算法语言讲义

（719计算机）

复旦大学数学系编

1980年1月重印

说　明

编写本讲义的目的是为了普及719机算法语言的使用。主要用途是为适应工程技术人员短训班教学与理科一些专业的数学课教学的需要。因此，在内容上尽量选择719机算法语言中基本的常用的部分，在使用时，可以灵活掌握以下各点：

学完第一、二章即初步知识与若干基本语句后，就可以编些小题目上计算机实习。

学完第三章若干基本语句（续）后，作为较低限度的要求，就可以告一段落。

学完第四、五章即分程序、半动态数组、过程等之后，可以掌握719机算法语言中比较常用的部分。

第六章为若干补充，可根据需要情况选用一部分。

第七章为上机实习，可结合上机计算陆续讲。

对于本讲义中的缺点与错误，希望同志们批评指正。

复旦大学数学系

1976年1月

目　录

第一章　关于算法语言的初步知识

第一节　719电子计算机简介……（1）
第二节　关于计算机中的数与程序……（2）
第三节　什么是算法语言……（6）
第四节　算法语言的基本符号……（6）
第五节　标识符、变量及其说明等……（7）

第二章　表达式及若干基本语句

第一节　算术表达式与赋值语句……（11）
第二节　逻辑表达式与条件语句……（12）
第三节　句名与转向语句……（16）
第四节　输入与输出语句……（17）
第五节　程序例……（19）

第三章　若干基本语句（续）

第一节　复合语句……（25）
第二节　循环语句……（26）
第三节　循环语句（续）……（30）
第四节　空语句……（33）
第五节　停机语句……（34）

第四章　分程序和半动态数组

第一节　分程序……（37）
第二节　半动态数组……（38）

第五章　过程

第一节　函数过程……（44）

第二节　函数过程（续）……（47）
第三节　单用过程……（50）

第六章　若干补充

第一节　一些简化与补充事项……（56）
第二节　控制台变量……（58）
第三节　关于输入、输出格式的补充……（60）
第四节　719机算法语言其他内容一览……（62）

第七章　上机实习

第一节　穿孔……（64）
第二节　控制台面板简介……（66）
第三节　上机操作……（67）
第四节　上机操作（续）……（69）

附录一　编排错误检查……（72）
附录二　语法错误检查……（74）
附录三　非正常停机表……（76）

第一章 关于算法语言的初步知识

第一节 719电子计算机简介

电子计算机是一种快速执行算术运算和逻辑运算的工具。它进行数字运算的原理同我们通常的四则运算非常相似。电子计算机结构复杂，由于用电信号的转换进行操作，所以，计算速度非常快，比人要快上万倍甚至上亿倍，它在一秒种内可以进行上万次甚至上亿次的计算。但是，电子计算机不是万能的，不能代替人的全部脑力劳动，没有创造思想的能力，它只是在人的指挥下，起到计算工具的作用。

在我国，电子计算机的数量越来越多，应用范围也越来越广泛，它已成为国防、工农业生产和科技工作各方面的有力计算工具，如火箭、天气预报、化工、机械、地质勘测、石油、交通运输等都已使用了电子计算机。

719机是一台通用浮点计算机，用国产集成电路元件制成，运算速度每秒十二万五千次。它同一般的计算机一样，主要有运算器、存贮器、控制器、外部设备和控制台等五个部分，下面分别简单介绍这五个部分：

（一）运算器：

运算器有三个寄存器，记为“A寄存器”、“B寄存器”、“C寄存器”。A、B、C三个寄存器除了存放参加运算的代码外，还有各自的作用。例如：A寄存器是专门用来存放运算结果的，同时，与外部设备打交道也都是经过它的；B寄存器在运算中起了累加器的作用；C寄存器是专门同“内存贮器”打交道的，凡是要从存贮器取出一个数，总是先经过C寄存器随后送到A寄存器，同样，要存入一个数，也是先由A寄存器送至C寄存器，再送到存贮器。在计算过程中，A寄存器的内容不断变化。如果在A寄存器里得到的中间结果暂时不用，就必须送到存贮器里保存起来。运算器除了三个寄存器外，还有一组加法器，是具体实现四则运算的（+、−、×、÷都由加法器实现）。

