DJS—6机

算法语言讲义

北京工业大学計算站

1974.12

毛主席语录

为什么人的问题，是一个根本的问题，原则的问题

前　言

这本讲义是在我校1974年3月《*DJS*—6机算法语言与操作使用讲义》的基础上改编而成的。我们编写这本讲义的目的，是希望推广*DJS*—6机算法语言和供非专业人员自学；所设想的对象是没有接触过电子计算机和算法语言的广大用戶。因此，在编写过程中，我们在主观上力求做到：内容简明，突出重点；叙述通俗，力求详尽；例子尽量多些；上机操作步骤也力求具体。

讲义内容主要包括三大部份。第一部份是算法语言，介绍了利用*DJS*—6机算法语言编写程序的方法，还通过例子指出了编写程序时应注意的一些问题，本部份也可作为学习*ALGOL*—60的入门。第二部份是上机操作，介绍了源程序和数据的书写和穿孔方式，上机操作的步骤、错误的查找等问题；第三部份是一些常用的算法。为了方便用戶，我们把一些最常用的算法编成了过程，在这部份中提供了这些算法的程序，介绍了它们的功能和使用方法。

讲义还包括三个附录。在附录1中我们介绍了数的二进制、八进制表示法及其在*DJS*—6机上的应用。附录2中介绍了代码语句，可供掌握*DJS*—6机指令系统的同志使用。附录3对*DJS*—6算法语言、*DJS*—21算法语言、*BCY*—乙算法语言进行比较。

讲义中的内容有些是基本内容，有些是属于进一步学习时应掌握的内容，对于后者，我们都加上“＊”号以示区别。同志们可以在通过一定阶段的学习和上机实践之后，再回头学习这些打“＊”的内容。这对于进一步理解算法语言的一些基本概念、灵活地编制程序、缩短调试程序的周期将是有益的。

讲义中的例子，上机操作注意事项，常用算法，代码语句，以及有关说明书和编译带方面的材料，都曾在我校的*DJS*—6机上进行过试算，并根据我们的实践对其中某些部份进行了修改或订正。

在编写本讲义的过程中，京字115部队、后字414部队和国营北京有线电厂编的“*DJS*-6机算法语言使用手册”，京字115部队编的“*DJS*—6机算法语言介绍”和南开大学编的“*DJS*—6机算法语言及应用”，为我们提供了宝贵的资料。校内外许多同志，如后字414部队的同志，对我们进行了许多具体的帮助，在此对上述单位和同志表示衷心的感谢。

我校计算站成立不久，程序工作刚刚开展，水平又低，讲义中肯定还存在着许多错误和不妥之处，我们恳切地希望阅读本讲义的同志和广大用戶对讲义提出宝贵意见，帮助我们改进工作，让我们遵照毛主席的教导：“**团结起来，争取更大的胜利**”。

北京工业大学计算站

1974年10月

目　录

第三部份　常 用 算 法

毛 主 席 语 录

中国人民有志气，有能力，一定要在不远的将来，赶上和超过世界先进水平。

绪　　言

一、DJS—6 机简介

DJS—6 机（*DJS* 分别是电子、计算机和数字三词的汉语拼音的第一个字母）又称 108—乙机，是我国生产的一台中型晶体管电子计算机。运算方式为二进制并行运算，指令为单地址指令。可作定点半字长运算与浮点全字长运算，定点运算平均每秒七万二千次，浮点运算平均每秒五万四千次。

DJS-6 机的主要部份的功能与规模如下：

1. **输入器**　有光电机输入与电传打字机输入两种。前者的功能是将穿孔纸带上所代表的程序或数据的信息送入机器的存贮器，速度为 800～1000 个字符/秒；后者一般用于从控制台上输入信息，输入速度很慢。

2. **存贮器**　是机器的最主要部分，它是用来存放程序或数的装置，有内存贮器与外存贮器两部份：

DJS—6 机配有两组大磁心体作为内存贮器，它们分别装在两个机柜里。每组磁心体含有 2×16384 个半字长单元(以下谈到容量问题，都是指半字长单元)，因此，共有 $2\times2\times16384$ 个单元。每个单元是由 24 颗磁心串成的一个磁心串，由于每颗磁心有两种不同的状态，因此表示了一个二进位的两个不同的数字 0 和 1 。

