電脳技術叢書

BASIC釋譯程式

工業技術研究院 電子工業研究所 編印

序言

電腦技術發展中心電腦系統部所出版的技術叢書系列，目的在將本中心內研製計劃有關之技術資料公諸社會，其一方面可供各級學校利用，另一方面也可供工商研究機關參考。電腦技術發展中心叢書之出版，盼望能給目前電腦工業發展帶來直接助益。

本書爲叢書之一，目的在介紹本所自行發展的ITRI BASIC語言的釋譯系統。

本書在編寫期間，承蒙凌淑明小姐協助整理，並在技術上諮詢交通大學計算機工程系杜敏文教授，獲益良多，在此一併致謝。

王 有 禮

識於電子工業研究所電腦技術發展中心電腦系統部

中華民國七十年二月二十五日

目 錄

第一章　緒論

在電腦剛問世的時候，是以機器語言（Machine Language亦稱之機器碼machine code ）來編寫（ Coding ）程式。而機器語言對人來說，非但不易學習，編寫繁雜，而且維護（ Maintenance ）困難，所以有組合語言（ Assembly ）的誕生。組合語言是以助憶符號（ Memonic ）來代表機器碼，而且組譯程式（ Assembler ）將組合程式轉換成機器碼。由於學習組合語言，對電腦系統要有相當程度的瞭解，所以學習起來還是很吃力，於是有高階語言（ High Level Language 如FORTRAN，COBOL，PASCAL 等等）的出現，而組合語言之類則稱之爲低階語言（ Low Level Language ），高階語言經由編譯程式（ Compiler ）可以轉換成低階語言。

所以以某種語言編寫好的原始程式（ Source Program，亦稱之原始碼 Source code ）轉換成機器碼的過程，謂之語言處理程式（ Language Processor ）圖 1 － 1 ，爲將原始碼轉換成機器碼的一種語言處理程式，即將原始碼輸進編譯程式得到目標碼（ Object Code ），再將目標碼輸進連結程式（ Linker ）可以得到機器碼，其中虛線的路經表示將原始程式直接編譯（ Compile ）成機器碼，亦稱之 Compile and Go.

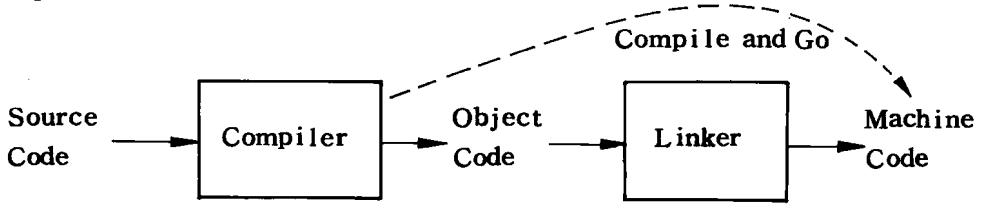

圖 1 － 1　Language Processor

另外，再介紹在語言處理程式中常見到四個名詞；組譯程式（ Assembler ）、編譯程式（ Compiler ）、轉換程式（ Translator ）、釋譯程式（ Interpreter ）圖 1 － 2 中列出這四個名詞。

此圖中可以看出組譯程式的輸入是低階語言，輸出是目標碼或是機器碼，編譯程式的輸入是高階語言，而輸出是低階語言、目標碼，或機器碼。組譯程式與編譯

程式的差別即在於其輸入的不同。轉換程式爲將一個語言所寫的程式轉換成另外一個語言的程式，如輸入FORTRAN程式轉換成COBOL ，所以組譯程式及編譯程式皆可稱爲一種轉換程式。釋譯程式爲輸入一個程式並直接求出該程式的結果。與轉換程式的差別在輸出的不同。

