

第七章 迴圈

迴圈是一種功能強大的語言結構，它擅長讓電腦能有效的處理重複性的流程或運算式。當電腦能夠快速、準確地重複執行一群語法任務時，程式設計師就不需要一遍遍地去重複的撰寫相關的程式碼。在本章中，您將學到以下的內容：

VLISP 部分

1. while 迴圈
2. repeat 迴圈
3. foreach 函數
4. mapcar 函數

VBA 部分

1. do...loop 迴圈
2. for...next 迴圈
3. while...wend 迴圈
4. for each...next 迴圈
5. exit 函數



7-1 Visual LISP 的迴圈控制函數

所謂「迴圈結構」就是透過反複的「測試條件 — 執行代碼 — 測試條件」方法，來重複的執行一些運算式，直至滿足某個測試條件為止。

在 Visual LISP 中主要的迴圈控制函數有：while、repeat、foreach 與 mapcar 四種。

7-1-1 while 迴圈

while 函數將對測試運算式進行求值，如果不為 nil（吻合條件），則重複執行循環子句中的運算式，直到測試運算式的求值結果為 nil（不吻合條件）為止，

然後跳出迴圈。其標準語法如下：

```
(while (testexpression)
[statement...]
)
```

while 函數將傳回最後 statement 運算式最新的值，如果不需傳回值，可以使用(princ)語法來靜默退出。我們將圖 7-1 來說明迴圈的原理觀念：

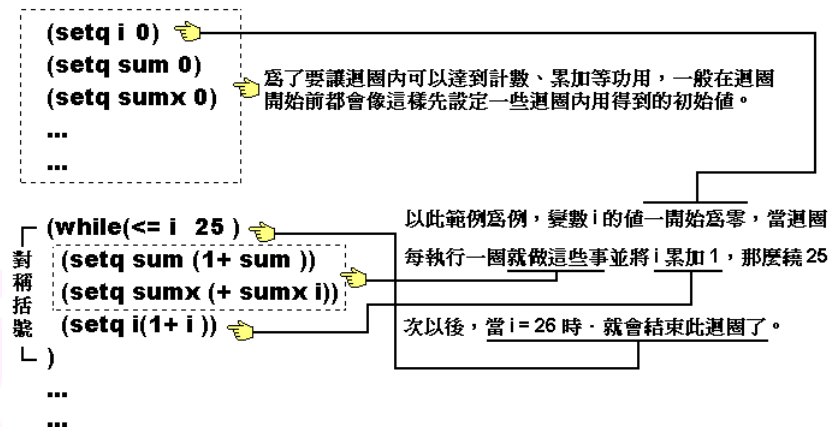


圖 7-1 迴圈的原理觀念

接下來，我們還要以圖 7-2 來說明迴圈的正确設計觀念：

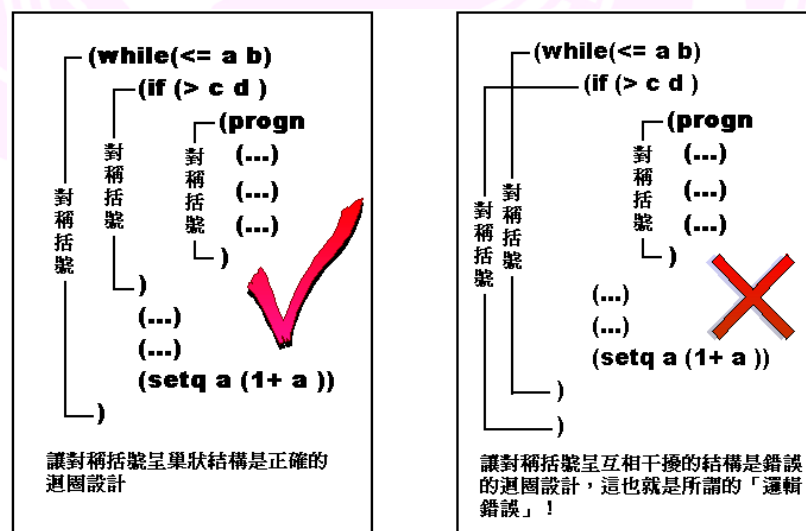


圖 7-2 迴圈的正确設計觀念

下示範例將提示操作者輸入整數的上下限、定出整數的範圍，然後計算整個範圍內的整數和與平均值：

```
(1);;;while loop demo-----whiledemo.lsp
(2);;;function: calculate sum and average of integers
(3)
(4)(defun c:whiledemo(/ fInt sInt temp sum sumx i avr)
(5)  (setq fInt(getint "\n 請輸入第一個整數:"))
(6)  (setq sInt(getint "\n 請輸入第二個整數:"))
(7)  (if (> fInt sInt)
(8)    (progn
(9)      (setq temp fInt)
(10)     (setq fInt sInt)
(11)     (setq sInt temp)
(12)    )
(13)  )
(14)  (setq i fInt)
(15)  (setq sum 0)
(16)  (setq sumx 0)
(17)  (while(<= i sInt)
(18)    (setq sum (1+ sum ))
(19)    (setq sumx (+ sumx i))
(20)    (setq i(1+ i ))
(21)  )
(22)  (setq avr(/ sumx sum))
(23)  (princ "\n 從 ") (princ fInt)
(24)  (princ " 到 ") (princ sInt) (princ " 的整數和 = ") (princ sumx)