（二）控制器：

控制器从存贮器里取出一条一条指令与数，根据程序把各种命令传送给计算机的其他部分，通过它去指挥各部件什么时候做什么事情，使计算机按一定节拍协调地工作。

（三）存贮器：

又叫内存贮器，是用来存贮代码的。它是一个由成千成万粒磁芯所构成的“磁芯体”，而每粒磁芯的体积比芝麻还小。存贮器划分成32768个单元，单元字长48位，单元编号为00000～77777。每一个单元只能存放一条代码。00000～00003单元分别对应于控制台单元S_0～S_3，00004单元对应于A寄存器。00005单元是计算机专用的单元。77700～77777是变址单元。

（四）外部设备：

包括输入器、输出器和外存贮器。输入器是光电输入机。初始数据和程序翻成代码，

象电报一样，经过穿孔机穿在纸带上，光照在纸带上，在有孔的地方透过去，射到光敏二极管上就变成了电信号。通过这种方式把代码输入到存贮器里，每秒钟可输入1180行。这个光电输入机还可相应地自动检查输入的正确性。这二台光电输入机对七单位和五单位两种穿孔纸带都能使用，其输入方式可为八进制、十进制、十六进制、指令型、符号六等五种。输出器是宽行打印机，每排可打印120个字符，打印数据的速度为24排/秒，输出方式可为八进制、十进制、指令型、符号六等四种（符号六输出速度为12排/秒）外存贮器是存贮器的一种，主要运用于存放大量暂时不参加运算的代码。当需要时，外存贮器的代码可成批地同存贮器交换，以解决内存贮器容量不够用的矛盾。外存贮器有二台立式磁鼓和二台磁带机，每台磁鼓的容量为一万四千多个单元，其转速为25转/秒，外存贮器存取代码的速度比内存贮器慢，应尽量不用外存贮器取代码。

（五）控制台：

我们上计算机算题主要是在控制台上进行操作的。控制台由扳键开关和氖灯所组成，当我们拨动某些扳键开关，就能操纵计算机，而控制台上的氖灯向我们显示出机器内部的工作状况。如：控制台上A、B、C三排氖灯，分别显示出A、B、C三个寄存器的状态。灯亮表示“1”，不亮表示“0”。下面“ZJ”旁的15位氖灯显示“指令计数器”的状态，表示机器将要执行那条指令的地址号码。控制台面板上四排扳键开关，对应于内存00000～00003号单元，每排有48个扳键对应于每个单元的48位代码，扳键向上表示“1”，扳键向下表示“0”。此外，还有按钮开关。“总清”按钮，一按就把控制台面板上指示的寄存器全部清“0”，只有“停机”和“T_{jk}”（T_{jk}表示指向控制台）二只灯亮。注意“总清”不是“内存总清”。“启动”按钮是命令计算机开始工作的，从那里开始做，由当时ZJ的状态决定。若是“总清”后再按“启动”，就从控制台上S_0的左指令开始工作。启动后是连续执行指令，还是做完一条指令后就停机，由“单指—连续—单拍”扳键开关的状态决定。“连续”——计算机启动后连续执行一系列指令；“单指”——计算机启动后执行一条指令后停机，A寄存器显示结果；“单拍”——计算机启动后执行一个节拍就停机，一般为调机人员用。对于控制台上的操作，我们在第七章逐步介绍。

第二节　关于计算机中的数与程序

为了学习算法语言与上机操作的需要，我们对于电子数字计算机中的数、指令、程序等概念应该有一个大略的了解。

一、十进制、二进制、八进制。

在日常生活中，通常采用十进制记数法。如：

$$256_{(10)}=2\times10^2+5\times10^1+6\times10^0,$$

$$2.375_{(10)}=2\times10^0+3\times10^{-1}+7\times10^{-2}+5\times10^{-3}.$$

在使用计算机时，我们将遇到二进制记数法与八进制记数法。如：

$$10.011_{(2)}=1\times2^1+0\times2^0+0\times2^{-1}+1\times2^{-2}+1\times2^{-3}=2.375_{(10)},$$

$$76_{(8)}=7\times8^1+6\times8^0=62_{(10)}.$$

从上述例子可知，二进制数的每一位数字由0、1中的一个构成，八进制数的每一位数字由

0、1、2、3、4、5、6、7 中的一个构成，一个数中不同位置上的数字具有不同的含义。

正如恩格斯所说：**“一切数的定律都取决于所采用的记数法，而且被这个记数法所决定。在二进位记数法和三进位记数法中，2×2 不＝4 而＝100 或 11。”** 在不同的记数法中，运算法则也是不同的，我们应该懂得这一点，至于具体的运算法则，对于使用算法语言算题关系不很大，我们不介绍了。

