计算机科学资料〔XB〕之二十二

符号处理语言FOY文本

DJS－21机FOY语言程序上机说明

1977年12月修订

中国科学院数学研究所计算站

符号处理语言FOY文本

一、目的和功能

FOY是“XR”（系列软件）的一个成员。可用来写数据结构复杂和要求高精度数值计算的程序。也可用实现公式处理语言GOY的工作具。主要特点如下：

㈠是高级语言。有通常高级语言的控制结构和运算表达式。变量类型动态可变，由赋值决定。函数可以递归定义和递归调用。数包括整数和分数。数的运算不受机器字长的限制。行的连接运算也不受机器字长的限制。

㈡有一个动态存贮分配系统，可在目标程序运行中随时按需分配成片的新存贮。当存贮用光时，又能自动收集当时已经无用的“废”存贮，联结成片，继续供用，程序自动继续执行。在此基础上，提供能够动态变化的表和矢。

表是符号处理的必要工具。每个表有两个元素，分别称为头和尾。表X的头和尾分别用HD(X)和TL(X)表示。他们可以是数、行、逻辑值、函数名，也可以是表或矢。表的产生请参看第七节。

矢与表不同的地方是它可以有多个元素，这些元素都用下标选取。矢X的第N个元素用X〔N〕表示。矢的元素可以是数、行、逻辑值、函数名，也可以是表或矢。如果矢X的第N个元素也是矢，则X〔N,M〕等价于X〔N〕〔M〕，余类推。矢的产生请参看第七节。

表和矢与ALGOL60的数组比较起来，最根本的不同是，数组的元素不能是数组，而表和矢的元素却可以是表和矢，从而可以构成非常复杂的结构，用以解决各种各样的符号处理问题。

所谓符号处理问题，包括形式微分、不定积分、公式演算、谓词演算、定理证明、博奕、情报检索、机器翻译、图象识别、文稿编排、机器人，以及其他人工智能问题。

二、基本符号、标识符、行、数、程序的当前值

FCY程序是用下列基本符号构造起来的：

<基本符号>::=<字母> <数字> <其他>

<字母>::= A|B|C|D|E|F|G|H|I|J|K|L|M|N|O|P|Q|R|S|T|U|V|W|X|Y|Z

<数字>::= 0|1|2|3|4|5|6|7|8|9

<其他>::= (|)|〔|〕|+|-|*|/|=|,|;|.|:|'|␣|#|回车|换行

␣表示空格，在行外无意义，但可用来分隔标识符和无符号整数。#就是小10。

<标识符>::=<字母>$\left\{<字母>|<数字>\right\}_0^n$

标识符没有固定意义，通常用作一个名字。但以下标识符保留作本语言的定义符：

<保留标识符>::= BEGIN|END|RETURN|FOR|DO|

UNTIL | WHILE | BREAK | GO | CALL | BACK | SWITCH | ON | CASE | ELSE | AD | HD | TL | TRUE | FALSE | NIL

<行>::='<基本符号的任意序列，其中引号加倍书写>'