(25)  (princ "\n 平均值 = ") (princ avr)
(26)  (princ)
(27))
```

執行結果：請輸入第一個整數:1
請輸入第二個整數:100
從 1 到 100 的整數和 = 5050
平均值 = 50

分析：這個程式的第(7)行～第(12)行是一個判斷子句，如果 fInt 大於 sInt，則交換兩數，並保證 fInt 為較小的整數。第(17)行～第(21)行則為 while 迴圈，它將計算從 fInt 到 sInt 的所有整數和。
在這個迴圈中，我們首先判斷 i (i 的初始值為 fInt) 是否小於 sInt，如果是，那麼計數器變數 sum 就等於 sum + 1，整數和變數 sumx 就等於 sumx + i。然

後再讓 *i* 再加 1（往前進），直到 *i* 等於 *sInt*，結束並跳出迴圈為止。

注意：在 *while* 內的運算式裡，一般都要對運算式裡的變數進行修改，否則很容易造成無限迴圈，導致所謂的「邏輯錯誤」。例如，在上例的迴圈語法中，*(setq i (1+ i))* 的語法作用就是希望每執行一次迴圈，就將 *i* 加 1，如此下去，*i* 一定會等於 *sInt*，這樣就會因為不吻合 *while(<= i sInt)* 而跳出迴圈了。

7-1-2 repeat 迴圈

repeat 函数的主要重點是對循環體中的每一個運算式進行指定次數的求值計算，並傳回最後一個運算式的值。其標準語法如下：

```
(repeat int  
  [expression...]  
)
```

其中，參數 *int* 為正整數，代表迴圈的次數。*repeat* 傳回值將變為最後一個計算的原子或運算式的值。如果不需要傳回值，您可以使用 *(princ)* 語法來靜默退出。例如，以下範例將繪出一個三角函數 *sin* 的曲線圖：

```
(1);;;repeat loop demo-----repeatdemo.lsp  
(2);;;function: draw a sin curve  
(3)  
(4)(defun C:repeatdemo(/ pt pt1 pt2 a1 a i bm cc)  
(5) (setq pt1 (getpoint "\n 請輸入起始點:"))  
(6) (setq pt pt1)  
(7) (setq oo pt1)  
(8) (setq i 0)  
(9) (repeat 36  
(10)   (angl i)  
(11)   (setq pt2 (list (+ (car pt) (/ i 90.0)) (+ (cadr pt) (sin ang))))  
(12)   (command "line" pt1 pt2 "")  
(13)   (setq pt1 pt2)  
(14)   (setq i (+ i 10))  
(15) )  
(16) (command "zoom" "w" (getvar "extmin") (getvar "extmax"))  
(17) (command "pedit" "l" "y" "j" "c")
```

```
(18) (getvar "extmin") (getvar "extmax") "" "s" "")
(19) (command "scale" "l" "" oo 50)
(20) (command "zoom" "E")
(21) (princ)
(22) )
(23)(defun angl (a1)
(24) (setq ang (* pi (/ a1 180.0)))
(25))
```

