03 CHAPTER

流程控制

3-1 認識流程控制

我們在前兩章所示範的例子都是很單純的程式，它們的執行方向都是從第一行敘述開始，由上往下依序執行，不會轉彎或跳行，但事實上，大部分的程式並不會這麼單純，它們可能需要針對不同的情況做不同的處理，以完成更複雜的任務，於是就需要**流程控制**（flow control）來協助控制程式的執行方向。

PHP 的流程控制分成下列兩種類型：

- **選擇結構**（decision structure）：用來檢查條件式，然後根據結果為 TRUE 或 FALSE 執行不同的敘述，PHP 提供如下的選擇結構：
    - if (if…、if…else…、if…elseif…)
    - switch
- **迴圈結構**（loop structure）：用來重複執行某些敘述，PHP 提供如下的迴圈結構：
    - for
    - while
    - do…while
    - foreach

備註

流程控制通常需要借助於邏輯資料，以下是常見的型別與邏輯資料之間的關聯：

- 等於 0 的數值會被視為 FALSE，不等於 0 的數值會被視為 TRUE。
- 空字串 "" 與字串 "0" 會被視為 FALSE，其它字串會被視為 TRUE。
- 沒有元素的陣列和沒有成員的物件會被視為 FALSE。
- NULL 會被視為 FALSE。

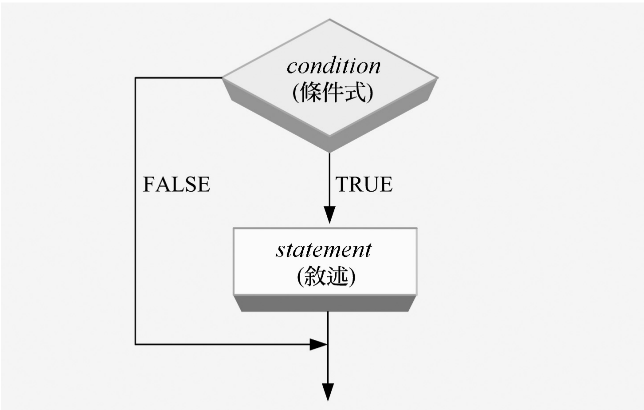
3-2 if

if 選擇結構可以用來檢查條件式，然後根據結果為 TRUE 或 FALSE 執行不同的敘述，又分為 if⋯、if⋯else⋯、if⋯elseif⋯等形式。

3-2-1 if⋯

```
if (condition) statement;
```

這種形式的意義是「若⋯就⋯」，屬於單向選擇。*condition* 是一個條件式，結果為布林型別，若 *condition* 傳回 TRUE，就執行 *statement* (敘述)；若 *condition* 傳回 FALSE，就離開 if 選擇結構，不會執行 *statement* (敘述)。



請注意，若 if 後面的 *statement* (敘述) 有很多行，那麼要加上大括號標示 *statement* (敘述) 的開頭與結尾，如下：

```
if (condition)
{
   statement1;
   statement2;
   …
   statementN;
}
```

隨堂練習

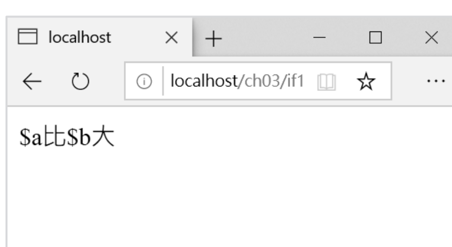
撰寫如下網頁，然後啟動瀏覽器執行網頁，看看執行結果為何？

\ch03\if1.php

```
<!DOCTYPE html>
<html>
  <head>
    <meta charset="utf-8">
  </head>
  <body>
    <?php
      $a = 20;
      $b = 10;
      if ($a > $b)
        echo '$a 比$b 大';
    ?>
  </body>
</html>
```

【解答】

這個網頁的執行結果如下，由於變數 a 的值（20）大於變數 b 的值（10），故條件式（$a > $b）會傳回 TRUE，進而執行 if 後面的敘述 echo '$a 比$b 大';，在網頁上顯示「$a 比$b 大」，此處使用單引號字串的原因是為了不要進行變數剖析。