在行内如果有引号，则每两个相邻的引号代表一个引号。

<无符号整数>::= $\{<数字>\}_1^n$

无符号整数参加运算后，结果可以是整数和分数：

<数>::=<整数> | <分数>

<整数>::=<表达式>

<分数>::=<表达式>

FCY属于所谓表达式语言。在程序执行中每计算了一个表达式，就得到一个值，叫做程序的当前值。但程序的当前值只在计算表达式时改变，在其他情形保持不变。

三、程序、语句

㈠FCY程序由一列语句组成：

<程序>::=<语句> | <程序><；| 回车换行><语句>

语句之间用分号或回车换行分隔。但在语句或表达式之前均可随意插入回车换行，参看后头。

㈡语句可带若干标号，还可带局部标号、局部子程序入口、子程序入口或函数体入口：

<语句>::= {=<局部子程序入口>::| <子程序入口>::|

<函数体入口><:|回车换行>$\}_0^1$ {=<局部标号>:$\}_0^1$ {<标号>:$\}_0^n$<基本语句>|回车换行<语句>

<局部子程序入口>::=<数字>|<字母>

<子程序入口>::=<标识符>

局部子程序入口和子程序入口都规定一个非递归子程序的起点和名字。子程序入口是全程的。

<函数体入口>::=$\{*\}_1^n$<函数名>(${<形参表>}_0^1$) {(<局部量表>)}$_0^1$

<函数名>::=<标识符>

<形参表>::=<标识符表>

<局部量表>::=<标识符表>

<标识符表>::=<标识符>|<标识符表>,<标识符>

函数体入口规定一个函数名及其形参、其局部量、其函数体的入口地址。函数名都是全程的。

<局部标号>::=<数字>|<字母>

<标号>::=BEGIN|<标识符>

局部标号和标号都规定程序的一个位置，可以由转向语句转向它。标号是全程的。程序从特殊标号 BEGIN 开始执行。

<基本语句>::=<函数体返回语句>|<函数体结束语句>|<表达式语句>|<转向语句>|<返回语句>|<算完语句>|<打破语句>|<循环语句>|<条件执行语句>|<条件跳出语句>|<

开关语句>|<控制台开关语句>|回车换行<基本语句>

㈢函数体返回语句、函数体结束语句

<函数体返回语句>::=RETURN

<函数体结束语句>::=$\{*\}_1^n$

函数体返回语句使函数体的执行结束，返回函数调用处。

函数体结束语句蕴含一个函数体返回语句，并且关闭所有未关闭的函数体。函数体可以有若干个入口，最后用一个公共的函数体结束语句来关闭。关闭的含义是，形参和局部量的作用域到此为止。一个函数体的不同入口的形参表和局部量表应该相同。

㈣表达式语句

<表达式语句>::=<表达式>

表达式语句使执行表达式一次，以该表达式的值作为程序的当前值。

㈤转向语句、返回语句、算完语句

<转向语句>::=GO<+|-><局部标号>|GO<标号>

<返回语句>::=BACK <+|-><局部子程序入口>|

BACK <子程序入口>

<算完语句>::=END

转向语句使无条件转去该局部标号或标号执行。对于局部标号，+表示下头最近一个，-表示上头最近一个。

返回语句使该局部子程序或子程序的执行结束，返回调用处。对于局部子程序入口，+表示下头最近一个，-表示上头最近一个。

算完语句使电传输出

SUAN WAN

然后等待电传命令。

(六)打破语句、循环语句

<打破语句>::= BREAK <无符号整数>

打破语句出现在循环语句中，其作用是打破由无符号整数的值指明的若干层循环，跳到该层循环语句之后。无符号整数缺席时认作1。

<循环语句>::=<DO 语句>|<DO WHILE 语句>|<DO UNTIL 语句>|<WHILE 语句>|<UNTIL 语句>|<FOR 语句>

<DO 语句>::=DO <语句> 循环执行该语句，直到被转出去为止。

<布尔表达式>::=<表达式> 其值是逻辑值，或者能产生与逻辑值同效的转移特征。

<DO WHILE 语句>::=DO <语句>WHILE <布尔表达式> 这里语句不允许是DO语句。先执行语句一次。若未被转走，则计算布尔表达式。若未被转走，且其值为 TRUE (或者能产生与 TRUE 同效的转移特征，下同)，则还执行语句一次。如