執行結果：此程式將在 AutoCAD 螢幕上畫出一個填滿螢幕的 Sin 曲線。

分析：程式中的第(9)行～第(15)行將畫出 36 條短線以擬合一條 $0^\circ \sim 360^\circ$ 的 sin 曲線。第(10)行將呼叫 angl 函數來將度度量轉為弧度量。第(16)行程式將用來取得這 36 條線所在的範圍。第(17)行會將這 36 條線段連成一條聚合線，並將之擬合為一條 spline。第(19)、(20)行將調整 Sin 曲線的顯示比例，並令其充滿螢幕。

7-1-3 foreach 函數

foreach 函數可將串列中的所有成員以指定變數的身份帶入運算式中求值。其標準語法如下：

(foreach name list [expression...])

foreach 函數將掃視串列 list，並將其中每一個元素依次賦予變數 name，並對每一個運算式求值。foreach 函數可以指定任意多個運算式。foreach 將傳回最後一個運算式所計算的值，如果未指定運算式，則傳回 nil。如果不需傳回值，您可以使用 (princ) 語法來靜默退出。例如：

```
(foreach n ' (1 2 3) (princ n))
```

其中，n 為變數，運算式(princ n)將順序列印串列(1 2 3)中的各值。其輸出結果為：

1233

最後的那個 3 是 foreach 的傳回值。

7-1-4 mapcar 函數

mapcar 會將一個或多個串列參數裡的各個元素提供給指定函數，以進行求值，並將求值結果變成串列傳回。其標準語法如下：

```
(mapcar function list1... listn)
```

其中，function 是要應用到串列 list1...listn 的函數。實例如下：

```
(setq a 1 b 2 c 3)  
(mapcar '1+ (list a b c))
```

這樣的語法將輸出：(2 3 4)。這樣的二條程式語法就等於以下三條程式：

```
(1+ a)  
(1+ b)  
(1+ c)
```

如果寫出這樣的程式語法：

```
(mapcar '* (list a b c)(list 10 10 10))
```

則將輸出：(10 20 30)。這樣的一條程式語法就等於以下三條程式：

```
(* a 10)  
(* b 10)  
(* c 10)
```

7-2 VBA 的迴圈控制函數

VBA 可支援的常用迴圈控制函數有：

```
do...loop  
for...next  
while...wend
```

```
for each...next  
do...loop
```

以下就為您分節敘述這些迴圈控制函數。

7-2-1 do...loop 迴圈

使用 do 迴圈來重複執行一群運算式，且重複次數不定。do...loop 語句有幾種變體型式，不過每種都是計算數值條件以決定是否繼續執行。例如，If...Then condition 必須是一個數值或者值為 True（非零）或 False（零）的運算式。其標準語法如下：

```
do while condition  
statements...  
loop
```

在 do...loop 迴圈中，只要 condition 為 True，就執行 statements。當 VBA 執行這個 do 迴圈時，首先會測試 condition。如果 condition 為 False（零），就跳過迴圈中所有的運算式。如果 condition 為 True（非零），則 VBA 就會執行迴圈中的運算式，然後退回到 Do While 的開頭，再測試條件，周而復始。因此，只要 condition 為 True 或非零，迴圈就可以隨意執行任意次數。如果 condition 一開始就得到 False 的結果，則不會執行該迴圈。例如，以下程式範例將計算某一目標字串 MyText 在另一字串中出現的次數，同時只要發現目標字串就執行迴圈：

```
Dim position, count  
position = 1  
Do While position>0  
    position = InStr(position+1,MyText, "")  
    count = count + 1  
Loop
```