同一个数可以用不同的记数法来表示。它们之间存在着一定的转换关系。一般的转换方法与我们的关系也不甚大，但对于下列表格内的转换关系我们应当掌握：

十进制	0	1	2	3	4	5	6	7	8	9
八进制	0	1	2	3	4	5	6	7	10	11
二进制	000	001	010	011	100	101	110	111	1000	1001

从八进制 0——7 与二进制的转换关系中可以看到，八进制的一位数字恰与二进制的三位数字相对应，从而八进制数化成二进制数可以采用“一位拉三位”的方法，如：

$113.64_{(8)} = 001001011.110100_{(2)}$。

相反，二进制数化成八进制数时，应以小数点为基准，往左往右都是“三位并一位”，如：

$1110.01111_{(2)} = 16.36_{(8)}$。

二、内存单元中数的表示

电子计算机在运算时，一般要求把参与运算的数表示为二进制浮点形式：

$$x = \pm 2^{\pm p} \cdot q,$$

其中 p 是二进制正整数，称为阶码，p 前面的“＋”或“－”称为阶符，q 为二进制小数，称为尾数，整个数的“＋”或“－”称为数符。当 $q \geqslant \frac{1}{2}$ 时，该数称为“规格化数”，这时，尾数 q 的第一位应为 1 。如：

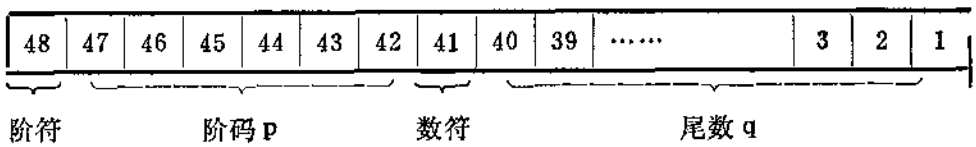
$2.375_{(10)} = 10.011_{(2)} = +2^{+10} \cdot 0.10011_{(2)}$，

$3_{(10)} = 11_{(2)} = +2^{+10} \cdot 0.11_{(2)}$。

在 719 机上对阶码 p 的范围限定为

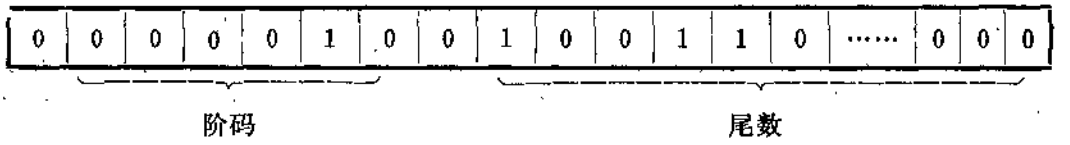
$-1000000_{(2)} \leqslant \pm p < 1000000_{(2)}$，

在 719 机的内存单元中，每个单元有 48 位，每一位可以为 0 或 1 ，二进制浮点数在 719 机的内存单元中的存放形式为



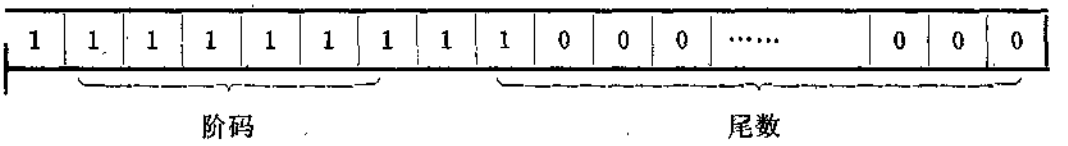
其中数符与阶符均以 0 表示正、1 表示负。尾数 q 的长度为 40 位，阶码 p 的长度为 6 位。另外还规定，当阶符为负时，阶码应改写为二进制的 $1000000 - p$。

