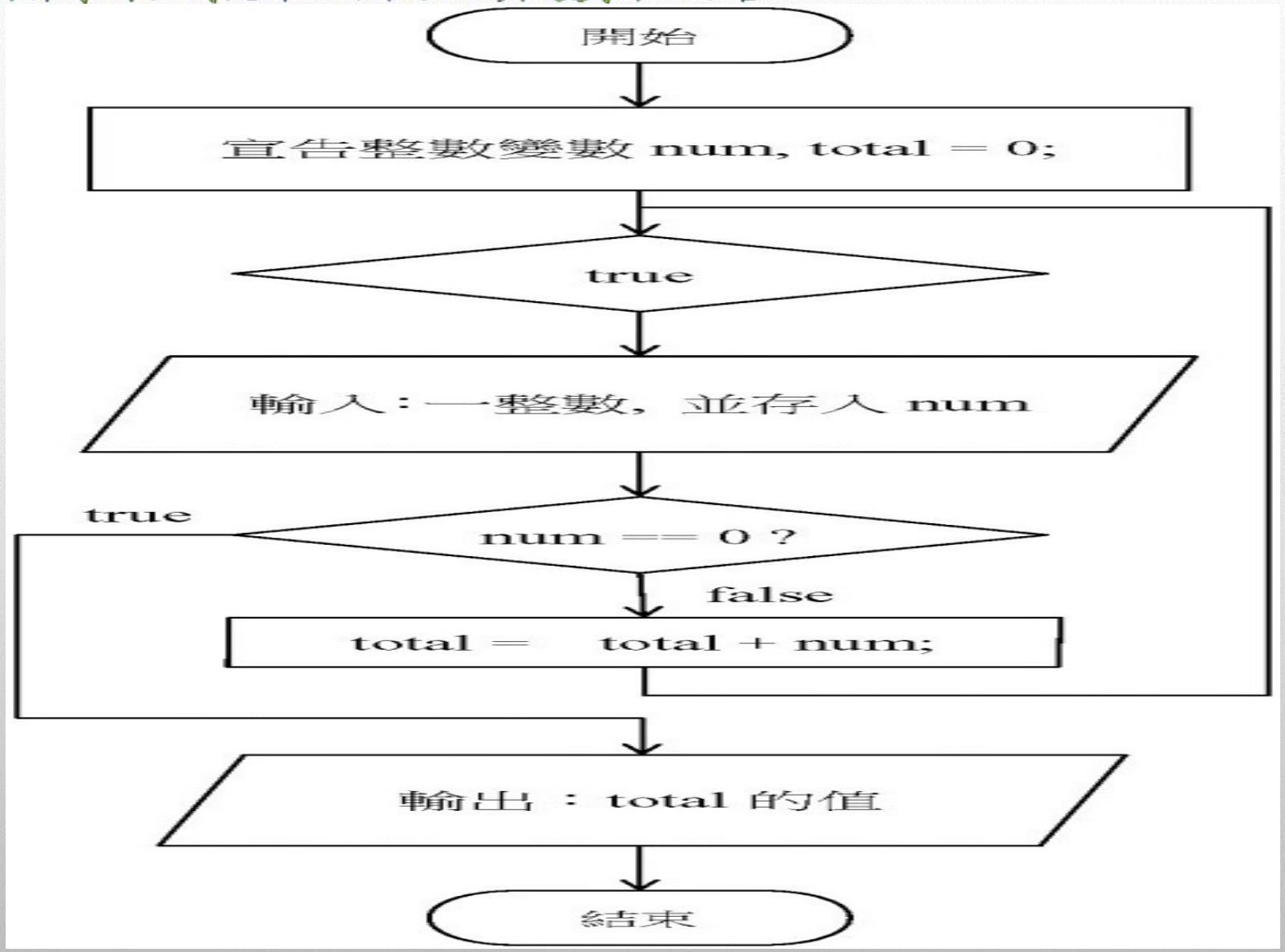
```
while (true)
  {
        程式敘述
        …
  }
```

- 程式執行到迴圈結構「while」時，程式會不斷地重複執行迴圈結構while「 { } 」內的程式敘述
- while「( )」內的「true」，表示迴圈的條件永遠為真

- 在while「{ }」內的程式敘述中，一定要包含「選擇結構」及「break;」敘述，否則會造成無窮迴圈或違反迴圈結構重複執行的精神。
  - 若選擇結構中的條件結果為「true」，則執行「break;」敘述，並離開迴圈結構「while」；否則繼續重複執行while「{ }」內的程式敘述

範例4：寫一個程式，連續將整數一個一個輸入，直到輸入0才表示結束輸入，最後輸出總和。
(開啟D:\C#\ch05\Ex4\Ex4.csproj)

執行結果	連續將整數一個一個輸入，直到輸入0才表示結束輸入: 10 20 30 0 總和=60


開始
宣告整數變數 num, total = 0;
true
輸入：一整數，並存入 num
true
num == 0 ?
false
total = total + num;
輸出 ： total 的值
結束