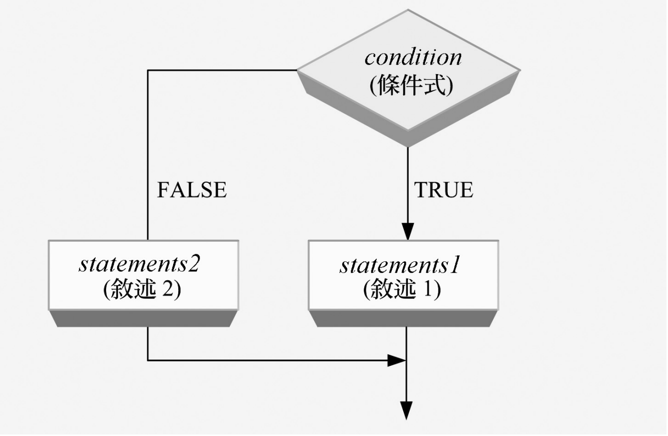


3-2-2 if…else…

```
if (condition)
{
   statements1;
}
else
{
   statements2;
}
```

這種形式的意義是「若…就…否則…」，屬於雙向選擇。*condition* 是一個條件式，結果為布林型別，若 *condition* (條件式) 傳回 TRUE，就執行 *statements1* (敘述 1)，否則執行 *statements2* (敘述 2)。

換句話說，若條件式成立，就執行 *statements1*，但不執行 *statements2*，若條件式不成立，就執行 *statements2*，但不執行 *statements1*。和單向 if…比起來，雙向 if…else…是比較實用的。



隨堂練習

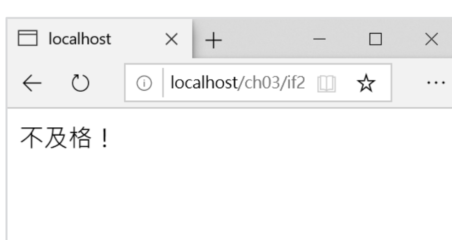
撰寫如下網頁，然後啟動瀏覽器執行網頁，看看執行結果為何？

\ch03\if2.php

```
<!DOCTYPE html>
<html>
  <head><meta charset="utf-8"></head>
  <body>
    <?php
      $score = 59;
      if ($score > 60)
        echo '及格！';
      else
        echo '不及格！';
    ?>
  </body>
</html>
```

【解答】

這個網頁的執行結果如下，由於變數 score 的值為 59，小於 60，故條件式（$score > 60）會傳回 FALSE，進而執行 else 後面的敘述，在網頁上顯示「不及格！」；相反的，當變數 score 的值大於 60 時，條件式（$score > 60）會傳回 TRUE，進而執行條件式後面的敘述，在網頁上顯示「及格！」。

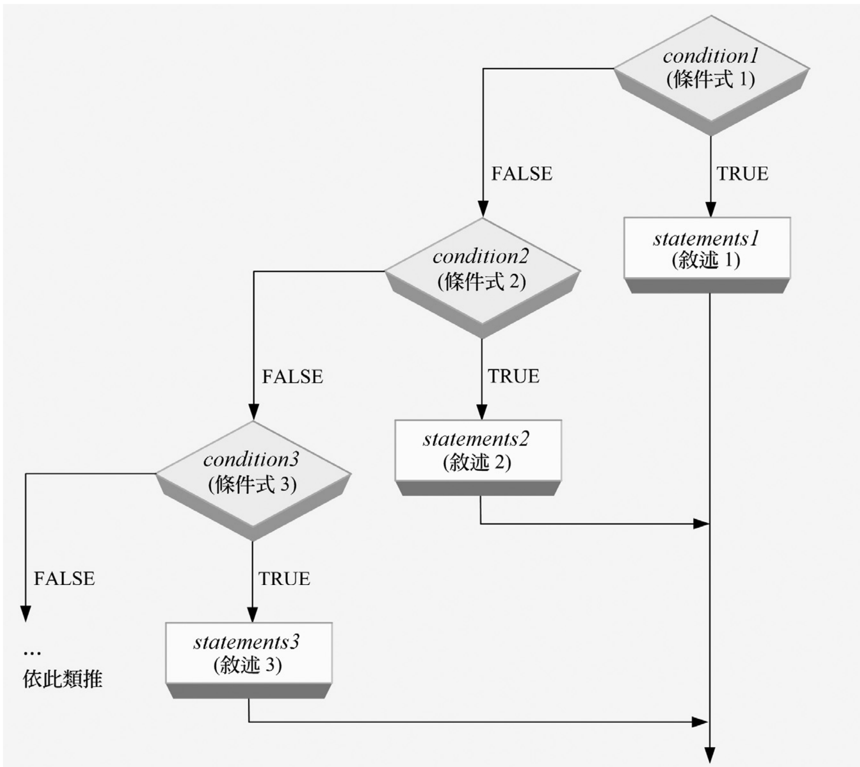


3-2-3 if…elseif…

```
if (condition1)
{
   statements1;
}
elseif (condition2)
{
   statements2;
}
elseif (condition3)
{
   statements3;
}
…
else
{
   statementsN+1;
}
```