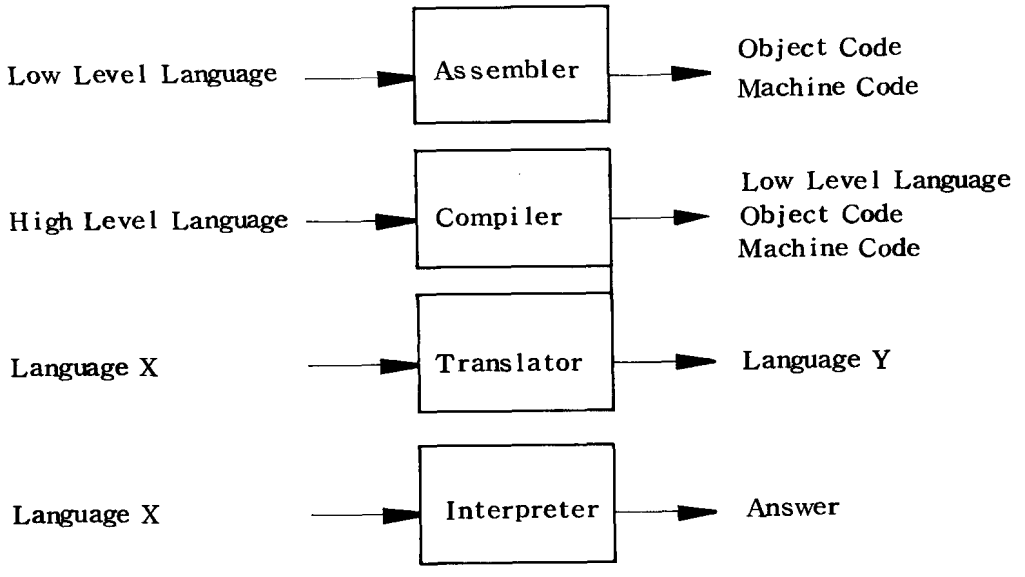

圖 1 － 2　Assembler. Compiler. Translator. Interpreter

圖 1 － 3 中表示出編譯程式及釋譯程式在內部處理上的區別，圖左表示編譯程式的大致流程（ flow ），圖右表示釋譯程式的大致流程。編譯程式在輸入程式後經字彙分析（ Lexical Analysis ）、語法分析（ Syntax Analysis ）、語意分析（ Semantic ）、碼的最佳化（ Optimization ）、建碼（ Code Generation ）而後產生目標碼；而釋譯程式在輸入程式後經過字彙分析、語法分析，直接釋譯（ Interpret ）而得到程式的結果。

圖 1 － 4 以實際的例子解釋編譯程式與釋譯程式的區別，圖中編譯程式輸入英文（原始程式）編譯成中文（目標碼），而蘋果完好如初。釋譯程式在輸入英文（原始程式）後，便直接釋譯（執行）此英文，所以蘋果在輸入時還是好好的，而在輸出時已經被吃光了。

本書第二章介紹釋譯程式之系統結構（ System Structure ），將分成系統之資料結構（ Data Structure ）及程式結構（ Program Structure ）兩部份，三、四章爲資料結構的延續，五、六章爲程式結構的詳細內容。第三章介紹使用者區域（