5-2-2 後測式迴圈結構

■ 迴圈結構「do while」的語法如下：

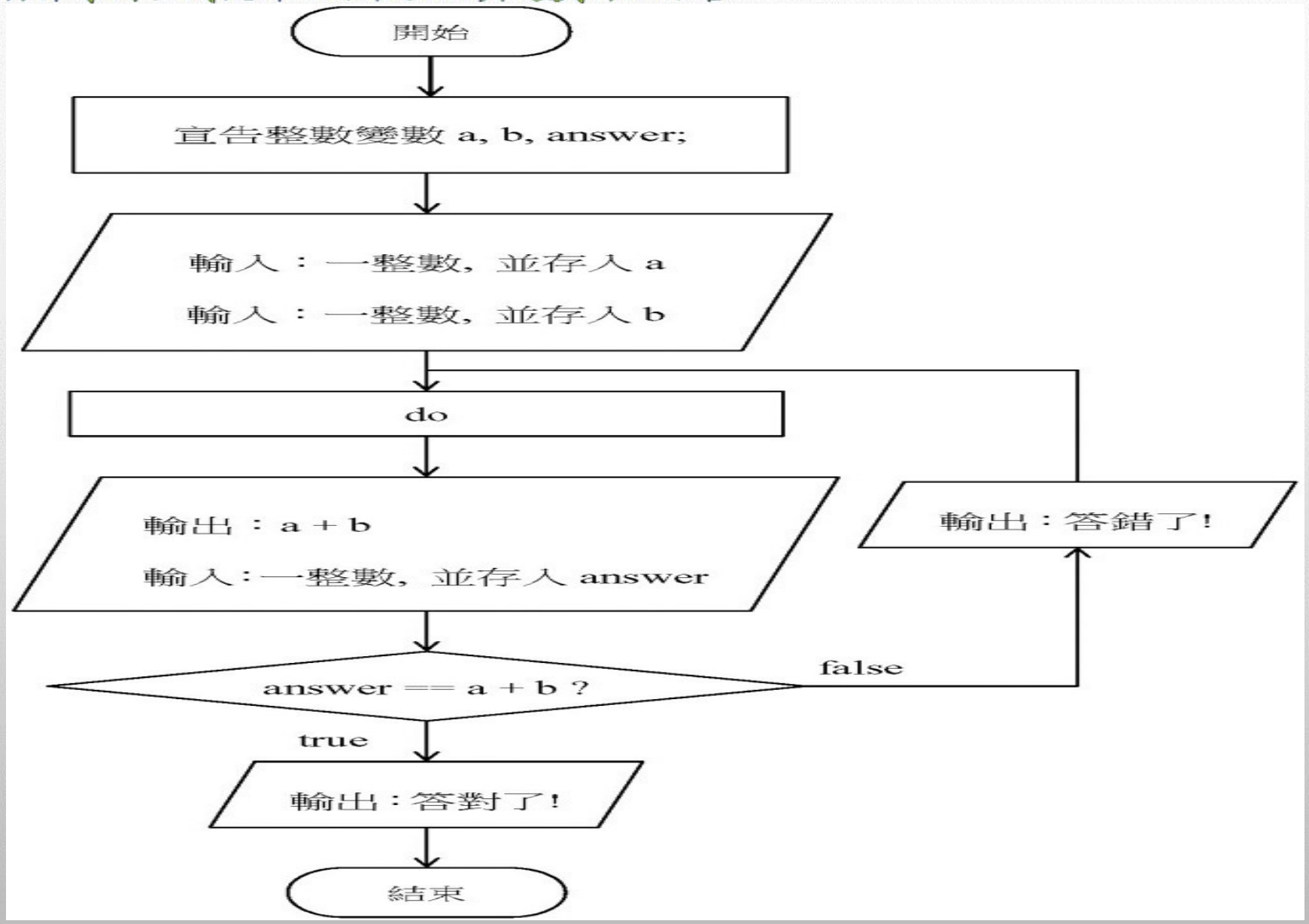
```
do
 {
        程式敘述
        …
 }
while(進入迴圈的條件)；
```

5-2-2 後測式迴圈結構

- 迴圈結構「do while」的執行步驟如下：
  1. 程式會直接執行do while「{ }」內的程式敘述
  2. 「{ }」內的程式敘述執行完畢，會檢查進入迴圈的條件結果是否為「true」？若為「 true 」，則執行步驟1；否則跳到圈結構「do while」外的第一列程式敘述

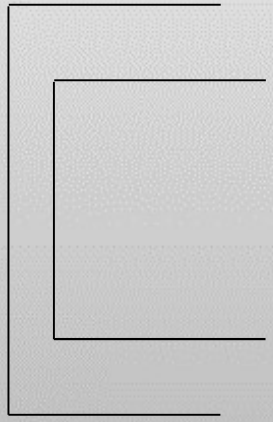
範例5：寫一程式，輸入整數a 及b，然後再讓使用者回答a+b 的值。若答對，則輸出答對了；否則輸出答錯了，並讓使用者繼續回答。(開啟 D:\C#\ch05\Ex5\Ex5.csproj

執行結果	輸入第1個整數:10 輸入第2個整數:20 10+20=30 答對了!

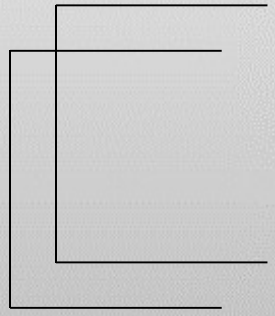

開始
宣告整數變數 a, b, answer;
輸入：一整數，並存入 a
輸入：一整數，並存入 b
do
輸出：a + b
輸入：一整數，並存入 answer
answer == a + b ?
false
輸出：答錯了!
true
輸出：答對了!
結束

5-2-3 巢狀迴圈

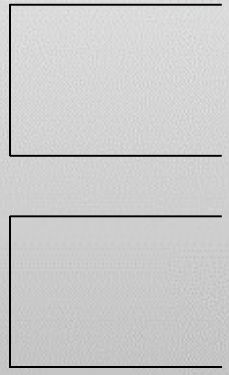
- 巢狀迴圈：一層迴圈結構中還有其他迴圈結構的架構
  - 當問題具有重複執行某些特定的敘述，且這些特定的敘述受到兩個或兩個以上的因素而改變，此時使用巢狀迴圈結構來撰寫是最適合的用法