這種形式的意義是「若…就…否則 若…」，屬於多向選擇。前面所介紹的 if… 和 if…else… 形式都只能處理一個條件式，而這種形式可以處理多個條件式。

一開始先檢查 *condition1* (條件式 1)，若 *condition1* 傳回 TRUE，就執行 *statements1* (敘述 1)，否則檢查 *condition2* (條件式 2)，若 *condition2* 傳回 TRUE，就執行 *statements2* (敘述 2)，否則檢查 *condition3* (條件式 3)，…，依此類推。若所有條件式皆不成立，就執行 else 後面的 *statementsN+1* (敘述 N+1)，所以 *statements1* ~ *statementsN+1* 只有一組會被執行。



隨堂練習

撰寫如下網頁，然後啟動瀏覽器執行網頁，看看執行結果為何？

\ch03\if3.php (下頁續 1/2)

```
<!DOCTYPE html>
<html>
  <head>
    <meta charset="utf-8">
  </head>
  <body>
    <?php
      $score = 85;
```

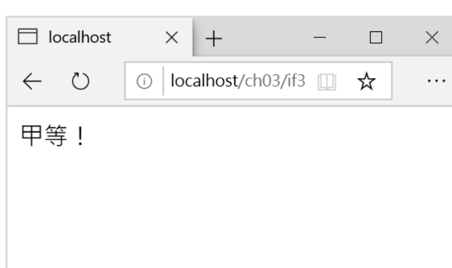
\ch03\if3.php (接上頁 2/2)

```
        if ($score >= 90)
          echo '優等！';
        elseif ($score < 90 && $score >= 80)
          echo '甲等！';
        elseif ($score < 80 && $score >= 70)
          echo '乙等！';
        elseif ($score < 70 && $score >= 60)
          echo '丙等！';
        else
          echo '不及格！';
      ?>
    </body>
  </html>
```

【解答】

這個網頁的執行結果如下，由於變數 score 的值為 85，小於 90 大於等於 80，故條件式（$score >= 90）會傳回 FALSE，於是跳到第一個 elseif，此時條件式（$score < 90 && $score >= 80）會傳回 TRUE，進而執行 elseif 後面的敘述，在網頁上顯示「甲等！」。

同理，當變數 score 的值小於 80 大於等於 70 時，條件式（$score >= 90）會傳回 FALSE，於是跳到第一個 elseif，此時條件式（$score < 90 && $score >= 80）會傳回 FALSE，於是又跳到第二個 elseif，此時條件式（$score < 80 && $score >= 70）會傳回 TRUE，進而執行 elseif 後面的敘述，在網頁上顯示「乙等！」。