此循环，直到被转走，或者布尔表达式的值为 FALSE 时为止。

<DO UNTIL语句>::= DO<语句> UNTIL <布尔表达式>

同上，但布尔表达式值为 FALSE 时循环。

<WHILE 语句>::= WHILE<布尔表达式> DO <语句>

同 DO WHILE 语句，但要先计算布尔表达式，而不是先执行语句一次。

<UNTIL 语句>::= UNTIL <布尔表达式> DO <语句>

同 DO UNTIL语句，但要先计算布尔表达式。

<FOR 语句>::= FOR <简单变量> :=<循环表> DO <语句>

<简单变量>::=<变量标识符>

<变量标识符>::=<标识符>

<循环表>::=<循环表元素> | <循环表>，<循环表元素>

<循环表元素>::=<表达式> | <表达式> { STEP <表达式> }$_0^1$ <UPTO | DOWNTO><表达式>

FOR语句类似ALGOL 60的循环语句，但用 UPTO 和 DOWNTO代替 UNTIL 。这使得在判断步长型元素用完没有时不考虑

步长的符号。步长若缺席分别认作1和－1。步长型元素用完时简单变量的值已超出终值范围。

(七)条件执行语句、条件跳出语句、开关语句、控制台开关语句

<条件执行语句>::=<布尔表达式>DO{/}'。<语句>

先计算布尔表达式，若未被转走，且值为无/时 TRUE 、有/时 FALSE ，则执行语句一次，否则不执行。

<条件跳出语句>::=<布尔表达式>=:{/}'。<表达式>

<无符号整数>　　无符号整数若在园括号之后必须用空格隔开。

先计算布尔表达式，若未被转走，且值为无/时 TRUE 、有/时 FALSE，则执行表达式一次，然后跳出由无符号整数的值指明的若干层程序段，无符号整数缺席时认作1。否则不执行表达式，也不跳出。程序段是用园括号括住的一段程序，请参看第四节。

<开关语句>::= SWITCH <表达式>ON<开关体>

<开关体>::=<CASE语句列>{<;|回车换行>

ELSE: <语句>}'。

<CASE语句列>::=<CASE 语句>|<CASE语句列>

<;回车换行><CASE 语句>

<CASE语句>::= CASE <比较对象表>:<语句>|

回车换行< CASE 语句>

<比较对象表>::=<比较对象>|<比较对象表>,<比较对象>

<比较对象>::=<单字长量>|<单字长量>-<单字长量>

<单字长量>::=<行>|{-}!。<无符号整数>|<简单变量>

单字长量必须能够容纳在一个机器字内。

开关语句的执行，先计算表达式，然后将其值依次与开关体中诸CASE语句的比较对象比较，迁到相符合时，即执行相应的语句，然后跳出此开关语句。所谓符合，就是相等，或者，对于<单字长量>-<单字长量>这样的比较对象，就是落在该闭区间里。如果都不符合，则执行带ELSE：的那个语句(如果有的话)。

<控制台开关语句>::=ON <无符号整数><单语句>

<单语句>::=<子程序调用>|<转向语句>|<返回语句>|<函数体返回语句>|<打破语句>|<算完语句>

控制台开关语句检查无符号整数的值所指明的那个控制台开关，若合上则执行单语句，否则不执行。

四 表达式

<表达式>::=<多项式>|<表达式>=<多项式>

如果有符号，则先计算表达式和多项式，然后将他们的值进行相等比较，得到逻辑值TRUE 或FALSE。相等比较是广义的，它能判断两个数、行、逻辑值、表、矢或函数名的相等。

<多项式>::={<+|->}!。<项>|<多项式><+|-><项>|回车换行<多项式>

<项>::=<因式>|<项><*|/><因式>

<因式>::=<初等量>|<因式><**|//><初等量>

+、-、*、/、**都要求运算量是数。对于整数除法，如果除不尽，则结果是分数。**的右运算量必须是整数。//是行的连接运算，要求两个运算量都是行。

<初等量>::=<当前值>|<程序段>|<常量>|<变量>|<子程序调用>|<函数调用>|<赋值量>|<快速量>|回车换行<初等量>

<当前值>::=( )　　程序的当前值。

<程序段>::=(<程序>{回车换行}$_0^1$)|<无符号整数>(<程序>{回车换行}$_0^1$)<无符号整数>|〔<基本汇编程序>〕

两个无符号整数必须相同，且与园括号之间不许有空格隔开。程序段的计算完成后程序的当前值，被规定作该程序段的值。所谓程序段的计算完成了，是指执行完该程序段的最后一个语句，或者被一个条件跳出语句跳出该程序段。