上示這個程式的另一個意義就是說：如果目標字串未出現在另一個字串中，則 InStr 將傳回 0，而且不再執行迴圈而跳出迴圈。

do...loop 語句的另一種變體型式是：先執行語法，然後在每次執行後才去測試 condition。這種型式將保證 statements 至少被執行過一次。其標準語法如下：

```
do
```



```
statements...
loop while condition
```

還有兩個類似前兩個語法的 do...loop 變體型式，所不同的是：只要 condition 爲 False 而不是 true，它們就執行迴圈。其標準語法如下：

do until condition	Do
statements...	statements...
loop do (迴圈零次或多次)	loop until condition(至少迴圈一次)

7-2-2 for...next 迴圈

在您不確定迴圈內需要重複執行多少次時，宜使用 do...loop 迴圈。但是，如果已知道要執行的次數時，則最好使用 for...next 迴圈。與 do...loop 迴圈不同的是：for...next 迴圈使用一個叫做計數器的變數，每重複一次迴圈之後，計數器變數的值就會增加或者減少。其標準語法如下：

```
for counter = start To end [Step increment]
statements...
next [counter]
```

其中，參數 Counter、Start、end 與 increment 都是數值型的。例如，一樣是 7-1-1 節中 whiledemo.lsp 範例的功能（計算兩個整數之間的所有整數和與平均值），我們以 for...next 迴圈的方式來改寫，程式本文如下：

```
(1)'for next loop demo-----fornext.dvb
(2)'function: calculate the sum and average of integers
(3)
(4)Option Explicit
(5)Public Sub fornext()
(6)Dim fInt As Integer, sInt As Integer
(7)Dim temp As Integer, i As Integer
(8)Dim sum As Integer, sumx As Integer
(9)Dim str As String, str2 As String
(10)fInt = InputBox("Input an integer:")
(11)sInt = InputBox("Input another integer:")
(12)If fInt > sInt Then
```



```

(13)    temp = fInt: fInt = sInt: sInt = temp
(14)End If
(15)sum = 0: sumx = 0
(16)For i = fInt To sInt
(17)    sum = sum + 1
(18)    sumx = sumx + i
(19)Next
(20)str = "ths sum from " & fInt & " to " & sInt & " = " & sumx
(21)str2 = "the average = " & CStr(sumx / sum)
(22)MsgBox str & vbCr & str2
(23)End Sub

```

分析：在程式的第(13)行與第(15)行採用了符號“:”，使原來分別要寫成 3 行和 2 行的程式，一行就解決了。第(16)行～第(19)行就是使用 for...next 迴圈來計算從 fInt 到 sInt 中所有整數和及個數。

7-2-3 while...wend 迴圈

while...wend 迴圈的作用是：while 函數將對條件運算式進行求值，如果不為 nil，則執行迴圈中的運算式。然後重複這個過程，直到測試條件運算式的求結果為 nil 並跳出迴圈為止。其標準語法如下：

```

While condition
[statements]
Wend

```

茲例舉如下：

```

Dim icount As integer
Dim ssetObj as acadselection
While (icontains 0)
    Set ssetObj = thisdrawing.selections.item (icontains -1)
    ssetObj.delete
    icount = icount -1
wend

```

這個範例將檢查選擇組裡是否有元素存在，如果有，就將其刪除。

7-2-4 for each...next 迴圈

for each...next 是 VBA 中一個非常有用的迴圈。這個迴圈語法提供了一種可以遍歷集合物件中所有元素的功能。例如，在 AutoCAD 中，要將一個選擇組 (Selection Set) 中所有物件的顏色變成紅色，就可用下列程式：

```
dim acadObj as Object      '聲明 acadObj 為物件變數
For Each acadObj in ssetObj 'ssetObj 為選擇組
AcadObj.color = acRed
Next acadObj
```