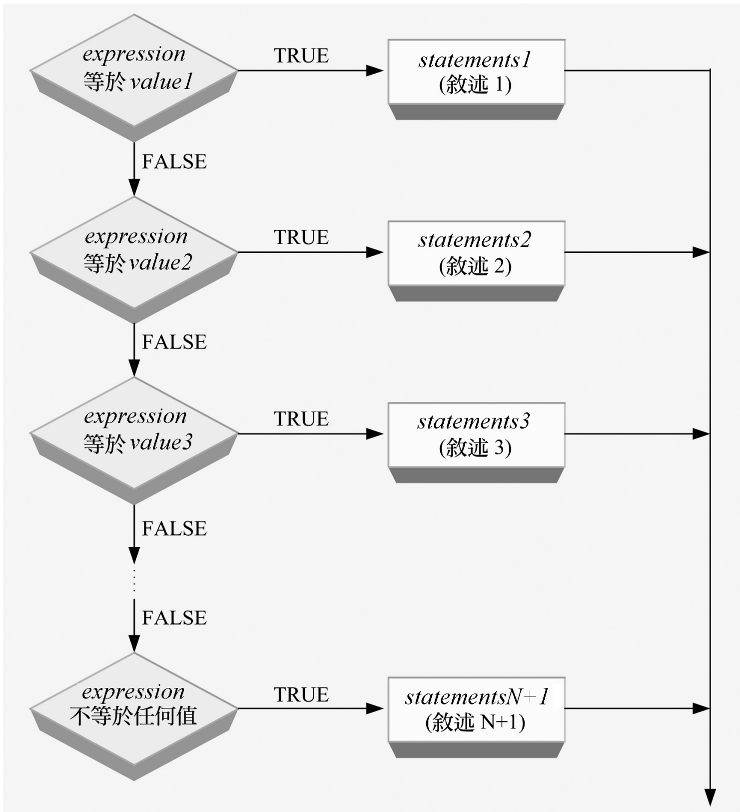
3-3 switch

switch 選擇結構可以根據變數的值而有不同的執行方向，您可以將它想像成一個有多種車位的車庫，這個車庫是根據車輛的種類來分配停靠位置，若進來的是小客車，就會進到小客車專屬的車位，若進來的是大貨車，就會進到大貨車專屬的車位，其語法如下：

```
switch(expression)
{
    case value1:
        statements1;
        break;
    case value2:
        statements2;
        break;
    …
    default:
        statementsN+1;
}
```

我們要先給 switch 選擇結構一個運算式 *expression* 當作判斷的對象，就好像上面比喻的車庫是以車輛的種類當作判斷的對象，接下來的 case 則是要寫出這個運算式可能的值，就好像車輛可能有數個種類。

switch 選擇結構會從第一個值 *value1* 開始做比較，看看是否和運算式 *expression* 的值相等，若相等，就執行其下的敘述 *statements1*，執行完畢後，break 敘述會令其離開 switch 選擇結構；相反的，若不相等，就換和第二個值 *value2* 做比較，看看是否和運算式 *expression* 的值相等，若相等，就執行其下的敘述 *statements2*，執行完畢後，break 敘述會令其離開 switch 選擇結構，…，依此類推；若沒有任何值和運算式 *expression* 的值相等，就執行 default 之下的敘述 *statementsN+1*，然後離開 switch 選擇結構。



請注意，若在 switch 選擇結構中省略了 break 敘述，將會執行其下的所有程式碼，直到抵達 switch 選擇結構的結尾。

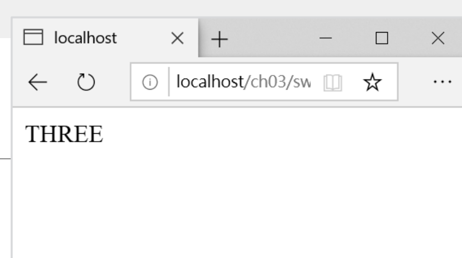
隨堂練習

撰寫一個 PHP 網頁，令它使用 switch 選擇結構根據變數 number 的值顯示對應的英文，當變數 number 的值為 1、2、3、4、5 時，就分別顯示「ONE」、「TWO」、「THREE」、「FOUR」、「FIVE」，當變數 number 的值不為 1 ~ 5 時，就顯示「超過範圍」。

【解答】

\ch03\switch.php

```
<!DOCTYPE html>
<html>
   <head>
      <meta charset="utf-8">
   </head>
   <body>
      <?php
         $number = 3;           //將變數 number 的值設定為 3
         switch($number)
         {
            case 1:             //當變數 number 的值為 1 時，就顯示「ONE」
               echo 'ONE';
               break;
            case 2:             //當變數 number 的值為 2 時，就顯示「TWO」
               echo 'TWO';
               break;
            case 3:             //當變數 number 的值為 3 時，就顯示「THREE」
               echo 'THREE';
               break;
            case 4:             //當變數 number 的值為 4 時，就顯示「FOUR」
               echo 'FOUR';
               break;
            case 5:             //當變數 number 的值為 5 時，就顯示「FIVE」
               echo 'FIVE';
               break;
            default:            //當變數 number 的值不為 1 ~ 5 時，就顯示「超過範圍」
               echo '超過範圍';
         }
      ?>
   </body>
</html>
```