<常量>::=<无符号整数>|<行>|<逻辑值>|NIL|<函数名>

<逻辑值>::=TRUE|FALSE

NIL就是TRUE，它常用作表终标志，有时称为空表，但并不是表，既无头也无尾。

<变量>::=<简单变量>|<下标变量>|<表变量>|<函数名>

<简单变量>::=<变量标识符>　　简单变量如果不是形参

和局部量，就一概是全程量，且其初值为0。

<变量标识符>::=<标识符>

<下标变量>::=<初等量>〔<下标表>〕　这里初等量的值必须是矢。下标表中每个下标的值应是自然数。

<下标表>::=<下标>|<下标表>，<下标>

<下标>::=<表达式>

<表变量>::=<头>|<尾>

<头>::= HD（表达式>)　这里表达式的值必须是表。

<尾>::= TL(<表达式>)　这里表达式的值必须是表或整数。对于后一情形，尾就是以该整数为地址的存贮单元。

<子程序调用>::= CALL(<+|-><局部子程序入口>)|CALL(<子程序入口>)　从局部子程序入口或子程序入口所规定的非递归子程序的起点进入该子程序。进入时，程序的当前值未破坏。直到迂见返回语句时，返回子程序调用处。返回时程序的当前值，被规定作该子程序调用的值。

<函数调用>::=<初等量>({<实参表>}$_0'$)

<实参表>::=<实参>|<实参表>，<实参>

<实参>::=<表达式>

函数调用的执行是，先计算初等量，它的值应是函数名，指示一个函数体入口。然后依次计算诸实参，把函数的诸形参赋以对应实参的值。

实参多余时，多余实参的值废弃不用。然后从函数体入口进入函数体执行，直到迂见函数体返回语句，返回函数调用处。返回时程序的当前值，被规定作该函数调用的值。形参和局部量都局部于函数体。另外，实参与形参的结合方式，可被特定的内部子程序改变，请参看第六节。

<赋值量>::=<变量>:=<表达式>　先计算变量，得到一个地址。然后计算表达式的值，赋给该地址指示的存贮单元。于是，变量的类型也由赋值决定。表达式的值，被规定作赋值量的值。另外，如果变量是<简单变量>、<标识符>[<无符号整数>]、<标识符>[<标识符>]、HD(<标识符>)或TL(<标识符>)，那末程序的当前值未破坏，可供表达式使用。例如

X:=()　　把程序的当前值赋给 X。

X:=()+1　　把程序的当前值加1后赋给 X。

<快速量>::=.-<单字长量>|<初等量>。<+|-|*|=|GR|LS|OR|AND|EOR|LL|RL|LC|RC|LOAD|STORE|IQR><单字长量>

.+、.-和.*运算结果必须是单字长的整数。

.=、.GR和.LS分别是全等比较、整数大于比较和整数小于比较，运算结果是左运算量的值，但还产生与 TRUE 或 FALSE 同效的转移特征。

.OR、.AND和EOR分别是逻辑加、逻辑乘和按位加。

.LL、.RL、.LC、和RC分别是左逻辑移位、右逻辑移位、左循环移位和右循环移位。

.LOAD和.STORE分别是取到乘商寄存器和将乘商寄存器内容送内存。.IQR是整数留余除，左运算量是累加器和乘商寄存器的内容，商在乘商寄存器，余数在累加器。例如

0.LOAD 7.IQR 2.STORE X的值是1，X的值是3。

五 对控制转移的限制

FCY的控制结构很丰富，但有三种情况一般是不允许的：一是从函数调用的实参转出去，一是从下标变量的下标转出去，一是从右运算量转出该运算以外。反过来，从外头转入上述三种成分一般也是不允许的。

六 内部子程序

㈠使函数能够接受任意个实参的子程序

凡恰有一个形参的函数体入口，如果以CALL(PP)作为第一个语句，则该函数自动扩充为能够逐个处理任意多个实参。例如

*TYPE(A)(B):CALL(PP)

.....