7-2-5 exit 函數

exit 函數特性將允許迴圈在遇到某些特殊值時跳出迴圈。exit 語法有五種型式，它們分別是：

Exit Do	跳出 Do...Loop 迴圈
Exit For	跳出 For...Next 或 For Each...Next
Exit Function	結束含有本語法的函數
Exit Sub	結束含有本語法的程序
Exit Property	結束含有本語法的屬性

下面的範例將接收所輸入的數字，並顯示它的倒數，如果輸入為 0，則跳出：

```
(1)'break loop program-----exitwhile.dvb
(2)'function: break loop
(3)
(4)Option Explicit
(5)Public Sub exitwhile()
(6)Dim x As Double, y As Double
(7)Do
(8)  x = InputBox("Input a real number:")
(9)  If (x = 0) Then
(10)    Exit Do
(11)  End If
```

- (12) $y = 1 / x$
- (13) MsgBox ("The reciprocal = " & y)
- (14) Loop While (1)
- (15) End Sub

執行結果：在 InputBox 中輸入數值，計算它的倒數後在訊息框中顯示出。
如果輸入數為 0，則結束程式。

分析：在本程式第(14)行中的 while 要判斷的運算式值為 1，照理說這個迴圈是不中斷的，會永遠執行下去。所以，我們在第(10)行中使用了 exit do 函數，它位於一個 if 判斷式中。當輸入值為 0 時，就會執行 exit do 語法來強制跳出迴圈。

啓發性習題

一.選擇題(單複選混合)

- 1.() while 迴圈與 repeat 迴圈差別在於：
 - (a) While 將根據判斷式與條件來循環執行，而 repeat 函數的主要重點則是對循環體中的每一個運算式進行指定次數的求值計算，並傳回最後一個運算式的值
 - (b) repeat 將根據判斷式與條件來循環執行，而 while 函數的主要重點則是對循環體中的每一個運算式進行指定次數的求值計算，並傳回最後一個運算式的值
 - (c) 沒有差別
 - (d) 以上皆非
- 2.() 以下有關 foreach 函數的敘述，何者為非？
 - (a) foreach 函數可將串列中的所有成員以指定變數的身份帶入運算式中求值
 - (b) (foreach n ' (1 2 3 4) (princ n)) 的傳回值是 12344
 - (c) 與 repeat 函數的功能類似
 - (d) 以上皆非
- 3.() 以下三條程式：

(- a 10)

(- b 10)

(- c 10)

就相當於：

(a) (foreach '- (list a b c)(list 10 10 10))

(b) (mapcar '- (list a b c)(list 10 10 10))

(c) (while '- (list a b c)(list 10 10 10))

(d) 以上皆非

4.() 當您不確定迴圈內需要重複執行多少次時，宜使用：

(a) for...next 迴圈

(b) while...wend 迴圈

(c) do...loop 迴圈

(d) 以上皆非

5.() 以下哪一個迴圈可以遍歷集合物件中所有元素：

(a) for...next 迴圈

(b) for each...next 迴圈

(c) while...wend 迴圈

(d) 以上皆非

6.() 以下何者為 exit 函數的特性：

(a) 允許迴圈在遇到某些特殊值時跳出迴圈

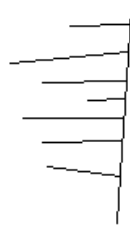
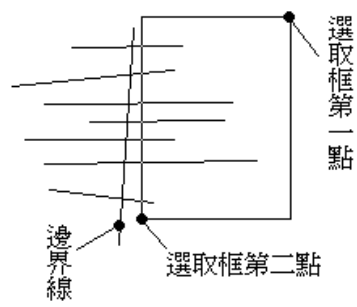
(b) 命令立刻跳出迴圈