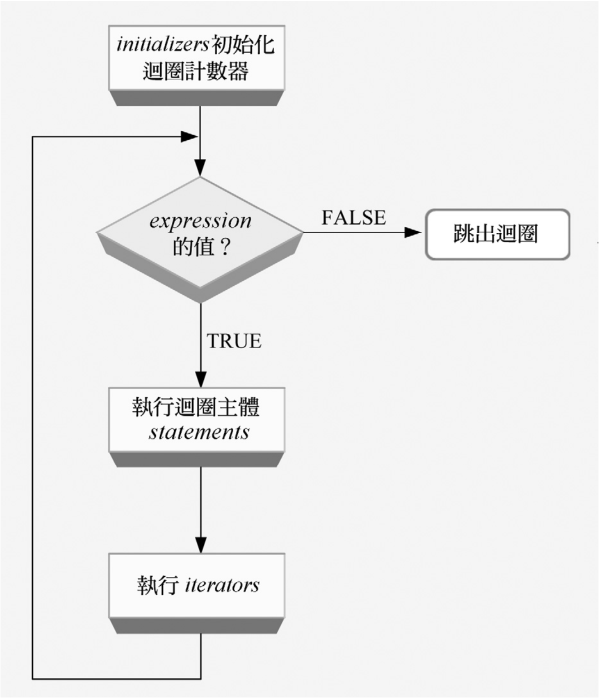


3-4 for

重複執行某個動作是電腦的專長之一，若每執行一次，就要撰寫一次敘述，那麼程式將會變得相當冗長，而 **for 迴圈**（for loop）就是用來解決重複執行的問題。舉例來說，假設要計算 1 加 2 加 3 一直加到 100 的總和，可以使用 for 迴圈逐一將 1、2、3、…、100 累加在一起，就會得到總和。

我們通常會使用控制變數來控制 for 迴圈的執行次數，所以 for 迴圈又稱為**計數迴圈**，而此控制變數則稱為**計數器**。

```
for (initializers; expression; iterators)
{
   statements;
   [break;]
   statements;
}
```

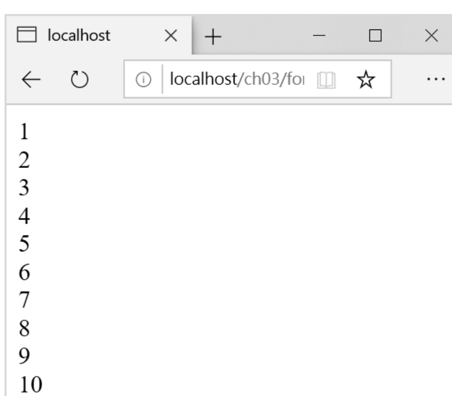


在進入 for 迴圈時，會先透過 *initializers* 初始化迴圈計數器，*initializers* 是使用逗號分隔的運算式清單或指派敘述，接著計算運算式 *expression* 的值，若為 FALSE，就離開 for 迴圈，若為 TRUE，就執行 for 迴圈內的 *statements*，完畢後跳回 for 處執行 *iterators*，再計算運算式 *expression* 的值，若為 FALSE，就離開 for 迴圈，若為 TRUE，就執行 for 迴圈內的 *statements*，完畢後跳回 for 處執行 *iterators*，再計算運算式 *expression* 的值，…，如此周而復始，直到運算式 *expression* 的值為 FALSE。

若要強制離開迴圈，可以加上 break 敘述；若 for 迴圈省略 *initializers*、*expression*、*iterators*，也就是 for (;;)，則會得到一個**無窮迴圈**（infinite loop）。

下面是一個例子 <\ch03\for1.php>，其中 $i = 1; 是宣告變數 i 做為計數器，初始值設定為 1，而 i <= 10; 是做為條件式，只要變數 i 小於等於 10 就會重複執行迴圈內的敘述，至於 i++ 則是迴圈每重複一次就將變數 i 的值遞增 1，因此，這個 for 迴圈總共執行 10 次 echo $i.'
'; 敘述，依序印出 1、2、3、…、10。

```
<?php
   for ($i = 1; $i <= 10; $i++)
      echo $i.'<br>';
?>
```



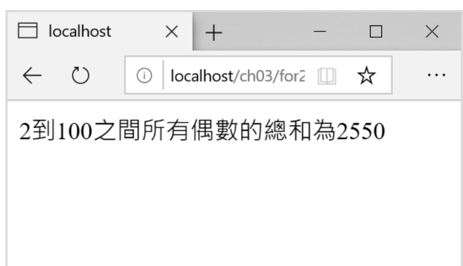
隨堂練習

撰寫一個 PHP 網頁，令它計算 2 到 100 之間所有偶數的總和，然後顯示出來。

【解答】

\ch03\for2.php

```
<?php
   $sum = 0;
   for ($i = 2; $i <= 100; $i+=2)
      $sum = $sum + $i;          //亦可寫成$sum += $i;
   echo '2 到 100 之間所有偶數的總和為'.$sum;
?>
```