巢狀迴圈



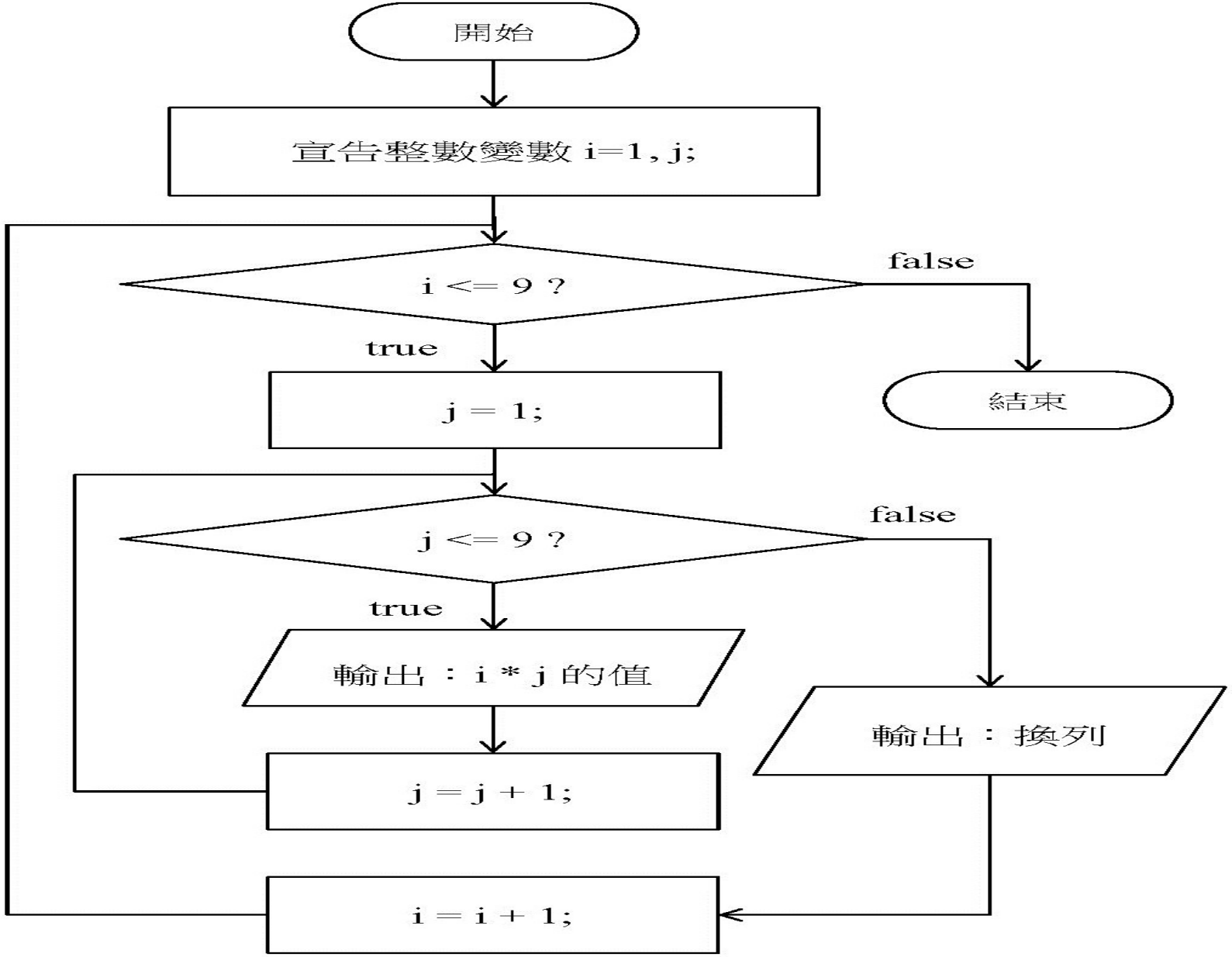
非巢狀，且為
錯誤的迴圈寫法



非巢狀，且為
兩個單層迴圈

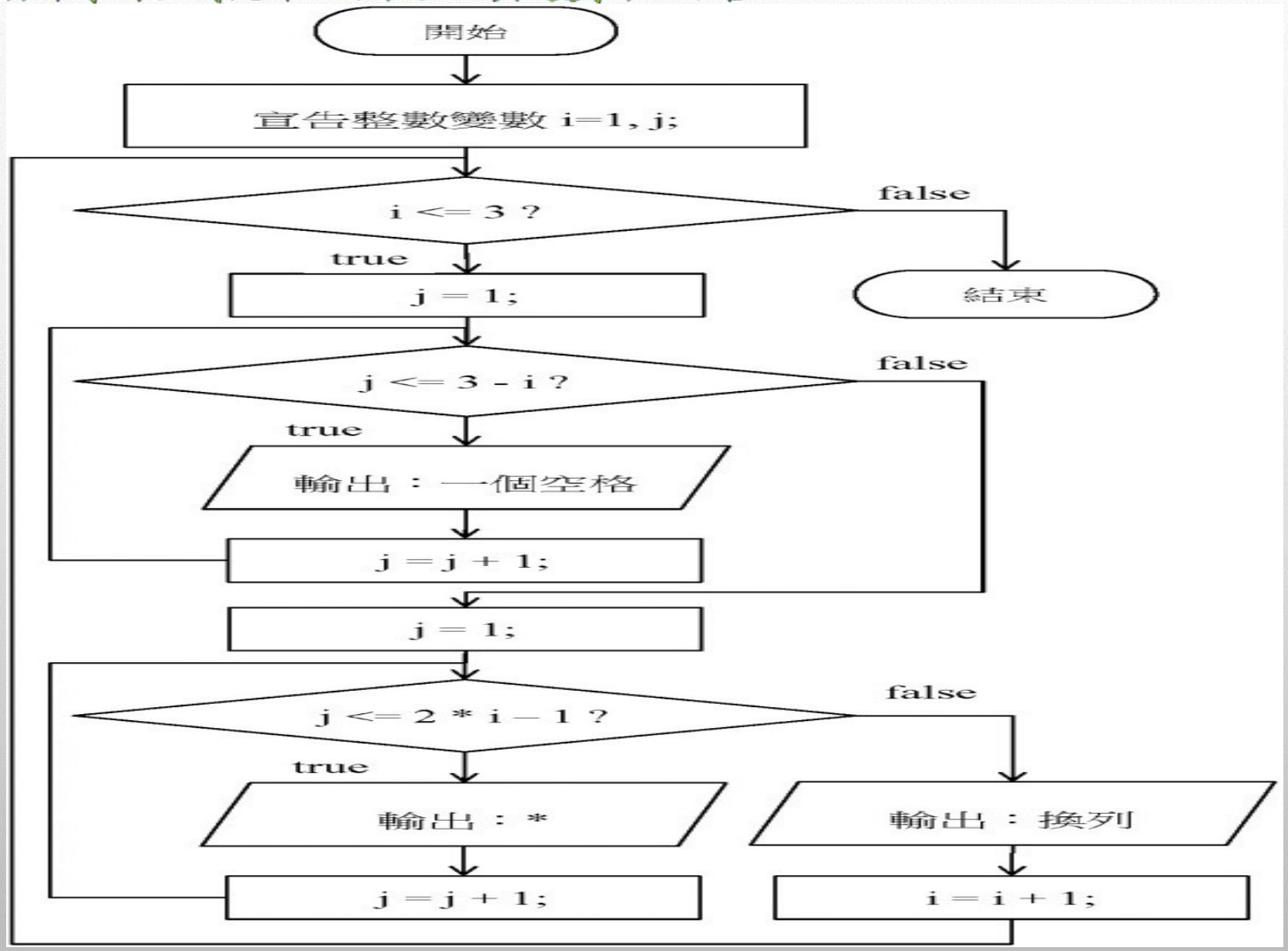
範例6：寫一程式，輸出九九乘法。
(開啟D:\C#\ch05\Ex6\Ex6.csproj)