DJS—6 机一般配有两台磁鼓作为外存贮器，每鼓容量为 $2\times2\times16384$ 个单元，鼓的转速为 1500 转/分，可以扩充至四台。另外，有些机器还配有磁带存贮器，最多可装八台磁带机。采用 1 吋(25.4*mm*)的具有 16 条信息道的磁带。每条磁带的容量可达 $32\times2\times16384$ 个单元。外存贮器的特点是容量大，但是，由于存、取的速度较慢，因此一般算题都是利用内存贮器，外存贮器只是用作存放机器所需的固定信息（如编译程序，程序库等）或当计算大型题目时弥补内存贮器的不足之用。

这里要特别指出的是，存贮器有一个重要的性质：从它的每一个单元中取出信息后，该单元中仍保留其原有信息。但是，当送入一个新的信息后，则原来的信息即被破坏，而被新送入的信息所取代。

2. **控制器**　它的功能是对程序中的每条指令进行分析、判别，决定机器要执行那一种操作，然后对有关部份发出命令，它是指挥机器各部份的“神经中枢”。

4. **运算器**　各种运算操作在运算器内进行运算。

5. **输出器**　一般有窄行快速打印机和电传打字机两种。快速打印机打印速度为 15行/秒；电传打字机可以输出 52 种字符，输出速度为 400 个字符/分钟。

有些机器还配备有宽行打印机和 X—Y 绘图仪。

二、什么是算法语言？

在电子计算机产生初期，人们在使用它进行计算时，是直接根据机器指令来编写程序的，这样编出的程序通常称之为“手编程序”。采用手编程序有很大的缺点，首先是工作量太大，机器几分钟能算完的题往往要用几个星期来编程序，因而需要一支庞大的编程序的队伍。其次，手编程序是由一大堆枯燥的数字组成的，很不直观，不便于阅读和检查，而且非常容易出错。最后，手编程序是用机器指令编成的，它对机器的依赖性很大，同一个题目，改用另一种型号的机器来计算时，原有的程序就不适用了，必须重新编制程序。

于是从1956年起，各国陆续采用算法语言来编程序。所谓算法语言，包括一套规定好的基本符号和由这套基本符号来构造程序的规则，使得按这套基本符号及构造规则所描述的计算过程与我们日常书写的计算过程较为接近。当我们要用计算机进行计算时，就按这套语言来描写计算过程，这样写出的程序称为源程序。源程序写好后，将它打在纸带上，再输入到机器中，利用存放在机器内的编译程序(编译程序是人工编好的一份庞大的程序，它专门用于将源程序翻译成机器指令的形式）将源程序翻译成机器指令程序（称为目标程序），然后让机器进行计算。

目前，国际上使用的算法语言种类很多，在科学计算中常用的有下面两种：

ALGOL—60 与 *FORTRAN*

这里 *ALGOL* 是 *ALGOrithmic Language* 的缩写，即算法语言之意；*FORTRAN* 是 *FORmula TRANslation* 的缩写，即公式翻译之意。

我们介绍的是在 *DJS*—6 机上使用的算法语言，它接近于 *ALGOL*—60，但作了一些增减。掌握算法语言要比掌握手编程序技巧容易得多，因此它大大地促进了计算机的应用。另外，国产机器配备的语言绝大部份都与 *ALGOL*—60 很接近，相互间差异很小，掌握了一种机器上的算法语言后，只要再经过一些时间的钻研，也就能掌握另一种机器上的算法语言。这也使编制的程序在很大的程度上具有通用性，只要稍加修改就能在另一种型号的机器上使用。

使用算法语言解题，使我们从手编程序设计的繁重而机械的工作中解放出来，大大减少了编制程序的时间，同时也便于人们阅读、检查、修改。例如，要计算一个底面半径为 R，高为 H 的圆柱体体积和侧面积，只要写成下列形式：

```
'BEGIN'
        'REAL' R, H, V, S;
        READR(R, H);
        V:=3.1415926*R*R*H;
        S:=2*3.1415926*R*H;
        OUTPUTR(V, S)
'END'
```

它的意思是：

开始

(本题中有)实型数 R, H, V, S;

(从光电输入机上通过数据纸带)读入 R, H;

计算 πR^2H 的值并把结果送到 V 中;

计算 $2\pi RH$ 的值并把结果送到 S 中;

(在快速打印机上)输出计算结果 V, S ;