這個 for 迴圈總共執行 50 次 $sum = $sum + $i; 敘述，其中變數 sum 用來存放總和，初始值為 0。

迴圈次數	= 右邊的 $sum	$i	= 左邊的 $sum
第 1 次	0	2	2
第 2 次	2	4	6
第 3 次	6	6	12
…	…	…	…
第 50 次	2450	100	2550

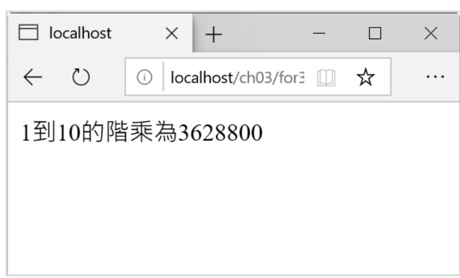
隨堂練習

撰寫一個 PHP 網頁，令它計算 1 到 10 的階乘，然後顯示出來。

【解答】

\ch03\for3.php

```
<?php
    $result = 1;
    for ($i = 1; $i <= 10; $i++)
        $result = $result * $i;          //亦可寫成$result *= $i;
    echo '1 到 10 的階乘為'.$result;
?>
```



這個 for 迴圈總共執行 10 次 $result = $result * $i; 敘述，其中變數 result 用來存放結果，初始值為 1。

迴圈次數	= 右邊的 $result	$i	= 左邊的 $result
第 1 次	1	1	1*1
第 2 次	1*1	2	1*1*2
第 3 次	1*1*2	3	1*1*2*3
…	…	…	…
第 10 次	1*1*2*3*4*5*6*7*8*9	10	1*1*2*3*4*5*6*7*8*9*10

break 敘述的用途

原則上，在終止條件成立之前，程式的控制權都不會離開 for 迴圈，不過，有時我們可能需要在迴圈內檢查其它條件，一旦符合該條件就強制離開迴圈，此時可以使用 break 敘述，下面是一個例子。

\ch03\break.php

```
01:<!DOCTYPE html>
02:<html>
03:   <head>
04:      <meta charset="utf-8">
05:   </head>
06:   <body>
07:      <?php
08:         $result = 1;
09:
10:         for ($i = 1; $i <= 10; $i++)
11:         {
12:            if ($i > 6) break;
13:            $result = $result * $i;
14:         }
15:
16:         echo $result;
17:      ?>
18:   </body>
19: </html>
```

猜猜看結果是多少呢？正確的答案是 720。事實上，這個 for 迴圈並沒有執行到 10 次，一旦第 12 行檢查到變數 i 大於 6 時（即變數 i 等於 7)，就會執行 break 敘述強制離開迴圈，故 result 的值為 1 * 1 * 2 * 3 * 4 * 5 * 6 = 720。

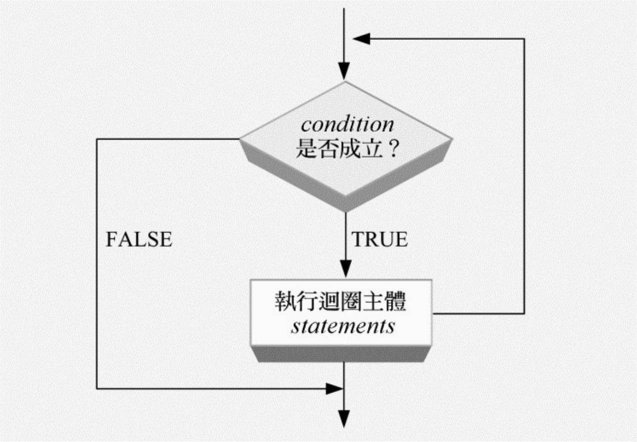
此外，break 敘述不僅可以用來強制離開 for 迴圈，也可以用來強制離開 while 迴圈、do…while 迴圈、函式等程式碼區塊。