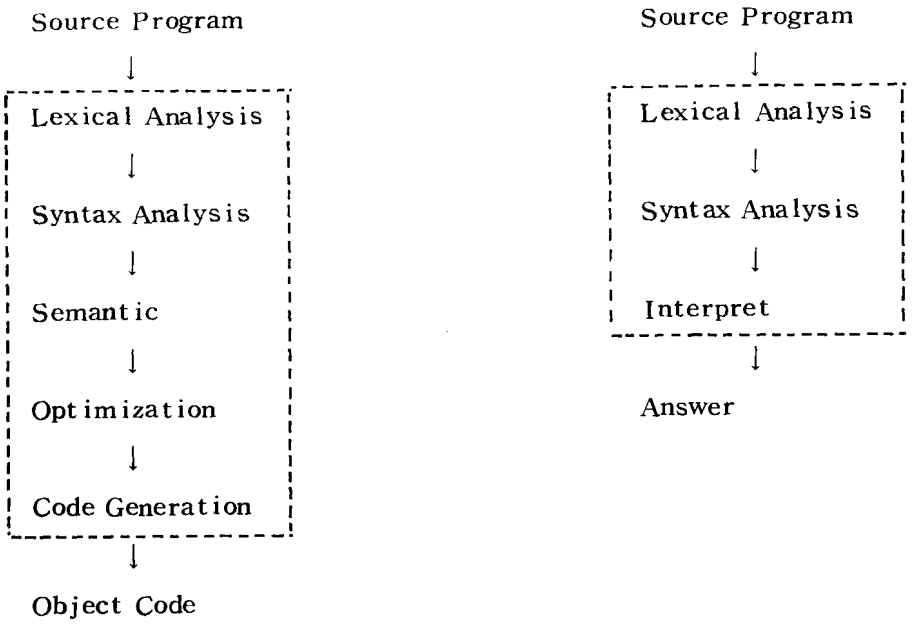

圖 1 － 3　Difference between compiler and interpreter

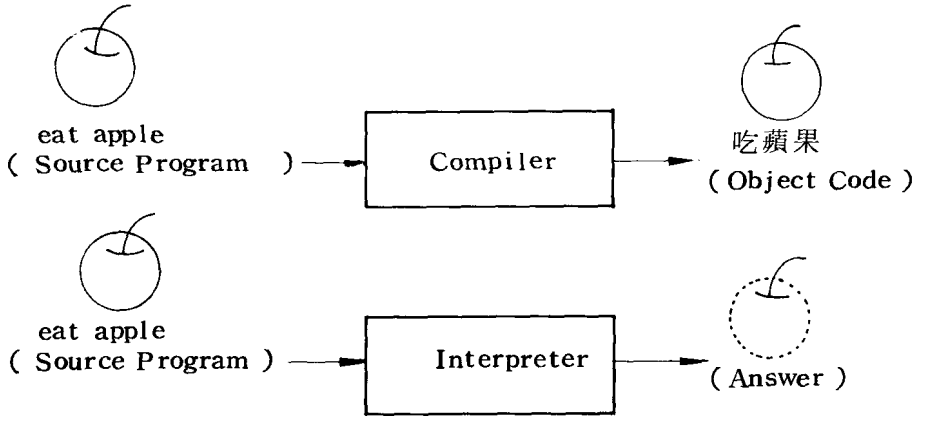

圖 1 － 4　Example of interpreter

User Region ）之結構，說明使用者程式在記憶體（ Memory ）中的存放方法及儲存格式。第四章介紹釋譯程式與使用者區域之關係，即使用者區域的資料如何透過各種表（ Table ）而做運算，此章應爲本書之精髓所在。第五章介紹釋譯程式的寫法，將逐一說明各個程式的內容以利讀者瞭解及系統維護（ System Maintenance ）。第六章介紹釋譯程式之重疊（ Overlay ），如何將佔60 k 位元組（本系統之大小）的釋譯程式利用重疊之技巧而變成32 k 位元組。第七章將做一簡單的結論。

第二章　系統結構

本章介紹系統之資料結構及程式結構，分述如下：

第一節　系統之資料結構

BASIC釋譯程式系統之全貌如圖２－１，分成三部份，分別爲操作系統（Operating System）、釋譯程式、使用者區域。圖中使用者區域內的虛線表示動態安置（Dynamic Allocation），箭頭方向表示安置方向，使用者區域的大小於編寫釋譯程式時決定。使用者區域又可分爲程式儲存區（User Program Area 簡稱UPA）、變數值區（Run Time Value Area 簡稱RTVA）、常數區（Literal Constant Area 簡稱LCA）及未使用區（Unuse Area 簡稱UA）。