结束

这与我们平常所写的计算公式极为接近，因此，算法语言是易于为广大工农兵群众所掌握的。

在党的十大路线光辉照耀下，在批林批孔运动的推动下，我国计算战线形势大好，*DJS*—6 机算法语言的编译程序正在不断完善，编译程序的功能正在不断扩充，各套编译程序相继问世，为我们使用算法语言解题创造了良好的条件。这里我们介绍的 *DJS*—6 机算法语言使用的编译程序是 73 年上海推广会上所发的纸带（简称第二版），如果今后同志们使用更新版的编译程序，請注意其间的差别。

毛 主 席 语 录

入门既不难，深造也是办得到的，只要有心，只要善于学习罢了。

第一部份 算 法 语 言

§1 基 本 符 号

任何语言都有一些基本的符号，例如英语是由26个字母与一些标点符号构成的。*DJS*—6机的算法语言，也是由一些基本符号构成的。也就是说，今后用*DJS*—6机的算法语言编写出来的程序只允许出现这些基本符号，否则计算机就不能识别了。

DJS—6机的算法语言的基本符号共有86个。

(1) **字母**（共26个）

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

(2) **数字**（共10个）

0 1 2 3 4 5 6 7 8 9

(3) **逻辑值**（共2个）

'TRUE'（眞） *'FALSE'*（假）

(4) **定义符**（共48个）

1. **运算符**（共21个）

① 算术运算符（共6个）

+ − × ／ // ↑

② 关系运算符（共6个）

'GR'（相当于>） *'LS'*（相当于<） *'NQ'*（相当于≠）

'GQ'（相当于≥） *'LQ'*（相当于≤） =

③ 逻辑运算符（共3个）

'OR'（或） *'AND'*（与） *'NOT'*（非）

④ 顺序运算符（共6个）

'IF'（如果） *'THEN'*（则） *'ELSE'*（否则）

'FOR'（对于） *'DO'*（做） *'GOTO'*（转向）

2. **分隔符**（共9个）

：（冒号） ；（分号） ，（逗号）

．（小数点） ₁₀（小拾） :=（赋值号）

'UNTIL'（直到） *'STEP'*（步长） *'WHILE'*（当…的时候）

3. **括号**（共9个）

（ ） 两个构成一对圆括号

[　　]　两个构成一对方括号

'(　　')　两个构成一对行括号

'BEGIN'（开始）'END'（结束）　两个构成一对语句括号。

'CODE'（代码）

4. **说明符**（共 6 个）

'INTEGER'（整）　'REAL'（实）　'BOOLEAN'（布尔）

'ARRAY'（数组）　'PROCEDURE'（过程）　'SWITCH'（开关）

5. **分类符**（共 3 个）

'VALUE'（值）　'LABEL'（标号）　'STRING'（行）

以上基本符号中有许多符号的意义是明显的，有些是不熟悉的，关于它们的意义与作用将在后面各节中分别加以解释。这里先作下面的一些说明：

(1) 为什么有些习惯的符号不用，而重新造一些符号，例如“大于”要写成 'GR' 而不用“>”呢？这是因为 *DJS*—6 机要用的基本符号有 86 个，但是作为本计算机的输入与输出设备的 55 型电传打字机仅有 52 个符号可供使用，因此有些符号就只好用 52 个符号的某些组合来表示。

(2) 用字母拼成的基本符号其两端必须带“'”号，否则它就不是基本符号了。例如，'STEP' 与 STEP 在算法语言中是表示不同含义的，前者是基本符号，它有确定的意义，表示步长。后者则是由基本符号 *S*, *T*, *E*, *P* 构造而成的，它只能用来给一些量或句子起名字，而不能用来代表“步长”这个意思。

(3) 字母只采用英文字母，共 26 个。大写、小写不分。即 *A* 与 *a*，'INTEGER' 与 'integer' 等被认为是等同的。由于 55 型电传打字机上的键盘与打印出的字母都是大写的，因此本讲义在书写源程序时都用大写字母。

(4) 要注意区分形状相似而意义不同的符号，例如乘号“×”与字母“*X*”，数字“0”与字母“*O*”，…… 。因此，在书写时使用者可以自己设计记号加以区别，例如可用记号“*”代替乘号“×”，用记号“θ”代替字母“*O*”等等。在本讲义中，当书写源程序时用记号“*”代替乘号“×”。

§2　数

在科学计算的题目中，总是离不开数。在 *DJS*—6 机算法语言中如何表示一个数呢？

在 *DJS*—6 机算法语言中，数的表示形式与通常十进制的数的书写形式差别不大。例如

DJS—6 机算法语言中的数	十进制的数
51	51
−1665	−1665
+3.1416	+3.1416
12.0	12
$2_{10}4$	2×10^{4}
$12.1_{10}-3$	12.1×10^{-3}

