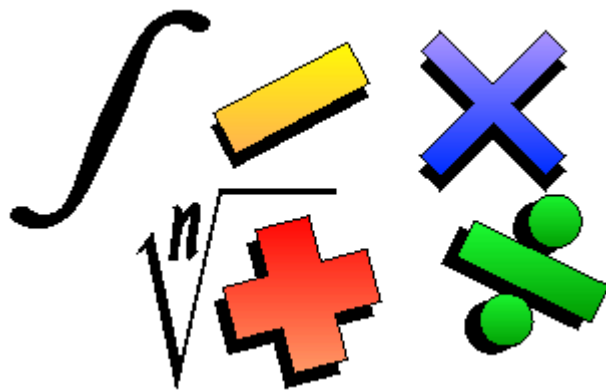


第四章 高階語言的運算子與運算式

對資料的處理就需要用到運算式，您可以說：程式就是由運算式所構成的，而運算式則是由運算元（即參數值）與運算子所組成的。所謂的「運算子」就是一些特殊的符號，它將利用一群運算元的值來計算產生新值。由於語言特質的關係，各種語言所規定的運算符和運算式往往不同。在本章中，您將學到下述主題：

- 運算子
- 運算式



4-1 Visual LISP 的運算子

Visual LISP 的運算子將包括：

- 算術運算子：

+ - * / 1+ 1-

- 比較運算子：

= /= < <= > >=

- 邏輯運算子：

~ And Or

表 4-1 將詳細列出了這群 Visual LISP 的運算子資訊：

表 4-1 Visual LISP 的運算子

| 運算子 | 作用 | 資料類型 |
|-----|------------------------------------|-----------|
| + | 傳回所有參數的和 | 數值型 |
| - | 將第一個參數減去其他參數的和並傳回差值 | 數值型 |
| * | 傳回所有參數的乘積 | 數值型 |
| / | 將第一個參數除以其他數的乘積並傳回商 | 數值型 |
| = | 比較參數是否相等，相等則傳回 T，否則傳回 nil | 數值型或字串型 |
| /= | 比較參數是否不相等，不相等則傳回 T，否則傳回 nil | 數值型或字串型 |
| < | 如果每個參數值都小於它右邊的參數，則傳回 T，否則傳回 nil | 數值型或字串型 |
| <= | 如果每個參數值都小於或等於它右邊的參數，則傳回 T，否則傳回 nil | 數值型或字串型 |
| > | 如果每個參數值都大於它右邊的參數，則傳回 T，否則傳回 nil | 數值型或字串型 |
| >= | 如果每個參數值都大於或等於它右邊的參數，則傳回 T，否則傳回 nil | 數值型或字串型 |
| ~ | 傳回參數的 NOT（即 1 的補數）運算 | 整數型 |
| And | 傳回參數的 And 邏輯運算 | 字串型、T、nil |
| Or | 傳回參數的 Or 邏輯運算 | T、nil |
| 1+ | 參數加 1 | 數值型 |
| 1- | 參數減 1 | 數值型 |

4-2 Visual LISP 運算式字首表示法

在 Visual LISP 中，運算式採用字首表示法，即將運算子放在運算元之前，並將運算子（即函數名稱）和運算元（即呼叫函數的參數）用括弧括起來，來作為一個運算式，這樣的運算式是串列的一種。例如：(+ 25 23 24)，結果將傳回 75。

4-3 Visual LISP 賦值運算式

如果要將 100.0 的值賦予變數 x，那麼一般我們會使用函數 setq 來構成運算式。如：

```
(setq x 100)
```

再看一個例子，要履行 $x=(a+b)/c$ 這樣的等式，Visual LISP 的運算式語法為：

```
(setq x (/ (+ a b) c))
```

編譯器將依運算元的類型來執行運算式計算，如果兩個運算元都是整數，那麼編譯器將產生整數的計算結果；若運算子中有一個以上的實數，那麼編譯器也將自動產生實數的計算結果。如下範例所示：

```
_$ (/ 5 3)
1
_$ (/ 5 3.0)
1.66667
```

最後，我們就舉一個簡單範例來讓您順利學習 Visual LISP 運算子與運算式的混合應用：

```
(1)'Visual LISP Operator Program-----oper.lsp
(2)'function:demo Visual LISP operator
(3)
(4)(defun c:oper ()
(5)  (setq int1 (GETINT "\nInput first integer:"))
(6)  (setq int2 (GETINT "\nInput second integer:"))
(7)  (setq real1 (GETREAL "\nInput first real:")
(8)    real2 (GETREAL "\nInput second read:"))
(9)  )
(10) (setq str1 (GETSTRING "\nInput first string:")
(11)   str2 (GETSTRING "\nInput second string:"))
(12) )
(13)
(14) (princ "\nInteger operation\n")
(15) (princ int1)(princ " + ")(princ int2)(princ " = ")
```