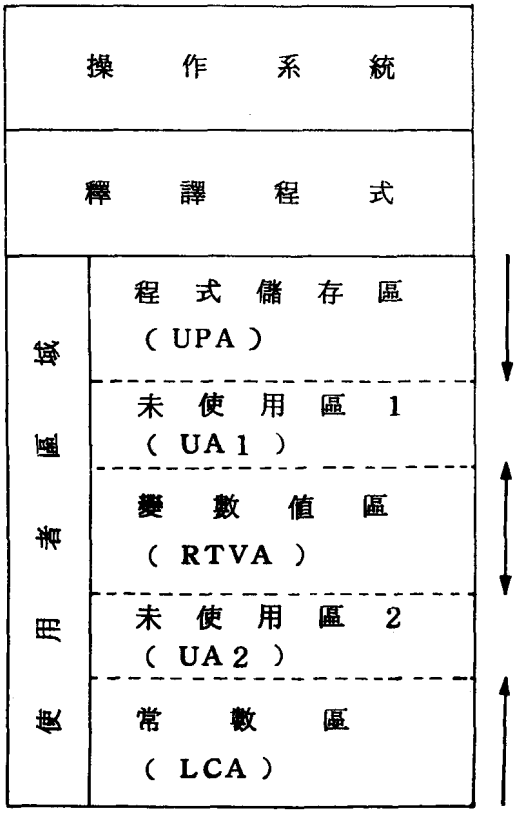

圖２－１

釋譯程式區域內含各種程式及表，其詳細內容將於第四、五章中說明。使用者區域的各區內容概述如下：

一、使用者區域之內容

使用者區域劃分成三部份的原因爲減少廢棄空間（ Garbage Space ）的產生，如將變數值區與常數區合而爲一則經過CLEAR命令陳述後會造成許多廢棄空間。

程式儲存區爲使用者鍵入（ Key in ）之陳述，經字彙分析、語法分析後產生之標準符號（ Uniform Symbol ），亦即句元（ Token ），儲存於此區內，亦即本系統之BASIC語言的中間碼（ Intermediate Code ）儲存區。

未使用區 1，此區除當作程式儲存區的生長空間外，並於執行時充當算式（ Expression ）的工作區（ Working Storage ），其詳細過程見第五章第六節運算優先次序程式模組。

變數值區爲執行時安置識別號（ Identifier ）值的區域。

未使用區 2，此區除當作變數值區及常數區的成長空間外，尚可提供問題函數（ Problem Function ）暫存實際參數（ Real Argument ）之用，其詳過程見第五章第六節。

常數區爲儲存原始程式中常數的地方。

二、使用者區域之動態安置

使用者區域之動態安置的發生情形有三種：

1. 程式儲存區成長而撞及變數值區。
2. 變數值區成長而撞及常數區。
3. 常數值區成長而撞及變數值區。

情形 1 將變數值區往下移，情形 2 ，3 將變數值區往上移，如記憶體所餘空間（ UA 1 + UA 2 ）不及所需空間大時，則發生記憶體超溢（ Memory Overflow ）。