1x1=1	1x2=2	1x3=3	1x4=4	1x5=5	1x6=6	1x7=7	1x8=8	1x9=9
2x1=2	2x2=4	2x3=6	2x4=8	2x5=10	2x6=12	2x7=14	2x8=16	2x9=18
3x1=3	3x2=6	3x3=9	3x4=12	3x5=15	3x6=18	3x7=21	3x8=24	3x9=27
4x1=4	4x2=8	4x3=12	4x4=16	4x5=20	4x6=24	4x7=28	4x8=32	4x9=36
5x1=5	5x2=10	5x3=15	5x4=20	5x5=25	5x6=30	5x7=35	5x8=40	5x9=45
6x1=6	6x2=12	6x3=18	6x4=24	6x5=30	6x6=36	6x7=42	6x8=48	6x9=54
7x1=7	7x2=14	7x3=21	7x4=28	7x5=35	7x6=42	7x7=49	7x8=56	7x9=63
8x1=8	8x2=16	8x3=24	8x4=32	8x5=40	8x6=48	8x7=56	8x8=64	8x9=72
9x1=9	9x2=18	9x3=27	9x4=36	9x5=45	9x6=54	9x7=63	9x8=72	9x9=81


開始
宣告整數變數 i=1, j;
i <= 9 ?
false
true
結束
j = 1;
j <= 9 ?
false
true
輸出：i * j 的值
輸出：換列
j = j + 1;
i = i + 1;

範例7：寫一程式，用「*」模擬金字塔(單面，高度3，寬度5)圖案。
(開啟D:\C#\ch05\Ex7\Ex7.csproj

執行結果	* *** *****


開始
宣告整數變數 i=1, j;
i <= 3 ?
false
true
結束
j = 1;
j <= 3 - i ?
false
true
輸出：一個空格
j = j + 1;
j = 1;
j <= 2 * i - 1 ?
false
true
輸出：*
輸出：換列
j = j + 1;
i = i + 1;

5-3 break與continue敘述

- 在「for」、「while」及「do while」這三種迴圈結構中，一般情況是在違反進入迴圈的條件時，才會結束迴圈的運作。
  - 若問題除了具有重複執行某些特定的敘述特性外，還包括某些例外性時，則在這三種迴圈結構中必須加入「break;」(目的：符合某個例外條件時，跳出迴圈結構)或「continue;」(目的：符合某個例外條件時，不執行某些敘述)，才能達成問題的需求