$-145_{10}-6$　　-145×10^{-6}

数按其书写形式的不同分成两类：

(1) *'INTEGER'* 型（**整型**）　凡是只出现“+”，“−”号（“+”号可省略）和数字 0—9 的数是整型数。例如，在上面列举的数中只有前二个是整型数。

(2) *'REAL'* 型(**实型**)　非整型数(即带有小数点或指数部份的数）叫实型数。例如，在上面列举的数中的后五个是实型数。

由于计算机字长的限制，数的取值范围也受到相应的限制。在 *DJS*—6 机中，实型数用浮点表示，占两个单元（48位），其绝对值在 8.7×10^{-78}～2.8×10^{76} 之间。绝对值小于 8.7×10^{-78} 的数 *DJS*—6 机认为是 0，绝对值大于 2.8×10^{76} 的数 *DJS*—6 机容纳不下，产生溢出。整型数也占有两个单元，但有用的只是其中的一个单元（24位），故整型数的绝对值不能超过 8388607，即 $2^{23}-1$.

从整型数与实型数的取值范围看出，后者的取值范围远远大于前者。但是，整型数是完全精确地表示的，而实型数则往往带有一些误差，只能达到一定的精度。尽管误差很小，在许多情况下可以忽略，但毕竟不是完全精确地表示的。所以，两者各有利弊，在需要用完全精确表示的数时（例如控制循环次数），整型数是方便的。

对于数，我们再说明几点：

1. 数的类型是按书写形式来区分的，而不管它所表示的内容是整数还是实数。例如 10000 是整型数，$_{10}4$ 也是代表 10000，但它却是实型数，又例如 12 是整型数，12.0 也代表 12，但它却是实型数。

2. 与十进制表示的数一样，正数之前的“+”号可要可不要，无意义的“0”也是可要可不要。例如 +0.5，0.5，0.50，.5 都是相等的。

3. 小拾“$_{10}$”是一个基本符号，它本身并不表示一个数，即不等于“10”。它的后面必须是且只能是整数。例如，1.5_{10} 与 $1.5_{10}3.5$ 的写法都是不允许的。如果需要用 $1.5\times10^{3.5}$ 这个值，可以利用 §7 将要介绍的指数记号“↑”，把它写成 1.5↑10↑3.5。

§3 变量、标识符

算法语言中的变量概念与数学中的变量概念是一致的，即变量是指可以改变其取值的量。在数学中常用 X，Y 等来表示变量，而在算法语言中则用“标识符”来标识（或命名）变量。

变量可分为算术变量与布尔变量两类，而每一类变量又可分为简单变量与下标变量两种。我们在这里只讨论算术变量，即算术简单变量与算术下标变量（以下就简单称为简单变量与下标变量），关于布尔变量今后(§18)再进行讨论。

