

TP 312
N 151
LISP 基礎

LISPcraft by Robert Wilensky

南 石 譯

儒林圖書公司 印行

序

1. LISP 程式語言

LISP 是目前還在使用的少數古老的程式語言之一，約翰·麥卡錫（John McCarthy）和他的學生在一九五〇年代末期首度發表了這種語言。

LISP 這個名字，是取 LISt Processing（串列處理）的頭幾個英文字母組成的，以表達隱含於大部份 LISP 程式和 LISP 資料之下的基本結構：串列（List）。LISP 語言過去及現在的主要用途仍然在於人工智慧領域。

在這個計算領域快速變遷的時代，LISP 是少數能夠幸存下來的古老程式語言之一。從它首度發表及實作成功至今，LISP 語言經歷了無數次的演化，增加了許多新的性能，其實作方法也有了長足的進步，和使用使用者之間的介面也做了明顯的改善。這種語言已經建立起屬於自己的程式設計環境。此外，LISP 更藉著硬體的協助，以具有高度工作效率的 LISP 機器的形式出現。

這些改良通常都是為了改善早期版本在觀念上或實作上所呈現的缺點而進行的。到了今天，LISP 某些令人難以接受的部份可說已不存在。但是，LISP 過去殘留在人們腦海中根深蒂固的印象常使得人們仍然認為它是種有趣但却不切實際的語言，不過由於近年來電腦科技的快速進展以及日趨完善的軟體發展環境，使得 LISP 的這種形象已大為改觀。許多 LISP 團體，已

如雨後春筍般大量的出現。

要探討 LISP，首先面臨的困難就是無法針對 LISP 語言究竟是什麼做個明確的定義。和別種語言不同的是，標準的 LISP 語言並不存在。LISP 語言擁有許多不同的發展方向，這導致 LISP 有個龐大的家族。LISP 語言的這些不同版本雖然具有相同的基本性質，但在某些方面又完全不同。有種 LISP 版本，稱為 Common LISP，嘗試著替 LISP 程式設定更具統一性的標準，但是目前仍無一種 LISP 的解譯方式得到所有 LISP 團體的支持而公認為是一種標準。

目前最常用的 LISP 版本是 MacLISP，它是在 MIT 發展出來的，還有就是 INTERLISP，由 Bolt, Beranek and Newman and Xerox Palo Alto 研究中心所發展。這些版本已廣泛應用於特殊用途的 LISP 機器上。Portable Standard LISP 是另一種 LISP 版本，其可輕易的在不同的機器上做轉換。至於 Scheme LISP 版本則由 MIT 的 Sussman 和 Abelson 所推出，其對於許多基本概念的解譯方式和時下大多數的 LISP 語言版本截然不同。

2. 範圍及目標

本書所用的 LISP 語言是 Franz LISP。Franz 是種外觀上近似於 MacLISP 的 LISP 版本，由加州大學柏克萊分校所發展，可在柏克萊 UNIX* 作業系統上使用。當然，本書儘量超然於各種不同版本之外，為達這個目標，除了在各種版本的差異點上指出其它版本的不同用法外，並加上超然於版本特色（意即，較一般性）的註解。此處所引用的哲學基礎是：與其從頭學習一種新的版本，不如透過其它的版本比較學習來進行。

本書編排的目標在於希望讀者能對 Franz LISP 的工作方式具有較完整的認識，因此不對語言本身做任何的形式定義，而完全採取實用主義的觀點。除了對 LISP 的基本特性做個說明以外，程式設計的慣常用法、各種程式設計風格的基本元素，以及系統的介面等論題也佔有同等的份量。根據作者

* UNIX 是貝爾實驗室的註冊商標。

個人的經驗，這些論題對於程式設計語言的影響不下於這語言的基本性能。

探討程式設計風格時已先進行過濾工作，以便提供某些有用的程式語言設計風格給讀者參考。對於好的和壞的程式設計風格也都加上完善的註解。這些註解完全是根據於對 LISP 程式設計實務所做的觀察，而非根源於程式風格上的偏好。

LISP 在人工智慧領域內的應用範圍很廣，但是這不是本書所要探討的重點，此處著重的是語言本身，而不是它的應用。等到熟悉這種語言的基本特性之後，讀者可以查閱其它的教科書，以進一步了解何謂人工智慧。