(c) 指定再繞三圈就跳出迴圈

(d) 以上皆非

二.實作問答題

1. 請設計一多重剪取功能，讓每次在設定邊界之後就能一次選擇一群圖形來做 Trim 的動作（以 AutoLISP 撰寫）。

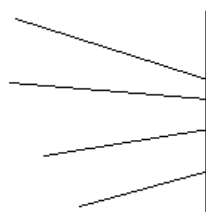
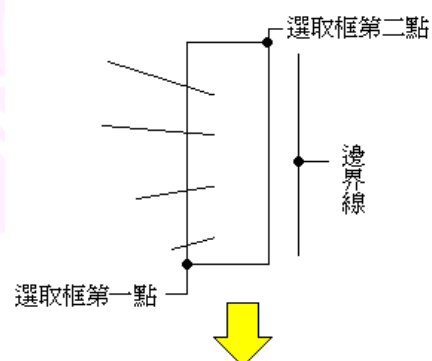


設計詢問句：

- (1) 請點取邊界線！
- (2) 請輸入選取框的第一點：
- (3) 輸入選取框的第二點：

解答檔案名稱：MTRIM.LSP

2. 如上面那題，但設計一多重延伸功能（以 AutoLISP/VLISP/VBA 撰寫）。

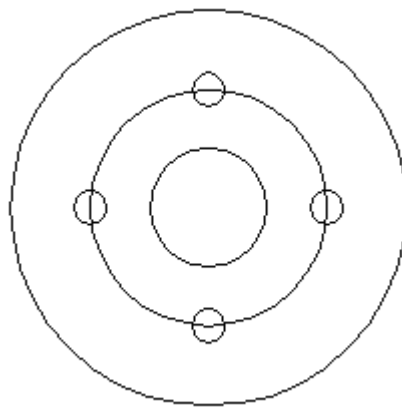


設計詢問句：

- (1) 請點取邊界線！
- (2) 請輸入選取框的第一點：
- (3) 輸入選取框的第二點：

解答檔案名稱：MEXTEND.LSP，V-MEXTEND.LSP，MEXTEND.DVB

3. Flange 圖形，在機械設計的繪製中，是很常見的設計元件。我們希望您能設計一個可依操作者輸入的設計條件來自動繪出 Flange 圖形的程式（以 AutoLISP/VLISP/VBA 撰寫）。

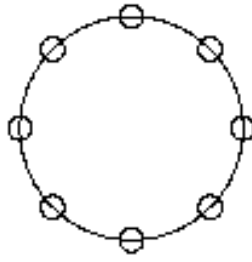


設計詢問句：

- (1) 請輸入 Flange 的中心點:
- (2) 請輸入 Flange 的外圓半徑:
- (3) 請輸入 Flange 的內圓半徑:
- (4) 通過圓孔中心的中心線:
- (5) 圓孔直徑:
- (6) 圓孔數:
- (7) 第一個圓孔的起始角度:

解答檔案名稱：FLANGE.LSP，V-FLANGE.LSP，FLANGE.DVB

4. 在 AutoCAD 中，「圖形鎖定」的其中之一功能 —Quadrant（鎖定圓或弧的四分之一點）雖然不錯，但是，卻沒有鎖定圓或弧的八分之一點的功能，您能設計一個嗎？（以 AutoLISP/VLISP 撰寫）。



設計詢問句：例如：您想拉一條線到一圓的八分點上，載入本程式後，請依下述指令流程執行之：

指令: Line <Enter>

起點: (請在螢幕上選取任一點)

下一點: (P8) <Enter> 或

下一點: (V-P8) <Enter> 或

的 (請點取螢幕上已經事先繪好的一圓上)

下一點: <Enter>

解答檔案名稱：P8.LSP，V-P8.LSP