(16) (princ (+ int1 int2))(princ "\n")
(17) (princ int1)(princ " - ")(princ int2)(princ " = ")
(18) (princ (- int1 int2))(princ "\n")
(19) (princ int1)(princ " * ")(princ int2)(princ " = ")
(20) (princ (* int1 int2))(princ "\n")
(21) (princ int1)(princ " / ")(princ int2)(princ " = ")
(22) (princ (/ int1 int2))(princ "\n")
(23) (princ int1)(princ " rem ")(princ int2)(princ " = ")
(24) (princ (rem int1 int2))(princ "\n")
(25) (princ "\n")
(26)
(27) (princ "\nReal operation\n")
(28) (princ real1)(princ " + ")(princ real2)(princ " = ")
(29) (princ (+ real1 real2))(princ "\n")
(30) (princ real1)(princ " - ")(princ real2)(princ " = ")
(31) (princ (- real1 real2))(princ "\n")
(32) (princ real1)(princ " * ")(princ real2)(princ " = ")
(33) (princ (* real1 real2))(princ "\n")
(34) (princ real1)(princ " / ")(princ real2)(princ " = ")
(35) (princ (/ real1 real2))(princ "\n")
(36) (princ "\n")
(37)
(38) (princ "\nIncreasement and decrease operation\n")
(39) (princ int1)(princ " + ")(princ 1)(princ " = ")
(40) (princ (1+ int1))(princ "\n")
(41) (princ int1)(princ " - ")(princ 1)(princ " = ")
(42) (princ (1- int1))(princ "\n")
(43) (princ "\n")
(44)
(45) (princ "\nLogical operation\n")
(46) (princ str1)(princ " < ")(princ str2)
(47) (princ " : ")(princ (< str1 str2))(princ "\n")
(48) (princ int1)(princ " <= ")(princ int2)
(49) (princ " : ")(princ (<= int1 int2))(princ "\n")
(50) (princ real1)(princ " > ")(princ real2)
(51) (princ " : ")(princ (> real1 real2))(princ "\n")
(52) (princ str1)(princ " >= ")(princ str2)
(53) (princ " : ")(princ (>= str1 str2))(princ "\n")

```
(54) (princ int1)(princ " = ")(princ int2)
(55) (princ " : ")(princ (= int1 int2))(princ "\n")
(56) (princ real1)(princ " /= ")(princ real2)
(57) (princ " : ")(princ (/= real1 real2))(princ "\n")
(58) (princ "\n")
(59)
(60) (princ "\nBit operation\n")
(61) (princ "~")(princ int1)(princ "=")
(62) (princ (~ int1))(princ "\n")
(63) (princ "~")(princ int2)(princ "=")
(64) (princ (~ int2))(princ "\n")
(65) (princ "\n")
(66)
(67) (princ)
(68))
```

執行方式：若提示輸入：

```
Input first integer:5
Input second integer:10
Input first real:3.25
Input second read:1.8
Input first string:book
Input second string:Book
```

則將輸出：

```
Integer operation
5 + 10 = 15
5 - 10 = -5
5 * 10 = 50
5 / 10 = 0
5 rem 10= 5
```

```
Real operation
3.25 + 1.8 = 5.05
3.25 - 1.8 = 1.45
3.25 * 1.8 = 5.85
```