例：$2.375_{(10)} = +2^{+10} \cdot 0.10011_{(2)}$ 在内存单元中表示为



$3_{(10)}=2^{+10}\cdot 0.11_{(2)}$在内存单元中表示为

$-0.25_{(10)}=-2^{-1}\cdot 0.1_{(2)}$在内存单元中表示为



其中因为阶符为负，阶码改写为二进制的 1000000－000001。

根据上述表示形式，719 机所能表示的规格化数的范围为

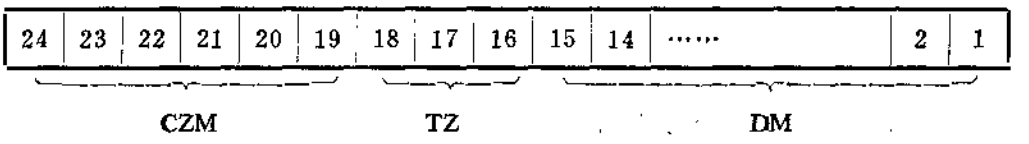
$$2^{-68}\cdot\frac{1}{2}\leqslant|x|\leqslant 2^{68}(1-2^{-40})。$$

若$|x|<2^{-64}$,计算机将它作为零处理，称为“机器零”。机器零一律表示为$+2^{-64}\cdot 0$;

若$|x|>2^{68}(1-2^{-40})\doteq 10^{19}$，则计算机将它作为无穷大处理，遇此情况，将出现溢出停机。

三、内存单元中的指令

电子计算机进行运算操作的过程由执行“指令”来完成的，719 机的每条指令用 8 位的八进制数（即 24 位的二进制数）表示，其形式为：



其中：CZM 称为操作码，占 6 位；TZ 称为特征位，占 3 位；DM 称为地址码，占 15 位。我们举例说明指令的作用，如指令：

400　　15234

这里的操作码 40 表示加法，它将 A 寄存器中的数和地址 15234 中的数相加后仍放在 A 中，特征位 0 表示加法运算后作规格化处理。又如指令

500　　15235

它与 400　　15235 类同。但操作码 40 表示加法，而操作码 50 表示乘法。

我们不准备介绍 719 机的整个指令系统，因为使用算法语言算题时，只须对指令有一个粗糙的了解就可以了。

既然每条指令表示为 24 位的二进制数，所以 719 机的每个内存单元也可以存放二条指令，分别称左指令和右指令。其形式如下所示：

400　15234　500　15235

（左指令）　（右指令）

四、机器语言程序

一个计算公式往往要用若干个指令来实现，一系列的指令根据执行的顺序联成一个整体，称为机器语言程序，简称“程序”。我们从解释下面一个简单例子来说明机器语言程序是怎么一回事。

设 $a=2$，$b=3$，$c=4$，现在要计算 $x=a+b\cdot c$；我们先把公式的有关数据与运算步骤用如下通常的计算表格来表示：

行次	数或运算公式	数　值
①	a	…… 2
②	b	…… 3
③	c	…… 4
④	②×③→④	……12
⑤	④+①→⑤	……14

这个表格中①、②、③行中放置数 a、b、c，而在④、⑤行中列出运算分式，在①、②、③行中填好具体数值后，运算从第④行开始，按其运算公式做完后，再做第⑤行的运算分式，便得运算结果 $x=14$。

对同一个计算问题，编制程序与上述编制表格在形式上是很相近的，我们可以通过解释下列程序看到这一点：

单元地址	数或指令	解释
10001：	0 0 0 0 0 1 0 0 1 0 0 …… 0	……放置数 2，
10002：	0 0 0 0 0 1 0 0 1 0 0 …… 0	……放置数 3，
10003：	0 0 0 0 0 1 1 0 1 0 0 …… 0	……放置数 4，
→ 10004：	700 10002 500 10003 …	左指令：将 10002 中的数取至 A， 右指令：A 中的数乘 10003 中的数，仍放 A。
10005：	400 10001 010 00004 …	左指令：A 中的数加 10001 中的数，仍放 A， 右指令：停机，运算结果显示在 A 中。

程序从 10004 左开始执行指令，该单元地址称为启动地址，执行 10005 右后停机。

了解上述关于计算机内的数、指令、程序的概念，对于我们以下学习算法语言及上机操作，将是有利的。

第三节　什么是算法语言

一、看一个例子

前面，我们从一个简单的例子观察了机器语言程序的形式。编制机器语言程序的工作需要熟悉机器的指令系统，而指令系统与平时我们书写的数学公式差异很大，使用起来很不方便，而且不同的机器有不同的指令系统，这更带来麻烦。