使用者區域之動態安置的流程圖見圖 2 － 2 ，圖中“ L ”爲所需安置空間之長度，UA 1 爲未使用區 1 ，UA 2 爲未使用區 2 。

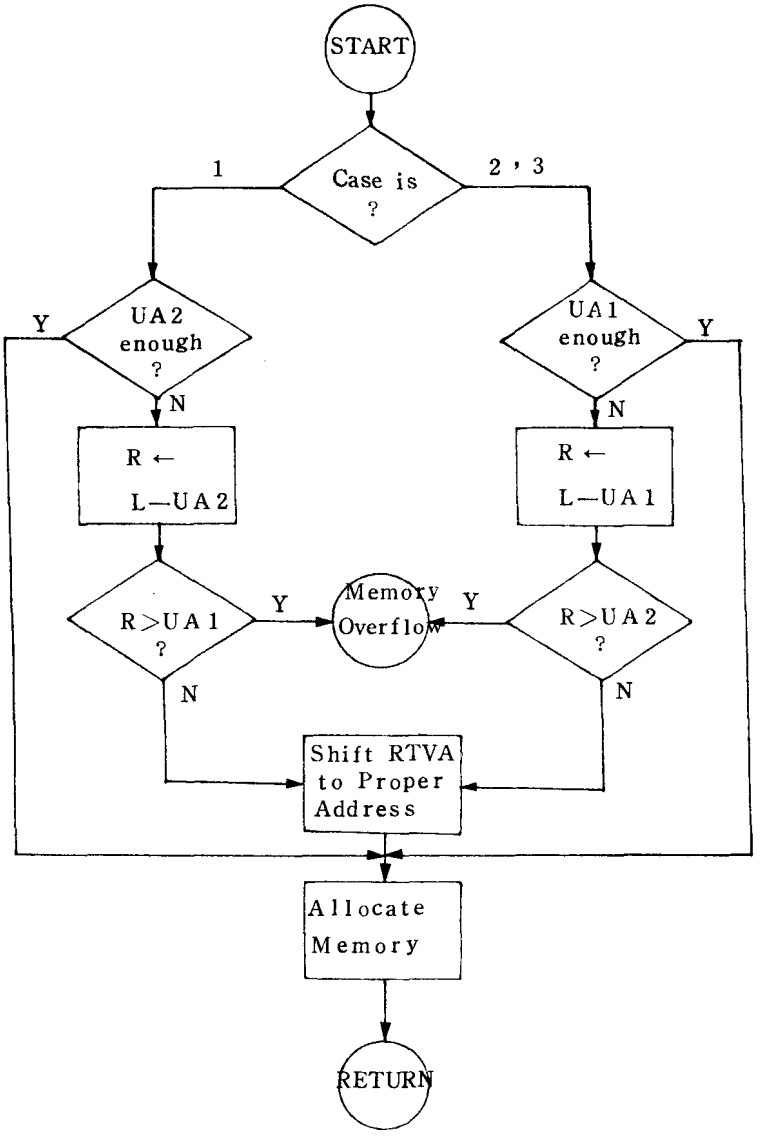

圖 2 － 2

第二節　程式結構

本系統之程式分成九個模組（ Module ），其名稱及用途分述如下：

1 根程式（ ROOT）—此模組之主要工作爲設定初值（ Initial Value ） 及印標頭（ Heading ）。

2. 控制程式（ CONTROL ）— 此模組在區分及呼叫（ Call ）BASIC 語言所接受的三種模態（ Mode ），即命令模態（ Command Mode ）、整批模態，（ Batch Mode ）和立即模態（ Immediate Mode ）。
3. 掃瞄程式（ SCANNER ）— 此模組做字彙分析的工作，找出句元。
4. 語法程式（ PARSER ）— 此模組做語法分析的工作，檢查陳述的結構（ Syntax ）對錯與否。
5. 運算優先次序程式（ PRECEDENCE ）— 此模組計算算式之值。
6. 模擬程式（ SIMULATOR ）— 此模組在模擬 BASIC 語言的各個陳述。
7. 命令程式（ COMMAND ）— 此模組執行 BASIC 語言的命令陳述。
8. 錯誤訊息程式（ ERROR ）— 此模組在印出使用者程式所產生的各種錯誤。
9. 共用程式（ LIBRARY ）— 此模組包含上述 8 個模組所需共用的程式。

上述九個模組間的相互關係如圖 2 － 3 。

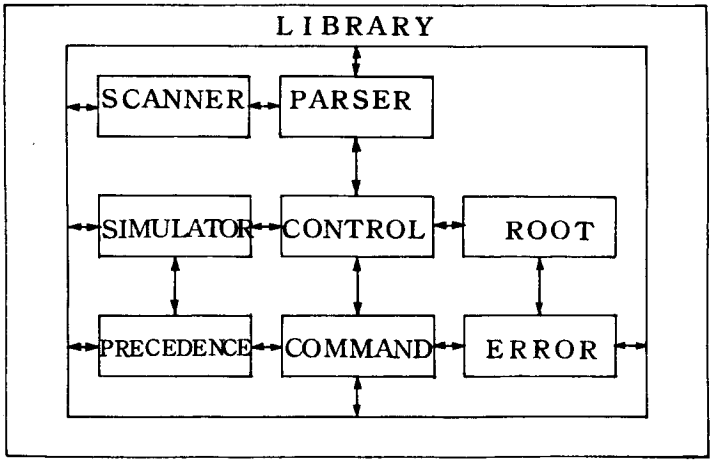

圖 2 － 3

圖 2 － 4 爲釋譯程式的大致流程，根程式設定初值後呼叫控制程式，控制程式讀入一行使用者鍵入的陳述，判別爲那一種模態，如爲命令模態陳述則呼叫命令模態處理器（ Processor ），如爲整批模態則呼叫整批模態處理器，如爲立即模態則呼叫立即模態處理器。