$3.25 / 1.8 = 1.80556$

Increase and decrease operation

$5 + 1 = 6$

$5 - 1 = 4$

Logical operation

book < Book : nil

$5 \leq 10 : T$

$3.25 > 1.8 : T$

book >= Book : T

$5 = 10 : \text{nil}$

$3.25 \neq 1.8 : T$

Bit operation

$\sim 5 = -6$

$\sim 10 = -11$

分析：整數運算的結果是整數，實數運算的結果仍是實數。

4-4 VBA 的運算子與運算式

在 VBA 中的運算子將包括：

● 算術運算子

\wedge $*$ $/$ \backslash Mod $+$ $-$

● 比較運算子

$<$ \leq $>$ \geq $=$ $<>$

● 連接運算子

$\&$

● 邏輯運算子

And Eqv Imp Not Or Xor

表 4-2 我們將詳細列出了這群 VBA 的運算子資訊：

表 4-2 VBA 的運算子

| 運算子 | 作用 | 資料類型 |
|-----|--|------------|
| ^ | 求數字的次方值 | 數值型 |
| * | 求乘積 | 數值型 |
| / | 求商的實數值 | 數值型 |
| \ | 求商的整數值 | 數值型 |
| Mod | 求餘 | 數值型 |
| + | 求和 | 數值型 |
| - | 求差 | 數值型 |
| < | 比較運算式是否小於另一運算式，若小於則傳回 True，否則傳回 False | True、False |
| <= | 比較運算式是否小於或等於另一運算式，若小於或等於則傳回 True，否則傳回 False | True、False |
| > | 比較運算式是否大於另一運算式，若大於則傳回 True，否則傳回 False | True、False |
| >= | 比較運算式是否大於或等於另一運算式，若大於或等於則傳回 True，否則傳回 False | True、False |
| = | 比較運算式是否等於另一運算式，若等於則傳回 True，否則傳回 False | True、False |
| <> | 比較運算式是否不等於另一運算式，若不等於則傳回 True，否則傳回 False | True、False |
| & | 以字串來串連接兩個運算式 | 字串型 |
| And | 傳回兩個運算式的 And 邏輯運算 | True、False |
| Eqv | 判斷兩個運算式是否邏輯等值，如果等值則傳回 True，否則傳回 False | True、False |
| Imp | 傳回兩個運算式進行邏輯運算，若第 1 個運算式為 True，則檢查第 2 個運算式，若為 True，傳回 False，否則均傳回 False | True、False |
| Not | 傳回兩個運算式的 Not 邏輯運算 | True、False |
| Or | 傳回兩個運算式的 Or 邏輯運算 | True、False |
| Xor | 傳回兩個運算式的 Xor 邏輯運算 | True、False |

4-5 VBA 的運算式範例

本節，我們就以下面這個範例來說明 VBA 運算子與運算式的混合應用：

```
(1)'VBA Operator Program-----oper.vba
(2)'function:demo VBA operator
(3)
(4)Public Sub Oper()
(5)
(6)Dim int1 As Integer, int2 As Integer
(7)Dim dbl1 As Double, dbl2 As Double
(8)Dim str1 As String, str2 As String, str As String
(9)Dim bln1 As Boolean, bln2 As Boolean
(10)
(11)int1 = 5: int2 = 2
(12)dbl1 = 3.2: dbl2 = 6.4
(13)str1 = "acad": str2 = "Acad"
(14)str = str1 & str2
(15)bln1 = True: bln2 = False
(16)
(17)Debug.Print int1 & "^" & int2 & "=" & (int1 ^ int2)
(18)Debug.Print int1 & "*" & int2 & "=" & (int1 * int2)
(19)Debug.Print dbl1 & "/" & dbl2 & "=" & (dbl1 / dbl2)
(20)Debug.Print dbl1 & "\" & dbl2 & "=" & (dbl1 \ dbl2)
(21)Debug.Print int1 & " Mod " & int2 & "=" & (int1 Mod int2)
(22)Debug.Print int1 & "+" & int2 & "=" & (int1 + int2)
(23)Debug.Print int1 & "-" & int2 & "=" & (int1 - int2)
(24)Debug.Print int1 & "<" & int2 & "=" & (int1 < int2)
(25)Debug.Print dbl1 & "<=" & dbl2 & "=" & (dbl1 <= dbl2)
(26)Debug.Print str1 & ">" & str2 & "=" & (str1 > str2)
(27)Debug.Print int1 & ">=" & CDbl(int1) & "=" & (int1 >= CDbl(int1))
(28)Debug.Print int2 & "=" & CDbl(int2) & "=" & (int2 = CDbl(int2))
(29)Debug.Print CInt(dbl1) & "<>" & dbl1 & "=" & (CInt(dbl1) <> dbl1)
(30)Debug.Print str1 & "&" & str2 & "=" & str
(31)Debug.Print bln1 & " And " & bln2 & "=" & (bln1 And bln2)
(32)Debug.Print bln1 & " Eqv " & bln2 & "=" & (bln1 Eqv bln2)
```


(33)Debug.Print bln1 & " Imp " & bln1; " = " & (bln1 Imp bln1)

(34)Debug.Print "Not " & bln2; " = " & (Not bln2)

(35)Debug.Print bln1 & " Or " & bln2; " = " & (bln1 Or bln2)

(36)Debug.Print bln1 & " Xor " & bln2; " = " & (bln1 Xor bln2)

執行結果：此程式將於視窗中列印：

$5 \wedge 2 = 25$

$5 * 2 = 10$

$6.4 / 2 = 3.2$

$6.4 \setminus 2 = 3$

$5 \bmod 2 = 1$

$5 + 2 = 7$

$5 - 2 = 3$

$5 < 2 = \text{False}$

$3.2 \leq 6.4 = 20.48$

$\text{acad} > \text{Acad} = \text{True}$

$5 \geq 5 = \text{True}$

$2 = 2 = \text{True}$

$6 \lt \gt 6.4 = \text{True}$

$\text{acad} \& \text{Acad} = \text{acadAcad}$

$\text{True And False} = \text{False}$

$\text{True Eqv False} = \text{False}$

$\text{True Imp True} = \text{True}$

$\text{Not False} = \text{True}$

$\text{True Or False} = \text{True}$

$\text{True Xor False} = \text{True}$

分析：略。在前一個範例的基礎上，這個範例應很容易理解。

啓發性習題

一.選擇題(單複選混合)