随着计算机及其应用的不断发展，“算法语言”正越来越广泛地被应用，用它来编制程序就比较方便，它与我们习惯的数学公式很相近。仍以上节中的一个简单计算问题为例，如 $a=2$，$b=3$，$c=4$ 时要计算 $x=a+b*c$，，其算法语言程序可写为：

```
begin                                  ……开始
    real  a, b, c, x;                  ……变量说明
    *read  (4, a, b, c);               ……输入 a, b, c 的语句
    x:=a+b*c;                          ……计算 x 的语句
    *print (T2, x)                     ……打印 x 的语句
end     转止                           ……结束
```

这个例子虽然简单，但可以说明算法语言程序是由英文字母、阿拉伯数字和数学记号等其本符号组成，并可看到它的大体结构——包括说明部分、语句部分及前面的 **begin** 和后面的 **end** 等。

二、用算法语言算题的过程

对于生产实际中的数学计算问题，应先构造数学模型及选择计算方法，即用一定的数学公式或计算步骤将所需要计算的量表示出来，然后用算法语言编制程序——称为源程序，习惯上也简称为“程序”。

在编好源程序后，需将源程序及数据按照一定的编码系统在穿孔机上穿成纸带。源程序的穿孔纸带被输入机器后，机器里存有的编译程序（用机器语言编制）对它进行语法检查，并将它译成一个机器语言程序，称为目标程序，此后，执行目标程序，即得计算结果。对于源程序与目标程序的形式，可以从本节及上节提供的例子中有一个粗略的理解。

末了，我们说明一下，719 机的算法语言是以国际算法语言 ALGOL 60 为基础，并吸取其他算法语言方案的某些特点而研制的。本教材介绍的内容是 719 机的算法语言中对数值计算比较基本和常用的部分，同时还介绍上机操作的基本知识。

第四节　算法语言的基本符号

常用的基本符号如下：

一、字母：26个英文字母（大写和小写字母不加区别）

二、数字：0，1，2，3，4，5，6，7，8，9

三、运算符：

（1）算术运算符

+（加），-（减），*（乘），/（除），↑（幂）

（2）关系运算符

<（小于），≤（小于等于），=（等于）

┐≤（大于），┐<（大于等于），┐=（不等于），

（3）逻辑运算符

┐（非），∧（与，∨（或）

（4）顺序运算符

goto（转向），**if**（如果），**then**（则），**else**（否则），

for（对于），**do**（做）。

四、括号

（与）组成一对圆括号

[与] 组成一对方括号

≡与≡组成一对行括号

begin（开始）与 **end**（结束）组成一对语句括号

五、分隔符

Φ（小拾），,（逗号），;（分号），:（冒号），·（小数点），:=（冒等号—赋值号），

#（井号），␣（空白），

step（步长），**until**（直到），**while**（当），**time**（次数），

→（换行），**pp**（换页），

六、说明符：

real（实型），**array**（数组），**proc**（过程，procedure 的缩写），

value（值）。

关于这些常用基本符号，有几点需强调一下：

（1）运算符、括号、分隔符、说明符统称为限制符或定义符，它与字母、数字不同，没有数量值。这些基本符号的含义与用法待以后陆续介绍。

（2）上述符号中有些符号是由英文单词下面加一条横线构成的，它是一个专用的不可分割的基本符号，称为粗体字，719 机的算法语言用前后加二个␣（空白）来表示粗体字，如 **begin** 就表示 ␣begin␣ 的缩写，它与 begin 的含义是不同的，后者的含义见下面第五节。

（3）书写源程序时，只能使用基本符号，其他符号不允许出现，如希腊字母 α,β,γ 是不允许在源程序里出现的，源程序中的 α 可写成 alpha，等等。

第五节　标识符、变量及其说明等

一、数

一般，在计算公式中的少量常数可以直接写在源程序中，而大量数据特别是要变更的数据可以用输入语句在执行程序时从外部输入（见第二章第四节）。

源程序中允许书写的数，在形式上与通常的十进制书写形式相近，如0.53，－28.105，3001，0等。719的机算法语言还允许将数写成带有Φ（小拾）的形式，如0.53Φ9，表示0.53×10^{9}，－5.6Φ－8表示-5.6×10^{-8}，其一般形式为