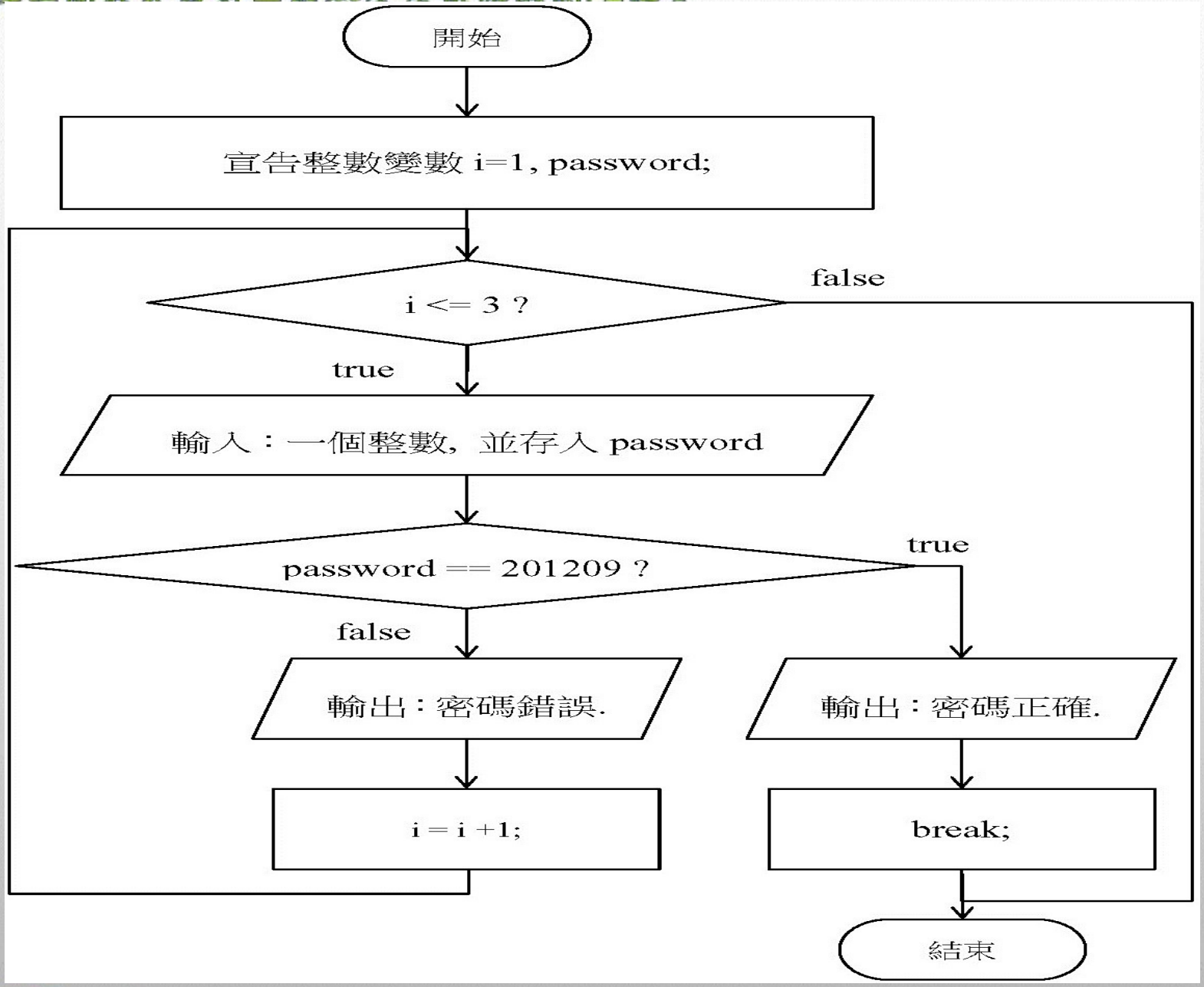
- break; 及 continue; 必須撰寫在選擇結構的敘述中(即，撰寫在某個條件底下)，
  - 否則 break; 及 continue; 敘述底下的程式碼會出現綠色鋸齒狀線條，表示「偵測到不會執行到的程式碼」(即break; 及continue; 敘述底下的程式碼不會被執行)

5-3-1 break敘述

- 「break;」敘述除了用在選擇結構「switch」(請參考「4-2-4 switch選擇結構」)外，還可用在迴圈結構
- 當程式執行到迴圈結構內的「break;」敘述時，程式會跳出迴圈結構，並執行迴圈結構外的第一列敘述，不再回頭重複執行迴圈結構「{ }」內的敘述
- 注意，當「break;」敘述用在巢狀迴圈結構內時，它一次只能跳出一層迴圈結構(離它最近的那層迴圈結構)，而不是跳出整個巢狀迴圈結構外。

範例8：寫一程式，模擬密碼驗證(假設密碼為201209)，最多可以輸入三次密碼。若輸入正確，則輸出密碼正確，否則輸出密碼錯誤。
(開啟D:\C#\ch05\Ex8\Ex8.csproj

執行結果	輸入密碼:123456 密碼錯誤 輸入密碼:201209 密碼正確


開始
宣告整數變數 i=1, password;
i <= 3 ?
false
true
輸入：一個整數，並存入 password
password == 201209 ?
true
false
輸出：密碼錯誤.
輸出：密碼正確.
i = i +1;
break;
結束