雖然本書的內容具有自我擴充的特質，但是仍然無法完全描述 Franz LISP 的每一部份。讀者可以查閱 J.K. Foderaro 和 K.L. Sklower 所著的 The Franz LISP Manual 一書，以瞭解所有的細節。那本手冊 (Manual) 著重於說明 Franz LISP 能做什麼，而非 LISP 解譯程式是什麼。

3. 格 式

用電腦語言來設計程式就和從事其它的工作一般，需要特殊的技巧：只擁有專業的知識而無實務經驗也是枉然。即使是最好的程式設計語言教科書，也無法讓讀者熟練的運用這種語言，除非讀者實際動手去使用這種語言。有了這種認識，本書的格式即設計成 LISP 解譯程式的交談式章節。若想善用本書，讀者需根據各節所列的對話實際動手執行 LISP，依照對話的內容輸入運算式，然後注意 LISP 系統會產生什麼樣的反應。

本書與其說是本教科書，不如視為旅遊指南來得恰當。若不實際動手試試，就無法真正的學到東西。讀者首先須熟悉書中前面部份的章節所描述的語言基本性能，這可讓讀者擁有足夠的知識，以充分應付未來章節中學習上所會遭遇的任何狀況。

最後，作者鼓勵讀者盡量嘗試各種不同的用法。並非讀者所碰到的每件事，在一開始時就能看出它的意義和價值。雖然要了解一個國家的很好方式是依循大眾化的路線而行，但是也不要忽略了“雖小道，亦有可觀者焉”這

句話。這也是事物總能引發人們的興趣和好奇心的主要原因。

祝各位一帆風順。

LISP 基礎

易懂、實用和高吸引力是這本書的特色——想要了解 LISP 的學生和想要深入發掘 LISP 語言特色的程式設計師，都應該閱讀這本書。本書的作者，威廉斯基教授，從最基本的觀念談起，把在 LISP 解譯程式上執行過的活生生例子呈現在讀者面前，鼓勵讀者動手去做，然後介紹了幾近於全部的 LISP 特性，例如 LISP 內部結構、罕用的資料型態和函數、偵錯問題、錯誤處理問題以及系統函數等等。這本書最精華的部份在於語言的工作環境之探討，包括常用的程式設計慣用法、繁雜的 LISP 結構和無數的程式設計風格，這些都以使用者的觀點做為討論的基礎；所有的“技巧和手腕”都得自於 LISP 程式設計經驗的累積。

威廉斯基教授把重心放在 Franz LISP 上。這個 LISP 版本可在柏克萊 UNIX 上使用。書中凡是和其它版本的 LISP 不同之處，都會附上詳細的說明。

這本書完全以語言的介紹為主，對於想研究語言架構的人而言，這是本最佳的教科書。書中最後兩章也分別針對樣式配對以及聯結式資料庫管理發展了兩個應用程式。

羅伯特·威廉斯基是加州柏克萊大學的計算機科學副教授。他在耶魯大學拿到博士學位，並參加了耶魯大學的人工智慧計畫。他的研究領域著重於常識理解和自然語言處理這兩方面。

儒林圖書公司 印行

目 錄

序	XV
1. LISP 程式語言	XV
2. 範圍及目標	XVI
3. 格 式	XVII
 第 1 章 基本概念	 1
1.1 引 言	1
1.2 LISP 解譯程式	1
1.3 求 值	3
1.3.1 對更複雜的 s-運算式進行求值	3
1.4 函數的引數	6
1.5 基元和繫結	7
1.6 離開 LISP	11
1.7 摘 要	12
習 題	13

第 2 章 符號計算 15

2.1	引 言.....	15
2.2	串列的更深入探討.....	15
2.3	把事物當成文字看待.....	17
2.4	car 和 cdr	21
2.5	cadr 等	25
2.6	空串列.....	27
2.7	cons	28
2.8	串列建構函數.....	31
2.9	摘 要.....	32
習 題		33

第 3 章 定義自己的函數 35

3.1	引 言.....	35
3.2	使用者定義的函數.....	35
3.3	自由變數.....	41
3.4	超級括弧.....	44
3.5	將函數儲存於檔案內.....	44
3.5.1	自動初始化.....	45
3.6	摘 要.....	46
習 題		46