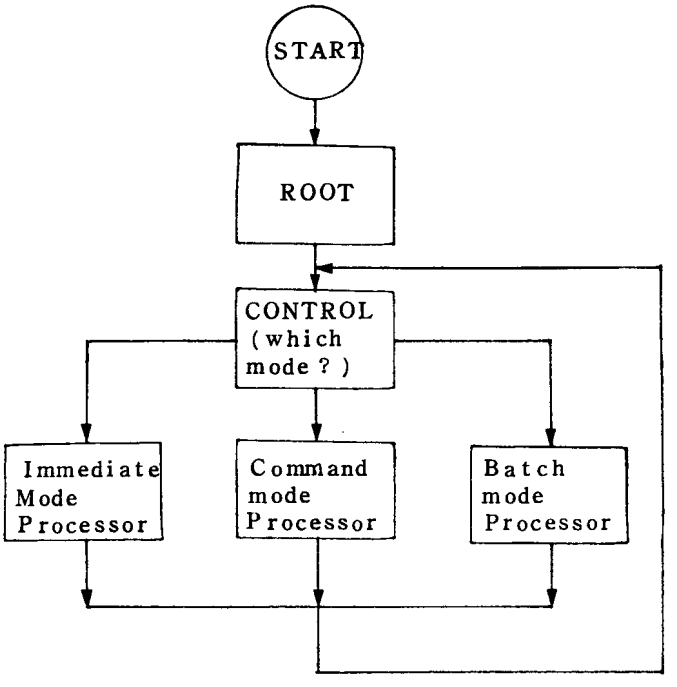

圖 2 － 4

BASIC 語言的三種模態爲整批模態，命令模態和立卽模態，其定義如下：

< batch mode > → < line number >< statement list >

< command mode > → < command statement >

< immediate mode > → < statement list >

< statement list > → < statement > \< statement list >
| < statement > | < null >

< statement > → < call statement > | < chain statement >
| < close statement > | ……
| < stop statement >

< command statement > → < append statement > | < clear
statement > | …… |
< unsave statement >

整批模態，爲由行號（ Line Number ）開始的陳述，經釋譯程式處理後，將該陳述的資訊（ Information ）儲存於程式儲存區及常數區。

立卽模態爲無行號的陳述，經釋譯程式處理後可得到該陳述所要的結果，整批模態要得到陳述的結果尙須經過 RUN 命令。立卽模態的陳述執行完後不存在程式儲存區中，整批模態的陳述則一直儲存在程式儲存區中。

命令模態的陳述有執行程式命令 RUN ，印出程式命令 LIST ，……等等，命令模態與立卽模態不同，立卽模態僅執行本身的陳述，命令模態的陳述具有與使用者交談的功能。

第三章　使用者區域之結構

使用者區域由程式儲存區，變數值區，常數區及未使用區組成，今將各區的資料結構詳述如下：

第一節　程式儲存區資料結構

本區的資料結構可概括分爲兩類，一爲原始碼型（ Source Code ），一爲標準符號型（ Uniform Symbol ）。所謂原始碼型卽將使用者所鍵入的陳述照本宣科一字不變的錄進程式儲存區，依此類方式儲存的陳述有兩個卽 DATA 陳述與REM陳述。所謂標準符號型卽使用者鍵入的陳述經字彙分析後所得的結果爲本系統所設定的標準符號，再將此標準符號存入程式儲存區。不論原始碼型或標準符號型，存入程式儲存區的每一陳述皆以“ 0 ”爲其結束符號。由於原始碼型較單純，不再贅述，僅介紹標準符號型。

標準符號分爲類別（ Class ）和對應值（ Value ），其基本格式如下：

VALUE	CLASS
(1 byte)	(1 byte)

本系統之類別及對應值各佔 1 位元組（ byte ），而某些類別的格式因情況而稍有變異，於後面介紹各種類別時仔細說明。

本系統將BASIC語言的句元（ Token）規劃成八種類別，分別爲陳述，命令，特殊符號（ Special Character ），算術函數（ Arithmetic Function ），字串函數（ String Function ），問題函數，識別號及常數。圖 3 － 1 中爲各類別及其代碼。

CODE	CLASS
0	STATEMENT
1	COMMAND
2	SPECIAL CHARACTER
3	ARITHMETIC FUNCTION
4	STRING FUNCTION
5	PROBLEM FUNCTION
6	IDENTIFIER
7	LITERAL CONSTANT