3-5　while

有別於 for 迴圈是以計數器控制迴圈的執行次數，while 迴圈則是以條件式是否成立做為執行迴圈的依據，只要條件式成立，就會繼續執行迴圈，所以又稱為**條件式迴圈**（conditional loop）。

```
while(condition)
{
   statements;
   [break;]
   statements;
}
```

在進入 while 迴圈時，會先檢查 *condition*（條件式）是否成立，即是否為 TRUE，若結果為 FALSE 表示不成立，就離開迴圈，若結果為 TRUE 表示成立，就執行迴圈內的 *statements*（敘述），然後返回迴圈的開頭，再度檢查 *condition* 是否成立，…，如此周而復始，直到 *condition* 的結果為 FALSE 才離開迴圈。若要在中途強制離開迴圈，可以加上 break 敘述。



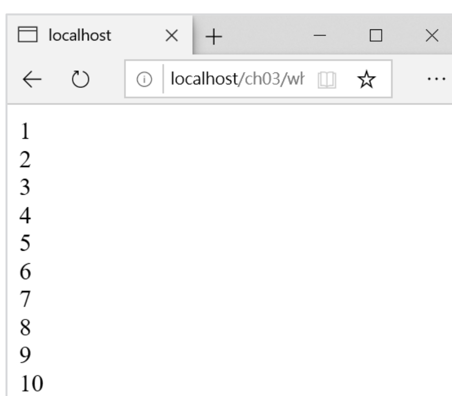
隨堂練習

撰寫一個 PHP 網頁，令它使用 while 迴圈顯示數字 1 到 10。

【解答】

\ch03\while.php

```
<!DOCTYPE html>
<html>
  <head>
    <meta charset="utf-8">
  </head>
  <body>
    <?php
      $i = 1;
      while ($i <= 10)
        echo $i++.'<br>';
    ?>
  </body>
</html>
```

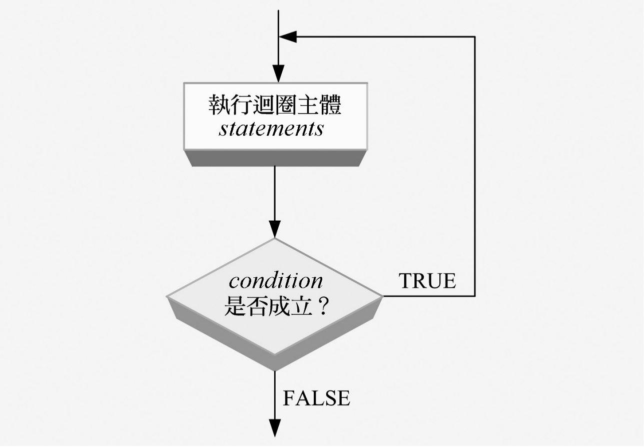


3-6 do…while

```
do
{
    statements;
    [break;]
    statements;
}while(condition);
```

do…while 迴圈也是以條件式是否成立做為執行迴圈的依據，在進入 do…while 迴圈時，會先執行迴圈內的 *statements* (敘述)，完畢後碰到 while，再檢查 *condition* (條件式)，若結果為 FALSE 表示不成立，就離開迴圈，若結果為 TRUE 表示成立，就返回 do，再度執行迴圈內的 *statements*，…，如此周而復始，直到 *condition* 的結果為 FALSE 才離開迴圈。

do…while 迴圈和 while 迴圈類似，主要的差別在於能夠確保敘述至少會被執行一次，即使條件式不成立。同樣的，若要在中途強制離開迴圈，可以加上 break 敘述。



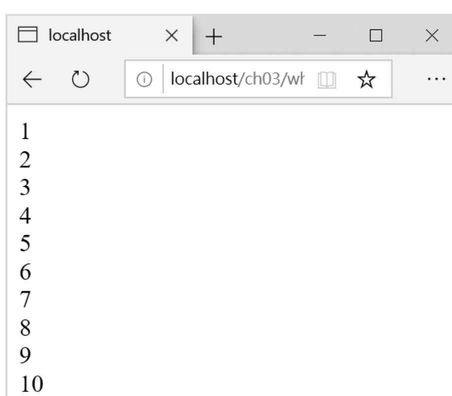
隨堂練習

使用 do…while 迴圈改寫第 3-5 節的隨堂練習 <\ch03\while.php>，您會發現 while 迴圈改寫成 do…while 迴圈後，執行結果是一樣的。