3.1 简单变量与类型说明

(1) **简单变量**　简单变量是指单个的变量，如等式

$$S=3.1416R^2$$

中的 S，R 就是简单变量。

简单变量用标识符来命名。

标识符是以字母开头的字母数字串。例如

$$X \qquad AI2B \qquad ROOT1$$

都是标识符。这里要特别注意的是，标识符必须以字母开头，且只能由字母与数字组成。因此

$$A3.5 \qquad A,B \qquad A+15 \qquad R\pi \qquad 5ABC$$

都不是标识符。

标识符的长度是没有限制的，但在 *DJS*—6 机的语言中仅前八个有区分意义。例如对于

$$ABC12DE46 \quad 与 \quad ABC12DE4FG$$

因为前八个符号相同（它们从第九个符号开始不同，长度也不一样），于是它们被认为是同一个标识符而不加以区别。其实标识符也没有必要搞得太长。

标识符的选取完全是随意的。一般讲，以选取符合于习惯，有助于理解的名称为佳。例如

T（时间）　　*SUM*（和）　　*ROOT*1（第一个根）

H（高度）　　*TERM*（项）　　*AVERAGE*（平均数）

在 *DJS*—6 机语言中，有一些标识符专门用于表示标准函数，表示输入、输出过程，表示控制台变量，表示已知地址量，例如

SIN　COS　LN　MAX　MIN

INPUT　READR　OUTPUTR　PRINTS　III　AAA

等（详见 §4，§17 和第二部份 §4），这些标识符不能再用于其它的目的。

同一标识符通常不用来表示多个不同的变量，因为这样会引起混乱和错误。但是，在一定的条件下也是允许的（详见 §12）。

今后还可看到，标识符不但可以作为变量的名称，而且还可以作为程序中其它对象的名称。

(2) **类型说明**　因为"数"分为整型和实型，所以变量的取值也有整型和实型之分。数的类型可以用其书写的形式来区分，而 *DJS*—6 机语言中变量的类型无法从标识符的书写形式来区分。因此，在变量的标识符首次出现之前必须对它的类型作出说明，这就是"类型说明"。

最基本的类型说明有两种：一种是 *'INTEGER'* 型，一种是 *'REAL'* 型（还有一种是 *'BOOLEAN'* 型，见 §18）。如果某变量的取值是整型数，则它就是 *'INTEGER'* 型的；如果取值是实型数，则它就是 *'REAL'* 型的。例如，若变量 *N* 是 *'INTEGER'* 型，而变量 *SUM* 是 *'REAL'* 型，那么类型说明的具体写法是：

'INTEGER' N

'REAL' SUM

如果有几个变量，例如 *N*，*COUNT*，*I*，*J* 都是属于 *'INTEGER'* 型的；*T*, *SUM*, *TERM* 都是属于 *'REAL'* 型的，那么它们可以一个个分别说明，也可以合起来说明（一般是合起来说明），而写成

'INTEGER' N, COUNT, I, J

'REAL' T, SUN, TERM

这里要注意的是，在几个变量合起来说明时，变量间必须用逗号“，”分隔开。至于先说明整型还是先说明实型，说明时变量应按那一顺序进行排列，这倒是随便的。

3.2 下标变量与数组说明

(1) **下标变量** 在解决实际问题时，常常会遇到按一定顺序排列的一组量，其中每一个量在计算过程中可以取不同的值，因此，这些量本身一般讲都是变量。我们称这些量的全体为一个数组。

例 1. 当进行数值计算时，常会遇到 n 次多项式

$$P(x)=c_0x^n+c_1x^{n-1}+\cdots+c_{n-1}x+c_n\ .$$

这时，我们可以把它的 $n+1$ 个系数组成一个数组：

$$(c_0,\ c_1, c_2, \cdots,\ c_{n-1},\ c_n).$$

例 2. 在齿轮设计中，常会遇到摆线方程

$$\begin{cases} x=t-\sin t, \\ y=1-\cos t. \end{cases}$$

这时，如果我们要计算当 $t=0,\ 0.05,\ 0.1,\ 0.15,\ \cdots,\ 1$ 时 $x,\ y$ 的对应值：

$$(x_0,\ x_1,\ x_2,\ \cdots,\ x_{19},\ x_{20})$$
$$(y_0,\ y_1,\ y_2,\ \cdots,\ y_{19},\ y_{20})$$

那么，可以把这些 x 看成一个数组，把这些 y 又看成一个数组。

例 3. 在解含多个（例如是三个）未知数的线性代数方程组

$$\begin{cases} a_{11}x_1+a_{12}x_2+a_{13}x_3=b_1 \\ a_{21}x_1+a_{22}x_2+a_{23}x_3=b_2 \\ a_{31}x_1+a_{32}x_2+a_{33}x_3=b_3 \end{cases}$$

时，它的解、常数项、未知数的系数都可以看成是数组：

$$(x_1,\ x_2,\ x_3),$$
$$(b_1,\ b_2,\ b_3),$$
$$\begin{pmatrix} a_{11} & a_{12} & a_{13} \\ a_{21} & a_{22} & a_{23} \\ a_{31} & a_{32} & a_{33} \end{pmatrix}$$

我们也可以把后两个数组看成是一个数组：

$$\begin{pmatrix} a_{11} & a_{12} & a_{13} & b_1 \\ a_{21} & a_{22} & a_{23} & b_2 \\ a_{31} & a_{32} & a_{33} & b_3 \end{pmatrix}$$