第 4 章 述句、條件式與邏輯運算子 49

4.1	引 言.....	49
4.2	LISP 的述句.....	49
4.3	條件式.....	54

4.3.1	更複雜的 cond	55
4.3.2	cond 的一般形式	58
4.4	邏輯運算子	61
4.4.1	使用 and 和 or 控制程式的流程	62
4.5	摘 要	63
習 題		64

第 5 章 遞歸計算 65

5.1	引言——遞歸計算與反覆計算之比較	65
5.2	遞歸的基本概念	66
5.3	遞歸計算與參數繫結	68
5.4	遞歸計算常犯的錯誤	69
5.5	更複雜的遞歸計算	70
5.6	摘 要	77
習 題		77

第 6 章 LISP 中的反覆計算 79

6.1	引 言	79
6.2	prog	79
6.2.1	一般的 prog 結構	81
6.3	反覆計算與遞歸計算之比較	83
6.4	其它的反覆形式	84
6.4.1	更結構化的反覆計算	85
6.5	摘 要	86
習 題		87

第 7 章 性質串列 89

7.1	引 言.....	89
7.2	基本概論.....	89
7.3	實 例	91
7.4	性質串列的重要性.....	93
7.5	性質串列以及基元的唯一性.....	93
7.6	分離式性質串列.....	94
7.7	摘 要.....	95
	習 題.....	96

第 8 章 以函數作為引數的函數：eval、apply 和對映函數 99

8.1	引 言.....	99
8.2	函數作為引數使用.....	99
8.3	apply	101
8.4	eval	105
8.4.1	eval 的背景問題.....	107
8.5	對映函數.....	108
8.5.1	各式各樣的對映函數.....	110
8.5.2	apply-append 技巧	112
8.6	摘 要.....	113
	習 題.....	114

第 9 章 lambda 119

9.1	引 言.....	119
9.2	不具名稱的函數.....	119

9.3	def、putd 和 LISP 的內部構造	124
9.4	一些參考範例	125
9.5	function 函數	127
9.6	lambda 用於繫結變數之法	128
9.7	lambda 的重要性	129
9.8	摘 要	130
習 題		131

第 10 章 資料的讀取和寫出

10.1	引 言	133
10.2	read 和 print	133
10.3	使用特異的名稱讀取和列印基元	137
10.4	字 串	141
10.5	I/O 的重定方向	142
10.6	美觀列印	144
10.7	其它的輸入／輸出	145
10.8	LISP 中的 LISP	145
10.8.1	evalquote 頂層	147
10.8.2	read-eval-print 迴圈	148
10.9	摘 要	149
習 題		149

第 11 章 偵 錯

11.1	LISP 中的偵錯問題	151
11.2	中斷的 read-eval-print 迴圈	152
11.3	檢查堆疊的內容	153
11.4	偵錯程式	155

11.5	追蹤及中斷	162
11.6	摘要	165
習題		166

第 12 章 其它類型的函數 169

12.1	引言	169
12.2	expr 與 fexpr	170
12.2.1	nlambda	172
12.2.2	fexpr 的背景效應	173
12.3	函數的規律	175
12.4	lexpr	175
12.4.1	定義 lexpr 的更簡便方法	177
12.4.2	lexpr 形式	178
12.4.3	其它的 lexpr 函數: LISTIFY 和 SETARG	179
12.5	閉鎖	179
12.5.1	封閉一組函數	183
12.6	已編譯以及外來的函數	184
12.7	擴充的 lambda 形式	186
12.8	摘要	186
習題		187

第 13 章 巨集 191

13.1	引言	191
13.2	以巨集設計可讀性更高的程式碼	191
13.3	巨集的定義法	193
13.4	巨集的展開	195
13.5	巨集範例	196

13.6	設計巨集的技巧	198
13.6.1	defmacro	199
13.7	let 巨集	201
13.8	setf	202
13.9	優雅的使用巨集	203
13.10	其 它	204
13.11	巨集的代換	205
13.12	摘 要	206
習 題		207

第 14 章 讀取巨集 209

14.1	引 言	209
14.2	在 read 時執行函數	209
14.3	讀取巨集的定義法	211
14.4	接合巨集	212
14.5	反引號巨集	214
14.5.1	在巨集定義內使用反引號	217
14.6	摘 要	218
習 題		219