1.() 何謂「運算子」？

- (a) 就是一些特殊的數字，它可利用一群運算元的值來計算產生新值。
(b) 就是一些特殊的文字，它可利用一群運算元的值來計算產生新值。
(c) 就是一些特殊的符號，它可利用一群運算元的值來計算產生新值。
(d) 以上皆非

2.() 以下何者屬算術運算子？

- (a) *
(b) &
(c) (
(d) /

3.() 以下何者是 $A=(123*25/45)+3$ 的正確 AutoLISP/VLISP 運算式寫法？

- (a) (setq A (+ (/ (* 123 25) 45) 3))
(b) A=(123*25/45)+3
(c) (setq A=123*25/45+3)
(d) 以上皆非

4.() 以下何者是 VBA 的邏輯運算子

- (a) And
(b) Eqv
(c) Imp
(d) Not

二.實作問答題

1. 請試比較 Visual LISP 與 VBA 運算子的異同。
2. 請依回歸法 (Recursive) 繪製一具中國古典美的英文字母 "C"。這是一道數學題，用意在請您練習代入公式（以 AutoLISP/VLISP 撰寫）。

$$\text{字長增量} = \frac{\text{字長}}{\sqrt{2}}$$

參考公式：

$$\begin{aligned} \text{角度減增量} = & \left(\text{角度} + \frac{\pi}{4} \right) \\ & \left(\text{角度} - \frac{\pi}{4} \right) \end{aligned}$$

ok

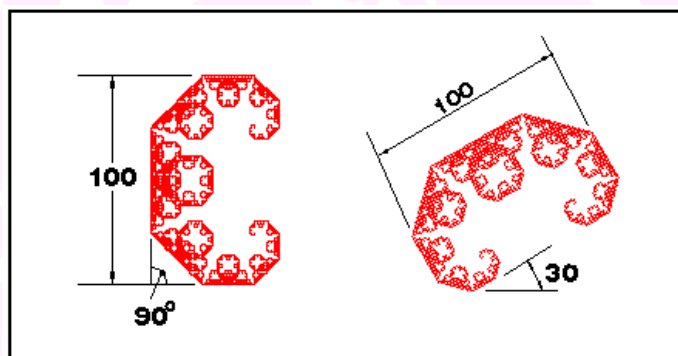
設計詢問句：

- (1) 請輸入字的長度 = 100 <Enter>
- (2) 請輸入字的角速度 = 90 <Enter>
- (3) 請指定字的長度增量 = 2 <Enter>
- (4) 請指定字的位置:

解答檔案名稱：RECUC-1.LSP，RECUC-2.LSP，V-RECUC-1.LSP，
V-RECUC-2.LSP

註：RECUC-2.LSP 與 RECUC-1.LSP 兩檔案的差別在於前者是以 LINE 指令來繪製，所以速度較慢，但屬實際圖形的繪製；而後者是以 grdraw 函數來繪製，所以速度較快。後者是以虛擬顯像來表示圖形，因此當您再執行另一指令時，此虛擬顯像圖形即消失。這樣的功能在您要做「指導性」的功能時，非常有用。這也是一非常值的您思考應用的習題範例。

完成圖例：



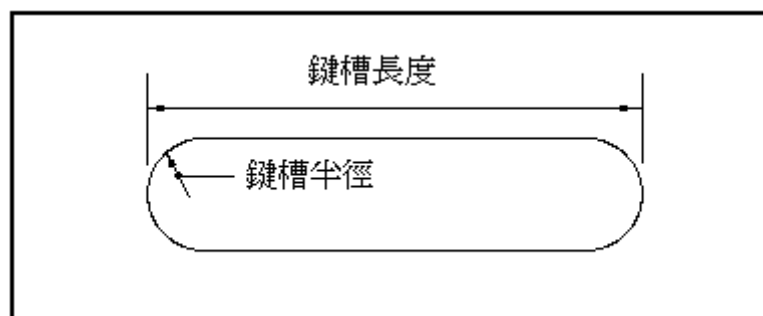
3. 請設計一機械上常用的鍵槽 (Keyway) 圖形 (以 AutoLISP/VLISP/VBA 撰寫)。

設計詢問句：

- (1) 請點取鍵槽插入點:
- (2) 請輸入鍵槽半徑:
- (3) 請輸入鍵槽長度:
- (4) 請輸入鍵槽圖與水平所呈的角度:

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

完成圖例：



註：對機械專業的讀友而言，若此鍵槽圖形還缺少什麼，請自行補上。