例 4. 晶体二极管的伏—安特性(即加在二极管两端的电压与流过二极管的电流的关系)是 $I=A_s(e^{qV/TK}-1)$,

其中 A_s 是反向饱和电流，q 是电子电荷，K 是玻尔兹曼常数。如果在不同温度

$$T=T_1,T_2,T_3$$

下，计算当电压

$$V=V_1,V_2,V_3,V_4,V_5$$

时电流 I 的对应值，那么这时 T，V 所取的值分别是一个数组，对应的 I 值也是一个数组：

$$\begin{pmatrix} I_{11} & I_{12} & I_{13} & I_{14} & I_{15} \\ I_{21} & I_{22} & I_{23} & I_{24} & I_{25} \\ I_{31} & I_{32} & I_{33} & I_{34} & I_{35} \end{pmatrix}$$

其中 $I_{jk}(j=1,2,3;k=1,2,\cdots,5)$ 表示在温度 $T=T_j$ 下当 $V=V_k$ 时对应的电流。

例 5. 在解决生产实际问题时，常要用到向量与矩阵，它们也都是数组。

在 *DJS*—6 机语言中，数组是用标识符来命名的。如例 3 中的三个数组，我们可以分别用标识符

$$X,\quad B,\quad A$$

来代表它们。

数组中的元素都是带有下标的，我们称为下标变量。如何表示下标变量呢？在 *DJS*—6 机语言中，下标值用方括号括起来，而在前面冠以所在的数组的标识符。例如

$$X[1],\quad X[2],\quad X[3]$$

是三个下标变量，它们分别代表数组 X 中的第 1、第 2、第 3 个元素。

如果某下标变量有不止一个下标，则把这些下标都写在方括号中，相互之间用逗号“，”隔开，而在前面冠以数组标识符。例如

$$A[1,2],\quad A[2,3]$$

分别代表数组 A 中的 a_{12}，a_{23} 两个元素。

下标的个数称之为数组的维数。因此，下标变量 $Y[3,4]$ 所在的数组 Y 是二维的，下标变量 $AB[1,2,3]$ 所在的数组 AB 是三维的等等。

从下标变量的概念看出，它实际上只不过是带下标的变量，所以从本质上讲它与简单变量是一样的。因此在程序中凡是简单变量能出现的地方，下标变量也理应能出现。但是，在 *DJS*—6 机语言中，只在循环语句与调用过程时，才对它有所限制。关于这一点，我们将在 §11.4 和 §16.2(1) 中分别加以叙述。

(2) **数组说明** 与简单变量一样，凡在程序中出现的下标变量都必须进行说明。但是，由于下标变量是某数组中的一个元素，因此对下标变量就不需逐个说明，而只要对数组进行说明，这就是“数组说明”。

数组说明指出某标识符是代表数组的，指出数组中元素的类型，以及数组的维数和下标的范围。例如

```
'INTEGER' 'ARRAY' I[1:4]
'REAL' 'ARRAY' A[1:2, 1:3]
```

在第一个例子中，用 *'ARRAY'* 说明 I 是数组标识符，用 *'INTEGER'* 说明 I 中的元素都是整型的，方括号内的 1 是“下界”，4 是“上界”，中间用冒号“:”分隔开，它说明

下标取值的范围是从1到4，即整数数组 I 的下标变量是 $I[1]$, $I[2]$, $I[3]$, $I[4]$ 等四个，它只有一对上、下界，因此这个数组是一维数组。第二个例子与其类似，只是出现两对上、下界，彼此间用逗号“,”分隔开，因此是二维数组，整个数组包含了 $A[1,1]$, $A[1,2]$, $A[1,3]$, $A[2,1]$, $A[2,2]$, $A[2,3]$ 这六个实型下标变量。

数组说明的一般形式是

〈类型〉 *'ARRAY'* 〈标识符〉$[B_1:C_1,\cdots,B_N:C_N]$

其中〈类型〉是指数组的类型是*'REAL'* 还是 *'INTEGER'*（还有一种 *'BOOLEAN'* 型，详见§18）。若为实型，则 *'REAL'* 可写可不写。*'ARRAY'* 是数组说明符。〈标识符〉是该数组的名字。方括号内的内容

$$B_1:C_1,\ B_2:C_2,\ \cdots,\ B_N:C_N$$

称为界偶表。界偶表由界偶组成，每对界偶之间用“,”分隔开，界偶的个数即是数组的维数，也就是下标的个数；界偶中 B_1, B_2, …, B_N 称为下界，C_1, C_2, …, C_N 称为上界；界偶的下界不能大于上界，且上、下界只取整型值，若不是整型值，则机器自动按四舍五入取整型值（即向最靠近的整数舍入），例如 $A[-2.6:4.3]$ 相当于是 $A[-3:4]$。