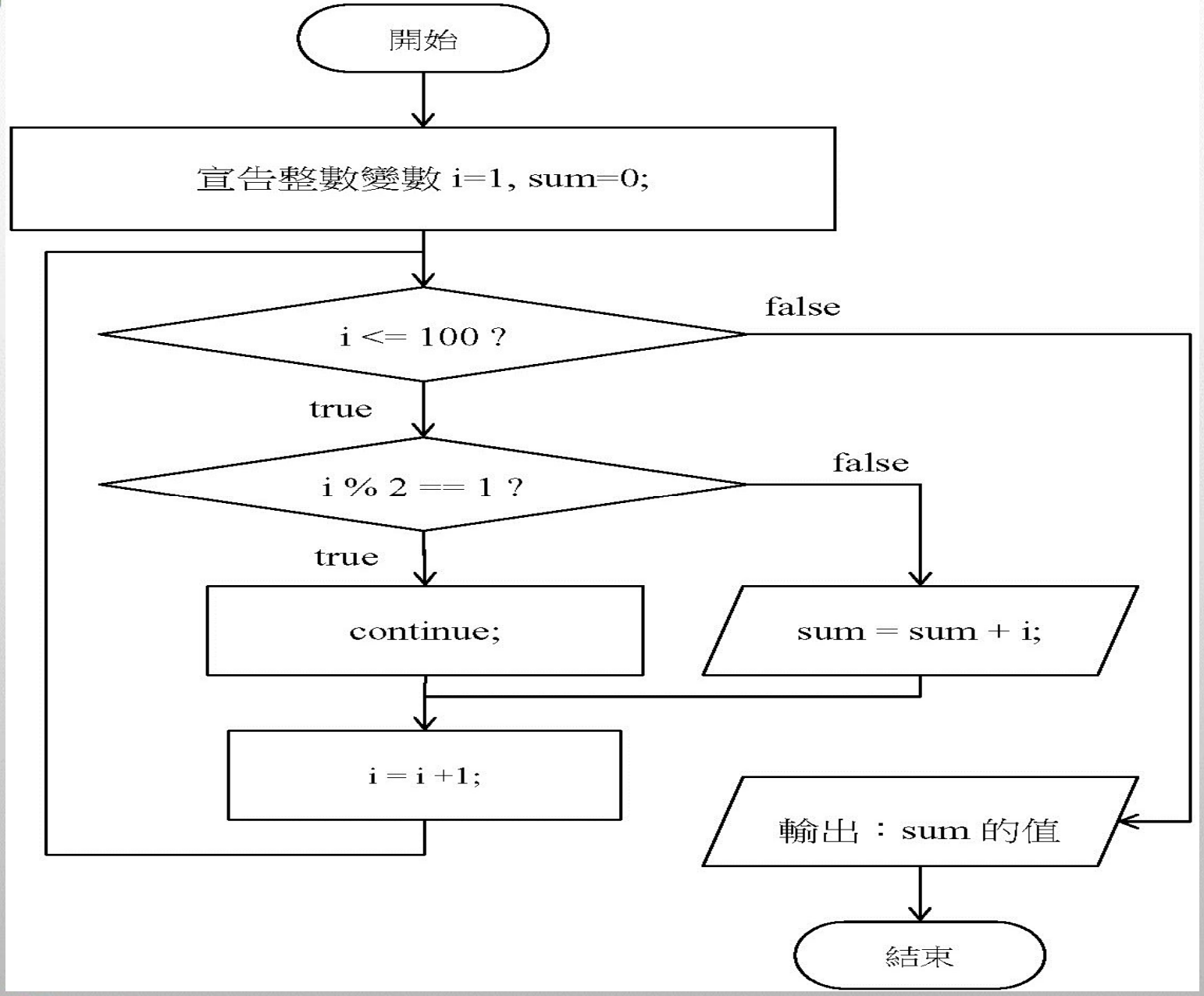
5-3-2 continue敘述

- 「continue;」 敘述的目的，是不執行迴圈結構內的某些敘述
- 「continue;」用在「for」、「while」及「do while」三種迴圈結構內部所產生的流程差異：
  - 在迴圈結構「for」內使用「continue;」：
    - 執行到「continue;」，程式會跳到該層迴圈結構for「( )」內的第三部分，執行迴圈變數增(或減)量

- 在迴圈結構「while」內使用「continue;」：
  - 執行到「continue;」，程式會跳到該層迴圈結構while「( )」內，檢查迴圈的條件結果是否為「true」。
- 在迴圈結構「do while」內使用「continue;」：
  - 執行到「continue;」，程式會跳到該層迴圈結構do while「( )」內，檢查迴圈的條件結果是否為「true」。

範例10：寫一程式，利用「continue;」指令的特性，計算1到100之間的偶數和
(開啟D:\C#\ch05\Ex10\Ex10.csproj)

執行結果	1到100之間的偶數和=2550


開始
宣告整數變數 i=1, sum=0;
i <= 100 ?
false
true
i % 2 == 1 ?
false
true
continue;
sum = sum + i;
i = i +1;
輸出：sum 的值
結束

■ 程式說明

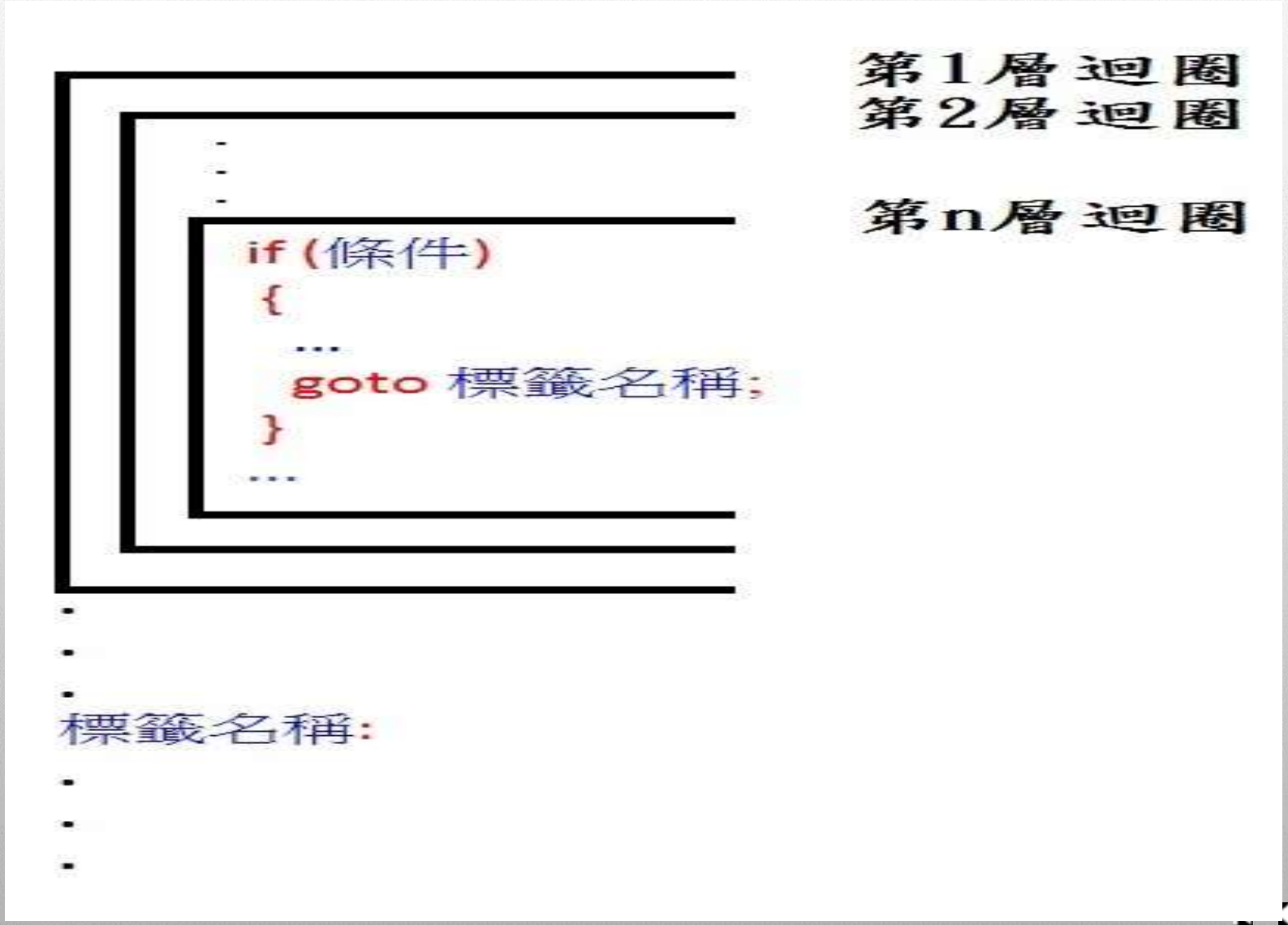
1. 迴圈結構「for」執行100次
   - ■ 只有i=2，4，…，100有被加到
   - ■ 當i=1，3…，99時，符合if「( )」內的條件，會執行「continue;」敘述，接著程式執行該層for「( )」內的第三部分，「sum=sum+i; 」敘述沒有被執行
2. 第14列到第20 列，可改寫成：

```
for (i=1;i<=100;i++)
  {
     if (i%2==0)
        sum=sum+i;
  }
```