$$SN\cdot N\Phi SN$$

其中

S——负号或正号，对正号可以省略；

N——无符号整数。

几点注意：

（1）Φ后面的N必须小于18。

（2）不能书写上述形式以外的数，如$\sqrt{2}$，±4.17，$\frac{1}{2}$，-10^{3}，3Φ2.3等。

（3）常用的数π（圆周率）、e（自然对数的底）要写成3.1415926，2.7182818。

（4）数在机器中存放与运算时，整数是完全精确的，一般数可能有误差。

（5）719机可表示的数范围约为$\pm10^{19}$，超过这个范围则溢出停机。

二、标识符（或称为名）

以字母开头的，由字母与数字组成的符号序列称为标识符。

标识符例：

$$a,\ x5,\ c74t,\ yy2,\ \text{alpha},$$

但下列写法不能作为标识符：

$$4y,\ a,\ a+b,\ z\cdot7,\ x[3];$$

标识符是算法语言中一个最基本的概念，用它来命名计算中的各种对象，如本节的下面即要介绍的简单变量与数组，又如以后陆续要介绍的句名与过程等。

719机的算法语言对标识符的可识别长度现规定为8个符号。如function与funclional被认为是同一个标识符。

注意一个标识符只能给一个计算对象命名，一般不能表示二种不同的意义。

三、简单变量及其说明　计算中遇到的单个量称为简单变量。如公式$s=3.1416\times r^{2}$中r与s都是简单变量。

对简单变量，一般应在程序头部加以说明，其形式为

real r, s, ……;

其中r, s等是其相应的标识符。

其后面的分号是作为说明与说明之间，说明与语句之间的分隔符。因次我们在说明之后写上分号，下同。

在对简单变量作说明后，编译程序将分配它们各占一个存贮单元。注意简单变量的值在计算中可以是变化的，当然也可以取固定值。

四、数组及其说明

按一定次序排列的一组数称为数组。如指数函数$y=10^{x}$在$x=-2,-1,0,1,\cdots\cdots9$的数值$y_{-2}$, y_{-1}, y_{0}, y_{1} …… y_{9}就构成一个数组，其中每一元素y_{i}称为下标变量，在算法语言中用$y[i]$来表示，i称为下标。

变量就分简单变量与下标变量二种，对下标变量y[－2]，y[－1]，y[0]，y[1]，……

y[9]，不是一个一个地加以说明，而是对整个数组作出说明。上述数组的说明为

array y[−2:9]

其中 y 称为数组标识符，−2 称为数组的下界，9 称为数组的上界，−2:9 称为界偶表。

一般情形的数组说明为

array $a[\Delta_1]$, $b[\Delta_2]$, ……;

其中 a、b 等为数组所相应的标识符，Δ_1、Δ_2 等为相应的界偶表。每个界偶把下界与上界用冒号连接起来。当然下界应当不大于上界。

如上所述，数学中的有限数列、向量都可以用数组表示，其下标只有一个，我们称为一维数组。在实际计算中还会遇到 2 个下标（2 维数组）及多个下标的情形。如 20 阶代数方程组

$$\sum_{j=1}^{20} a_{i,j} x_j = b_i, \ (i = 1, 2, \cdots\cdots 20),$$

其系数矩阵为 (a_{ij}) $(i, j = 1, 2, \cdots\cdots 20)$。类似地，我们将下标变量 a_{ij} 表示为 $a[i,j]$，而其数组 a 连同数组 b、x 的说明为

array a[1:20, 1:20], b[1:20], x[1:20];

其中数组 a 为二维数组，其界偶表由二对界偶组成，中间用逗号分开。

对于维数与界偶表均相同的若干个数组，其说明可以简化，如上述数组 a, b, x 的说明可简化为

array a[1 : 20, 1 : 20], b, x[1 : 20];

其余类同。

在对数组作说明后，编译程序将分配它们各占一批存贮单元，对 1 维数组，各下标变量的存贮单元按下标顺次安排，对 2 维数组，各下标变量的存贮单元将按行按列顺次安排，如 a[1:3, 1:2] 的排列顺序为 a[1, 1], a[1, 2], a[2, 1], a[2, 2], a[3, 1], a[3, 2]，多维数组的情形类同。