最后，我们指出，如果几个数组都是同一类型的，那么可以合在一起说明，例如

'ARRAY' $A[1:10]$, $B[3:8,\ 1:15]$

如果几个数组不仅类型相同，而且维数和上、下界也都相同，则可简写成

'INTEGER' 'ARRAY' $A,\ B,\ C[3:10]$

§4 标准函数

在一般的计算问题中，常常会遇到一些函数。对于一些常用的函数，*DJS*—6 机已予先配备好，用到时只要写出它的名字和自变量就可以了。对于这些函数的名字（即标识符），使用者无需（也不允许）在程序中予先说明。这些函数叫做标准函数。

DJS—6 机算法语言中有以下标准函数：

标准函数符	意　义
$SIN(E)$	E 值的正弦
$COS(E)$	E 值的余弦
$TAN(E)$	E 值的正切，其中 $E \neq K\pi+\frac{\pi}{2}$ （K 为整数）
$ARCSIN(E)$	E 值的反正弦的主值，其中 $\lvert E\rvert\leqslant 1$
$ARCTAN(E)$	E 值的反正切的主值
$EXP(E)$	E 值的指数函数 e^E
$LN(E)$	E 值的自然对数，$E>0$
$SQRT(E)$	E 值的平方根 $\sqrt{E}$，$E\geqslant 0$
$CUBT(E)$	E 值的立方根 $\sqrt[3]{E}$
$ABS(E)$	E 值的绝对值 $\lvert E\rvert$
$SIGN(E)$	E 值的符号函数，即

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

$ENTIER(E)$　　E 值的整数部份，它是取向零方向舍入后的整数值，结果是整型量。例如

$ENTIER(5.4)=5$

$ENTIER(5.9)=5$

$ENTIER(-5.4)=-5$

$ENTIER(-5.9)=-5$

$MAX(X,Y)$　　X,Y 中的大数

$MIN(X,Y)$　　X,Y 中的小数

注意 1. 在使用标准函数时，圆括号切不可丢失。例如，$SIN(E)$表示 E 的正弦，而 $SINE$ 则是一个普通的标识符。

2. 要用 SIN, COS, TAN 这三个标准函数时，必须先把自变量化为弧度。

最后指出，在 DJS—6 机语言中的标准函数，E 可以是任意的算术表达式（见§7.2）。因此，允许出现标准函数相套的情形。例如

$$SIN(SIN(X)),$$

等等。

§5 简单的输入、输出语句

要解决一个实际问题，需要先整理出计算公式和已知数据。我们先把计算公式编成源程序，输入到机器，当源程序中的计算过程需要用到数据时，就由输入语句把已用穿孔机穿在纸带上的数据输入到机内使用。当机器遇到输出语句时，就把语句中指定的输出对象的值打印在纸上。

我们先介绍一些较简单的输入、输出语句。关于输入、输出标准过程将在§17 中专门介绍。在本节中，$C1$, …, CN, $A1$, …, AN 都代表标识符。

5.1 输入（光电输入机）

(1) 输入常数

语句　　$READR(C1, C2, \cdots, CN)$

表示从光电机上输入 n 个实型数，分别送到 $C1$, $C2$, …, CN 中。

语句　　$READI(C1, C2, \cdots, CN)$

表示从光电机上输入 n 个整型数，分别送到 $C1$, $C2$, …, CN 中。

例如，计算某题时，遇到 X, Y, A, B 四个实型量。它们的初始值分别为 0.1，−3.2，$_{10}5$，−15.1，为了把这些数值分别送到机器中相应的位置上，只要在源程序中写上如下的输入语句：

$$READR(X,Y,A,B)$$

并将 0.1，−3.2，$_{10}5$，−15.1 四个数依次用穿孔机穿在数据纸带上（穿孔方法见本讲义第

二部份）。源程序输入后，我们将数据纸带装到光电机上，当计算机执行到输入语句时就会自动地启动光电机将数据纸带上的数据顺序“读入”并放入指定的标识符所占的单元中去。因此，上面语句执行的结果是