*

则TYPE(X,Y,Z)等价于TYPE(X);TYPE(Y);TYPE(Z)。

凡恰有两个形参的函数体入口，如果以CALL(PM)作为第一

个语句，则该函数自动扩充为能夠按左结合方式处理任意多个实参。例如

```
*SUB(A,B):CALL(PM)
A-B
*
```

则SUB(X,Y,Z)等价于SUB(SUB(X,Y),Z)。

凡恰有两个形参的函数体入口，如果以CALL(MP)作为第一个语句，则该函数自动扩充为能夠按右结合方式处理任意多个实参。例如

```
*POWER(A,B):CALL(MP)
A**B
*
```

则POWER(X,Y,Z)等价于POWER(X,POWER(Y,Z))。

㈡宽行准备和关闭

```
CALL(OPEN)      准备宽行输出。
CALL(CLOSE)     关闭宽行。
```

㈢从指定的列开始输出

```
CALL(TYPEC)
CALL(WIDEC)
```

以上分别使电传输出和宽行输出从程序的当前值所指定的列开始。

七 内部函数

AD(X)　　简单变量 X 的地址。

ATOM(X)　　判断 X 是否原子，其反面是表或矢。

TABLE(X)　　判断 X 是否表。

NULL(X)　　判断 X 是否NIL。

NUMBER(X)　　判断 X 是否数。

NAME(X)　　判断行 X 是否标识符。

SIZE(X)　　矢 X 所含元素的数目。

LENGTH(X)　　行 X 所含字符的数目。

DIGIT(X)　　整数 X 有多少位数字。

STRING(X)　　非负整数X转换成的行。

CHAR(X,Y)　　行 X 的第 Y 个字符。

CONS(X,Y)　　新产生的表，以 X 和 Y 的值为其头和尾。

VEC(...)　　新产生的矢。若只有一个实参，则此实参的值规定该矢有多少个元素，且诸元素的值皆是0。否则，以诸实参的值为元素，因而，元素的数目等于实参的数目。若要产生只有一个元素X的矢，可借用LIST(X)。

LIST(...)　　新产生的表。其头是第一个实参的值，尾的头是第二个实参的值，依次类推，最后的尾是NIL。最后的尾是NIL这样的表最常见，以后我们称这种表为列表。

PUSH(...)　　设共有N+1个实参，则造一新表

LIST(后N个实参)，其最后的尾被赋以第一个实参值的尾，然后将此表赋给第一个实参值的尾，并且作为这函数调用的值。

POP(...)　设共有N+1个实参，则将第一个实参值的第2至第N+1个头依次赋给其余N个实参值的尾，然后将第一个实参值的第N+1个尾赋给第一个尾，并且作为这函数调用的值。

若X是化简了的分数 $\pm a\frac{c}{b}$，则

INT(X)　$\pm a$，

DEN(X)　b，

FRA(X)　$\pm\frac{c}{b}$，

NUM(X)　$\pm(a*b+c)$，

ABS(X)　$a\frac{c}{b}$。

SIGN(X)　$\begin{cases} 1 & X>0 \\ -1 & X<0 \\ 0 & X=0 \end{cases}$

MOD(X,Y)　整数X除以整数Y所得的余数，符号同X。

IQ(X,Y)　整数X除以整数Y所得的商，余数舍去。

IQR(X,Y)　整数X除以整数Y所得的余数和商构成的矢。

AND(...)　诸实参值的逻辑乘。

OR(...)　诸实参值的逻辑加。