【解答】

\ch03\do.php

```
<!DOCTYPE html>
<html>
  <head>
    <meta charset="utf-8">
  </head>
  <body>
    <?php
      $i = 1;
      do
        echo $i++.'<br>';
      while ($i <= 10);
    ?>
  </body>
</html>
```



3-7 foreach

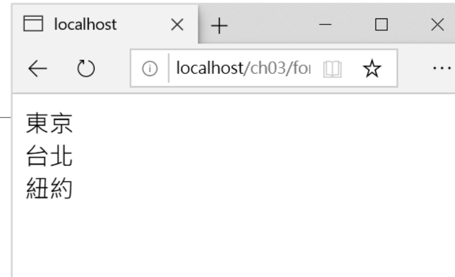
foreach 迴圈是設計給**陣列**（array）使用的 for 迴圈，其語法有下列兩種，*array_name* 為陣列名稱，$*value* 為陣列內目前元素的值，$*key* 為陣列內目前元素的鍵，foreach 迴圈每重複一次，就會移往陣列的下一個元素，若中途要強制離開 foreach 迴圈，可以加上 break 敘述。

```
foreach (array_name as $value)
{
   statements;
   [break;]
   statements;
}
```

```
foreach (array_name as $key => $value)
{
   statements;
   [break;]
   statements;
}
```

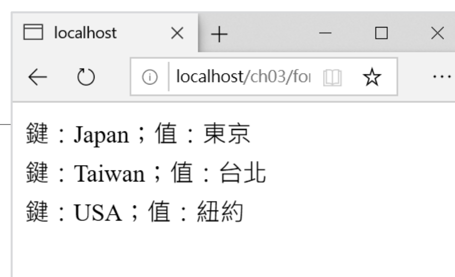
陣列和變數一樣是用來存放資料，不同的是陣列雖然只有一個名稱，卻可以用來存放多個資料。陣列所存放的每個資料叫做**元素**（element），每個元素有各自的**值**（value），至於陣列是如何區分它所存放的元素呢？答案是透過**鍵**（key），在預設的情況下，陣列內第一個元素的鍵為 0，第二個元素的鍵為 1，…，依此類推，第 n 個元素的鍵為 n - 1，第 4 章有進一步的介紹。

以下面的程式碼為例，<\ch03\foreach1.php> 的第 08 行是定義一個名稱為 city 且包含三個元素（'東京'、'台北'、'紐約'）的陣列，而 <\ch03\foreach2.php> 的第 08 行也是定義一個名稱為 city 且包含三個元素（'東京'、'台北'、'紐約'）的陣列，不同的是它還指派了這三個元素的鍵（'Japan'、'Taiwan'、'USA'）。



\ch03\foreach1.php

```
<!DOCTYPE html>
<html>
  <head>
    <meta charset="utf-8">
  </head>
  <body>
    <?php
      $city = array('東京', '台北', '紐約');
      foreach ($city as $value)
        echo $value.'<br>';
    ?>
  </body>
</html>
```



\ch03\foreach2.php

```
<!DOCTYPE html>
<html>
  <head>
    <meta charset="utf-8">
  </head>
  <body>
    <?php
      $city = array('Japan' => '東京', 'Taiwan' => '台北', 'USA' => '紐約');
      foreach ($city as $key => $value)
        echo '鍵：'.$key.'；值：'.$value.'<br>';
    ?>
  </body>
</html>
```

3-8　break 與 continue 敘述

誠如前面所言，break 敘述可以用來強制離開 for 迴圈、while 迴圈、do…while 迴圈、函式等程式碼區塊，此處就不再重複講解。

至於 continue 敘述則可以用來在迴圈內跳過後面的敘述，直接返回迴圈的開頭，例如下面的程式碼只會在網頁上顯示 11 到 15，因為在執行到第 10 行時，只要變數 i 小於等於 10，就會跳過 continue; 後面的敘述，直接返回迴圈的開頭，直到變數 i 大於 10，才會執行第 11 行，在網頁上顯示變數 i 的值。

\ch03\continue.php

```
01:<!DOCTYPE html>
02:<html>
03:   <head>
04:      <meta charset="utf-8">
05:   </head>
06:   <body>
07:      <?php
08:         for ($i = 1; $i <= 15; $i++)
09:         {
10:            if ($i <= 10) continue;
11:            echo $i.'<br>';
12:         }
13:      ?>
14:   </body>
15:</html>
```