$$X=0.1,\ Y=-3.2,\ A=10^5,\ B=-15.1\ 。$$

又例如，当机器执行输入语句

$$READI(A, AA, B)$$

时，如果装在光电机上的数据纸带中的数依次为

$$12,\ -5,\ 24,$$

那末，上述输入语句执行后，A，AA，B 三个标识符中的值分别为

$$A=12,\ AA=-5,\ B=24\ 。$$

(2) 输入数组

语句　　$INPUT(A1,\ A2,\ \cdots,\ AN)$

表示从光电机上输入 n 个实型数组，分别送到 $A1, A2, \cdots, AN$ 中。

语句　　$INPUTI(A1,\ A2,\ \cdots,\ AN)$

表示从光电机上输入 n 个整型数组，分别送到 $A1, A2, \cdots, AN$ 中。

5.2 输出（快速打印机）

(1) 输出常数

语句　　$OUTPUTR(C1,\ C2,\ \cdots,\ CN)$

表示在快速打印机上打印出 $C1$，$C2$，$\cdots$，CN 的值（实型数）。

语句　　$OUTPUTI(C1,\ C2,\ \cdots,\ CN)$

表示在快速打印机上打印出 $C1$，$C2$，$\cdots$，CN 的值（整型数）。

(2) 输出数组

语句　　$OUTAR(A1,\ A2,\ \cdots,\ AN)$

表示在快速打印机上打印出 n 个数组 $A1$，$A2$，$\cdots$，AN 的值（都是实型数）。为了便于区分与查找，每个数组的值打印完后，机器自动空开五行再打印下一数组的值。

语句　　$OUTAI(A1,\ A2,\ \cdots,\ AN)$

表示在快速打印机上打印出 n 个数组 $A1$，$A2$，$\cdots$，AN 的值(都是整型数)。为了便于区分与查找，每个数组的值打印完后，机器也自动空五行。

注意：在上面的输入、输出语句中，正整数 N 也可以是 1，即只输入（或输出）一个常数或一个数组。

§6 源程序的组成部份

根据毛主席“**由特殊到一般**”的教导，我们先从一个最简单的例子入手，以了解编写源程序的基本思想。

例如，已知 x，a 的值，要计算

$$y=\frac{x^2}{a+3}$$

的值。

首先，我们引用标识符 X，A，Y 分别表示 x，a，y 的值。其次，就应该对机器说明，X，A，Y 都是实型的，写成

$$'REAL'\ X,\ A,\ Y$$

然后把已知的 X，A 值输入到机器内去，由于 X，A 是实型数，故写成

$$READR(X,\ A)$$

已知值输入到内存后，可以按给定的算式进行运算，写成

$$Y:=X*X/(A+3)$$

这里在 Y 与 $X*X/(A+3)$ 之间用“:=”而不用“=”。冒等号是表示一个赋值过程，即把右边表达式的运算结果赋给左边的变量 Y，也就是把右边表达式的运算结果送到 Y 所占有的存储单元里去。因此，它与通常的“=”意义不同。例如，$Y:=-Y$ 表示把变量 Y 反其符号后再送回到原来 Y 的单元里去，而不是表示 $Y=-Y$，即 $Y=0$。

最后，将计算得到的结果通过快速打印机打印出来。由于 Y 是实型的，故写成

$$OUTPUTR(Y)$$

总的，这个问题的源程序的完整形式是：

```
'BEGIN'
    'REAL' X, A, Y;
    READR(X, A);
    Y:=X*X/(A+3);
    OUTPUTR(Y)
'END'
```

其中 $'BEGIN'$ 和 $'END'$ 分别表示源程序的开始和结束。

我们对这个源程序的结构进行分析，可以看出，它大体上由下面四部份组成：

(1) **开始部份**　　$'BEGIN'$

(2) **说明部份**　　$'REAL'\ X, A, Y;$

(3) **语句部份**　　$READR(X, A);$
$Y:=X*X/(A+3);$
$OUTPUTR(Y)$

(4) **结束部份**　　$'END'$

以后遇到的源程序不管如何复杂，它总是由这四个部份组成。通俗地说，此处 $'BEGIN'$ 与 $'END'$ 向机器宣告了程序的开始与结束；说明部份是通知机器，下面程序中需要用到以这些标识符命名的单元，請机器给他们安排位置（即分配地址）；语句部份是告诉机器要执行怎么样的一些动作。因为“开始部份”与“结束部份”比较简单，所以今后就着重介绍“语句部份”和“说明部份”。

最后，我们要特别指出，在这份简单的源程序中，我们在某些地方加上了分号“;”(加在那些地方?)。分号本身既不属于那一个说明，也不属于那一个语句，例如前面介绍类型说明，数组说明，输入、输出语句时，在它们的后面都沒有加上分号“;”。但是，*DJS*—6 机算