5-4 goto陳述式

- 在多層的迴圈結構中，若想跳出此特定層的迴圈結構，則在此特定層的迴圈結構外，必須有「標籤名稱:」敘述，且在此特定層的迴圈結構中，使用「goto 標籤名稱;」陳述式
- 「 goto 標籤名稱;」必須撰寫在選擇結構中(即，撰寫在某個條件底下)
  - 否則 goto 標籤名稱; 底下的程式碼會出現綠色鋸齒狀線條，表示「偵測到不會執行到的程式碼」(即 goto 標籤名稱; 底下的程式碼不會被執行)

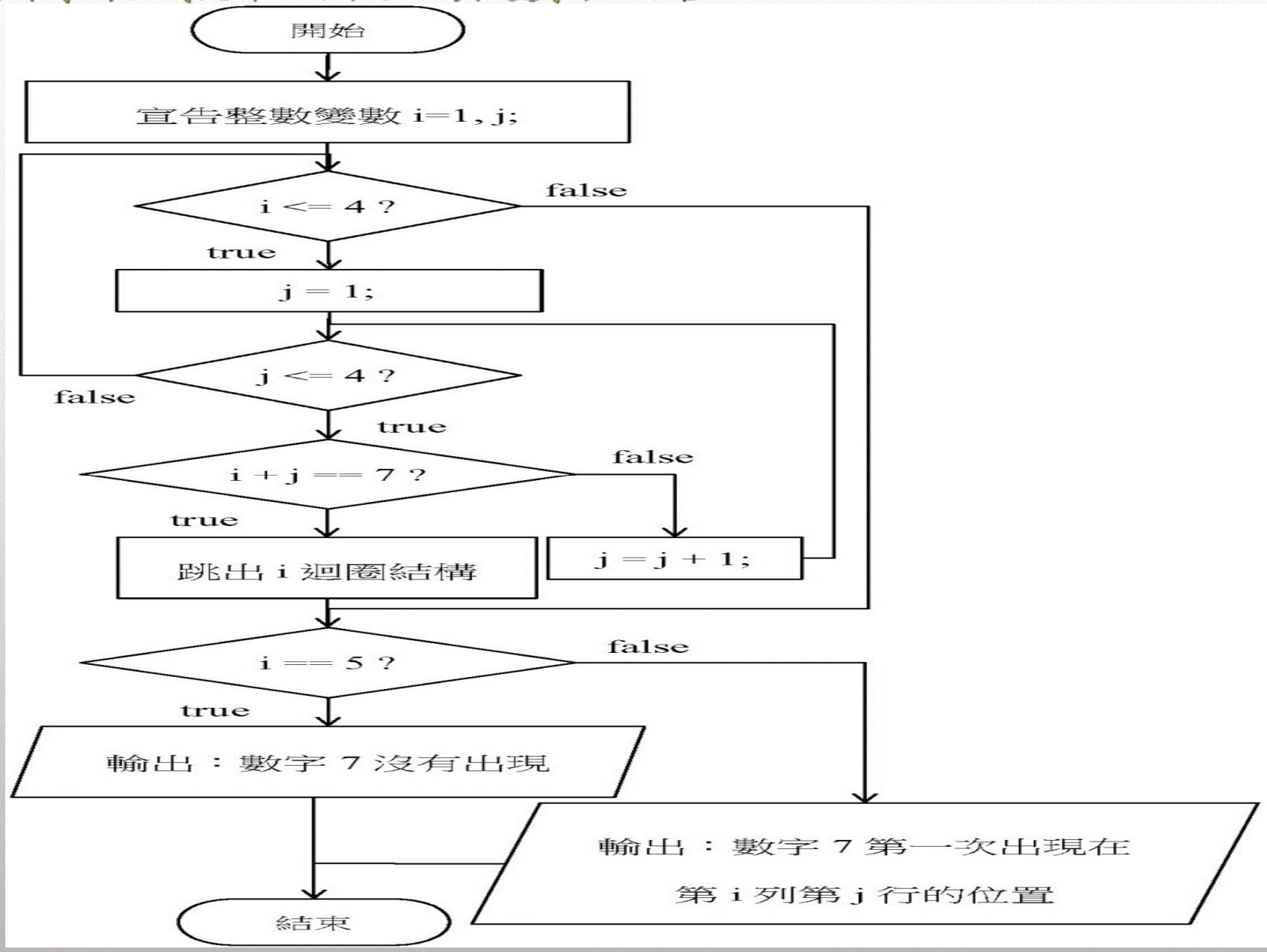
■goto 陳述式的撰寫位置示意圖:



範例 11：寫一程式，判斷在四列四行的資料中，數字7是否有出現過。
(開啟D:\C#\ch05\Ex11\Ex11.csproj)

2	3	4	5
3	4	5	6
4	5	6	7
5	6	7	8

執行結果	四列四行的資料中,數字7第一次出現在第3列第4行.