第 15 章 點對與其它內部結構 221

15.1	引 言	221
15.2	串列的內部表示法	221
15.3	點對記號法	223
15.4	指向同一個物件的多重指標	225
15.4.1	eq	227
15.5	rplaca 和 rplacd 的危險性	228

15.6	摘 要	233
習 題		234
第 16 章 錯誤處理以及非標準的控制流程		239
16.1	以程式處理錯誤的狀況	239
16.2	errset	240
16.3	用 err 模擬錯誤	241
16.4	catch 與 throw	241
16.4.1	更具彈性的 catch 與 throw	243
16.5	errset、err 與 catch、throw 的比較	244
16.6	自動載入	245
16.7	摘 要	245
習 題		246
第 17 章 LISP 符號表		247
17.1	再談 read	247
17.2	登錄基元的名稱	248
17.3	基元的表示法	249
17.4	oblist	249
17.5	處理 oblist 的函數	251
17.5.1	製造新基元	251
17.5.2	自動產生名稱	252
17.5.3	從物件串列中移去基元	255
17.6	應用——模組內的區域變數	256
17.7	摘 要	258
習 題		259

第 18 章 其它的資料型態	261
18.1 引 言	261
18.2 FRANZ 所能提供的資料型態	262
18.3 字 串	264
18.4 陣 列	266
18.4.1 各式各樣的陣列存取架構	267
18.5 小型向量	268
18.6 向量與直接向量	269
18.6.1 以向量做為使用者定義的資料型態	270
18.6.2 直接向量	272
18.7 value	272
18.8 二進位	274
18.9 摘 要	275
習 題	276
第 19 章 系統函數	277
19.1 引 言	277
19.2 作業系統的函數	277
19.2.1 LISP 的繁衍	277
19.2.2 建立一個 shell 子處理過程	279
19.2.3 執行 UNIX 命令	279
19.2.4 覆蓋現行的 LISP 處理過程	282
19.2.5 終止現行的 LISP 處理過程	283
19.2.6 儲存現行的 LISP 處理過程	283
19.2.7 改變現行目錄	283
19.2.8 UNIX 環境的相關資訊之取得	284
19.2.9 捕捉訊號	284
19.3 LISP 內部的系統函數	285

19.3.1	判斷基元是否已繫結某個值	286
19.3.2	明顯地呼叫 LISP 頂層	286
19.3.3	廢物收集	286
19.3.4	計時	286
19.3.5	取得與 LISP 相關的線上資訊	286
19.3.6	內部的選擇項	287
19.4	摘要	287

第 20 章 編譯 289

20.1	LISP 編譯簡介	289
20.2	LISP 編譯程式的用法	290
20.2.1	liszt	292
20.3	編譯程式中的宣告	292
20.3.1	特殊的變數	293
20.3.2	其它的宣告	295
20.3.3	區域性的宣告	296
20.4	編譯程式的其它特殊形式	297
20.4.1	eval-when	297
20.4.2	引入其它檔案	298
20.4.3	特殊的 progn	298
20.4.4	函數定義	298
20.4.5	其它的形式	299
20.5	編譯程式特有的程式設計顧慮	299
20.5.1	lexpr 的問題	299
20.5.2	文字的問題	299
20.5.3	編譯算術函數	300
20.5.4	只供編譯程式使用的巨集	300
20.5.5	巨集的自動載入	300

20.6	轉移表·····	300
20.7	編譯程式所提供的選擇項·····	301
20.8	初始化·····	302
20.9	摘 要·····	302
習 題	·····	303

第 21 章 LISP的應用：樣式配對 ····· 305

21.1	引 言·····	305
21.2	樣式配對·····	305
21.3	有關近似性的說明·····	306
21.4	樣式配對變數的繫結·····	308
21.5	統一性·····	311
21.6	配對程式·····	312
21.7	摘 要·····	317
習 題	·····	318

第 22 章 LISP的應用：聯結式資料庫管理系統 ····· 319

22.1	引 言·····	319
22.2	聯結式資料庫的實作·····	319
22.2.1	某些實作上的考慮事項·····	320
22.2.2	管理程式·····	321
22.3	建立索引·····	326
22.4	推理式擷取程式·····	330
22.4.1	向後鏈結·····	330
22.4.2	推理式擷取程式的實作·····	331
22.5	摘 要·····	333