圖 3 — 1

CLASS=0

VALUE	STATEMENT
01	ASFILE
02	CALL
03	CHAIN
04	CLOSE
05	COMMON
06	DATA
07	DEF
08	DIM
09	END
10	FOR
11	GOSUB
12	GOTO
13	IF
14	INPUT
15	KILL
16	LET
17	LINE
18	LINPUT
19	NAME
20	NEXT
21	ON
22	OPEN
23	OUTPUT
24	OVERLAY
25	PRINT
26	RANDOMIZE
27	READ
28	REM
29	RESET
30	RESTORE
31	RETURN
32	STEP
33	STOP
34	THEN
35	TO
36	USING
37	ASSIGNMENT

圖 3 — 2

一陳述之標準符號

本系統之BASIC陳述共有 30 種，除指定陳述（ Assignment ）不是由關鍵字（ Keyword ）開頭外，其餘每一陳述皆由關鍵字開頭，故有 29 個開頭的關鍵字，而陳述中間的關鍵字（ Intermediate Keyword ）共有 7 個，並付予指定陳述一虛擬關鍵字（ Dummy Keyword ），所以陳述類別的對應值共有 37 個。

由於“ 0 ”爲程式儲存區的特殊符號，而陳述的類別爲“ 0 ”，如對應值仍爲“ 0 ”，則與程式儲存區陳述的結束符號“ 0 ”發生混淆，故陳述的對應值定爲從“ 1 ”開始。圖 3 － 2 爲陳述類別各陳述的對應值。

二命令之標準符號

BASIC語言的命令共有 17 種，其中LIST及RUN 兩個命令可以控制是否要印標頭，如LISTNH（ NH爲No Heading 的意思 ）意即不要印標頭，而 RUN 命令的文法（ Grammar ）格式爲

< run command > → RUN | RUNNH | RUN < filename >
| RUNNH < filename >

如將LISTNH與RUNNH列入命令的標準符號中，易與檔案名稱（ Filename ）混淆，故有關NH的字彙分析單獨處理，而不將LISTNH與RUNNH列入命令之標準符號中。

命令標準符號的對應值“ 0 ”及“ 14 ”，爲備用的對應值，無特殊意義。命令之標準符號見圖 3 － 3 。

CLASS=1

VALUE	COMMAND
01	APPEND
02	CLEAR
03	COMPILE
04	DEL
05	LENGTH
06	LIST
07	BYE
08	NEW
09	OLD
10	RENAME
11	REPLACE
12	RESEQ
13	RUN
14	NULL
15	SAVE
16	SCR
17	SUB
18	UNSAVE

圖 3 － 3

三特殊符號的標準符號

圖3－4中列出BASIC語言的特殊符號共有 19 種，其中對應值“2”，“6”各爲單運算元（Unary）的加和減，而加和減的對應值分別爲“1”和“3”，當字彙分析得到“＋”和“－”的對應值“1”和“3”，經判別後如爲單元的“＋”，“－”，只須將雙運算元（Binary）的值向左移（shift）一位即可。特殊符號“＝”有兩個意思，一爲指定運算元（Assignment operator）一爲關係運算元（Relational operator）。“＜＝”爲小於等於，“＜＞”爲不等於，“＞＝”爲大於等於，“＼”爲多陳述（Multi-statement）的隔離符號。

CLASS=2

VALUE	SPECIAL CHAR.
00	#
01	+
02	UNARY +
03	-
04	*
05	/
06	UNARY -
07	^
08	(
09	)
10	,
11	=
12	＼
13	<
14	>
15	<=
16	<>
17	>=
18	;

圖3－4

四算術函數的標準符號

算術函數與字串函數在本系統中必無必要分成兩種類別，但是爲了和語言參考手册（Language Reference Manual）配合，故劃分成兩種類別。本系統函數之虛擬參數取代法爲傳送值法（Call by Value）。算術函數的虛擬參數爲一數字算式，不可爲字串算式，且只可有一個虛擬參數，但是算術函數 PI 不可有虛擬參數，RND 可以有一個虛擬參數，也可以沒有虛擬參數。圖3－5爲算術函數的標準符號。