開始
宣告整數變數 i=1, j;
i <= 4 ?
false
true
j = 1;
j <= 4 ?
false
true
i + j == 7 ?
false
true
跳出 i 迴圈結構
j = j + 1;
i == 5 ?
false
true
輸出：數字 7 沒有出現
輸出：數字 7 第一次出現在
第 i 列第 j 行的位置
結束

5-5　發現問題

■ 範例 12(浮點數的缺失):寫一程式，判斷 0.1+0.1+0.1 與 0.3 是否相等。

(開啟D:\C#\ch05\Ex12\Ex12.csproj)

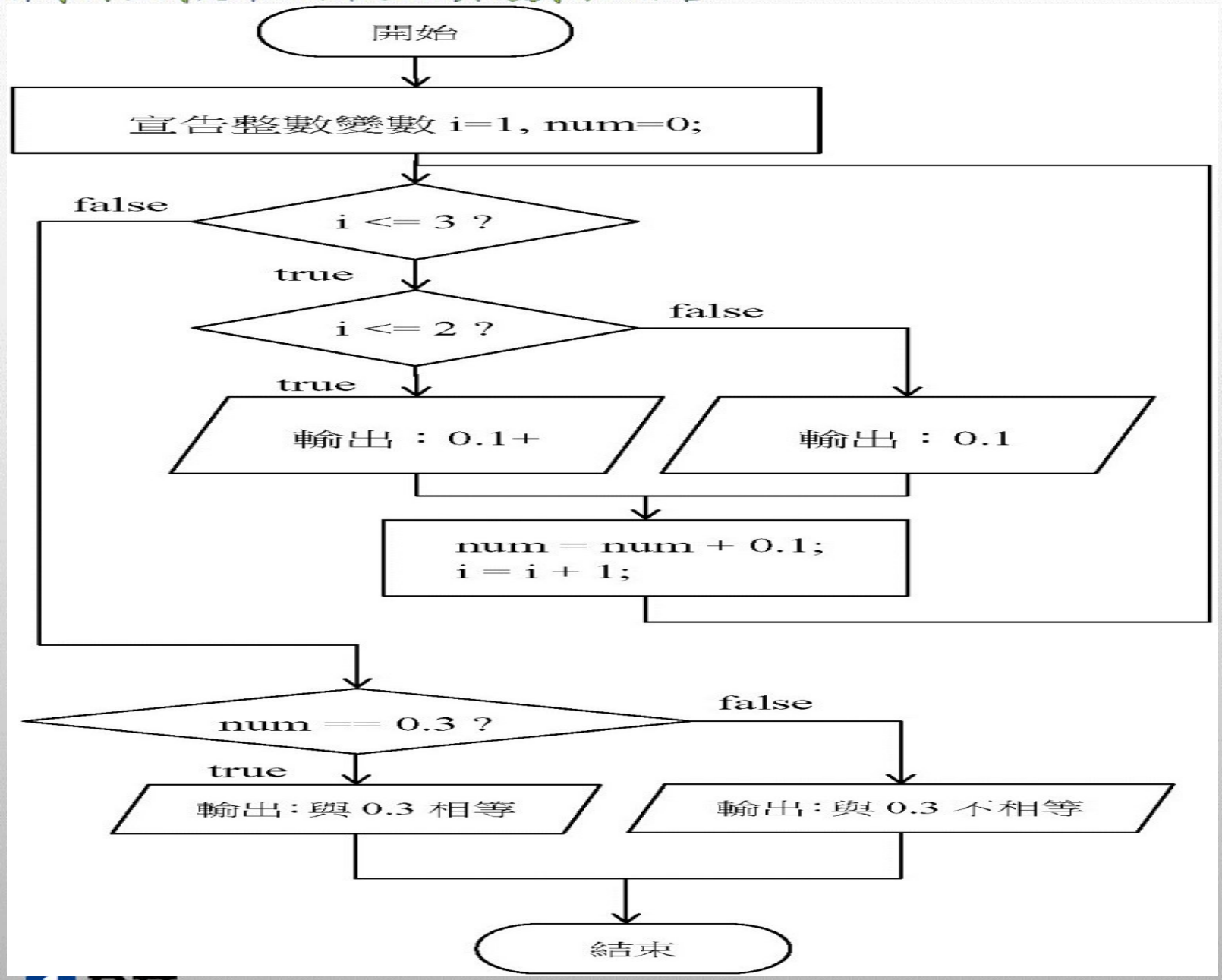
執行結果	0.1+0.1+0.1 與 0.3 不相等

5-5　發現問題

- 程式說明
  - 浮點數儲存入記憶體會產生誤差，造成浮點數運算時所得到的結果與我們認為的結果有所不同。
  - 若需判斷兩個浮點數是否相等，則改為判斷兩個整數是否相等：num=num+0.1; 改成 num=num+1;

    if (num == 0.3) 改成 if (num == 3)

    結果：相等


開始
宣告整數變數 i=1, num=0;
false
i <= 3 ?
true
i <= 2 ?
false
true
輸出 ： 0.1+
輸出 ： 0.1
num = num + 0.1;
i = i + 1;
num == 0.3 ?
false
true
輸出：與 0.3 相等
輸出：與 0.3 不相等
結束