五、标准函数符

在计算公式中，常常会遇到一些初等函数的计算，例如，$\sin x + \cos y + 0.1 - \ln m$。当用算法语言描述这个计写公式时允许把这些初等函数直接写进去，但必须按标准函数符的形式书写，如上式，可算成

$$*\sin(x) + *\cos(y) + 0.1 - *\ln(m).$$

719 机算法语言备有如下一些较常用的标准函数符：

*abs(x)	x 的绝对值；
*entier(x)	不大于 x 的最大整数；
*sqrt(x)	x 的平方根($x \geqslant 0$)；
*cubrt(x)	x 的立方根；
*arcsin(x)	x 的反正弦主值($\lvert x \rvert \leqslant 1$)；
*arctg(x)	x 的反正切主值；
*exp(x)	x 的指数函数(e^x)；
*ln(x)	x 的自然对数；
*sign(x)	x 的符号函数，按 x 为正，0，负时分别取值 1，0，−1；

*sin(x)	x 的正弦（其中 x 以弧度为单位，对 *cos(x)、*tg(x)，亦如此）；
*cos(x)	x 的余弦；
*tg(x)	x 的正切；
*sh(x)	x 的双曲正弦 $\left(\frac{e^x-e^{-x}}{2}\right)$；
*ch(x)	x 的双曲余弦 $\left(\frac{e^x-e^{-x}}{2}\right)$；
*th(x)	x 的双曲正切 $\left(\frac{e^x-e^{-x}}{e^x+e^{-x}}\right)$；
*wg(x)	将 x 按弧度制化成度·分·秒表示形式。如 *wg(0.7145) 为 40.56436，意即为 40°56'43·6''
*gw(x)	将 x 按度·分·秒表示形式化成度制。

第一章 习 题

1. 下列符号是否算标识符？

   3a, 5 mx, mla, tb(3), cyl, yi, sin, b[2], * sqrt, β, xol, bet a.

2. 在程序中是否允许出现下列形式的数？

   10^2, 5.7Φ − 8, 3Φ3.6, − 1.6Φ + 5.

第二章　表达式及若干基本语句

第一节　算术表达式与赋值语句

一、算术表达式通常由数、变量、标准函数符、圆括号及算术运算符所组成，它代表一个有数学含义的算术式，如

算术式	算术表达式
$\frac{a+b}{2}\cdot 10^{-2}$	(a+b)/2*Φ—2
Asin(ωt)	A* *sin(omega*t)
a_i+2b_i	a[i]+2*b[i]
$ln(1+\sin a)$	*ln(1+*sin(a))

从这些例子可以看到，通常的算术式要写成相应的算术表达式是很方便的，只须将其中的数、变量、函数、括号、运算符改写成算法语言规定的基本符号，并写在同一条横线上。注意在程序中不能直接写 $\frac{a}{b}$，3·b，2i－1 等，而要改写成 a/b，3*b，2*i－1 等。同时，应注意标准函数符允许反复套用。

下面对 719 机算法语言规定的算术运算符作一解释：

+、－、*、/ 即普通的加、减、乘、除。

↑ 表示乘幂，幂次可以是任意实数，如 2↑3，4↑1.5 都是允许的，须注意幂次为非整数时，底必须为正数，如不可以出现(－8)↑(1/3)。

二、一个算术表达式中可以多次出现算术运算符，为了避免其含义的混淆，与通常情形相近，对算术运算符的优先顺序（级别）规定为：

（1）先乘幂(↑)，次乘除(*，/)，后加减(+，－)；

（2）优先顺序相同时，自左至右计算；

（3）遇到圆括号时，其中的式子作为一个整体优先计算。

根据上述规定，便可以说明下列算术表达式的含义及其中各个算术运算符执行的次序：

a+b↑2，	(a+b)↑2，
－1+a↑(－3)；	128/2↑3↑2；
a+b/c↑d*e－(f*g+h)；	

注意算术表达式中二个算术运算符不可并置在一起，如不许可写 a+*b，a/－b↑2，后者应改写为 a/(－b)↑2。

三、赋值语句

赋值语句的作用通常是把某算术表达式 E 的值赋给某个变量 v（简单变量或下标变量），其形式为：

$v := E;$

其后面的分号是作为语句与语句之间的分隔符，因次，我们一般在语句之后也写上分号，以下类同。

例1、将一元二次方程 $ax^2+bx+c=0$ 的求根公式写成赋值语句，为

$x1 := (-b+*\mathrm{sqrt}(b\uparrow 2-4*a*c))/2/a;$

$x2 := -b/a-x1;$

其中 $x1$、$x2$ 表示二个根，要求 $a\neq 0, b^2-4ac\geqslant 0$。注意上述二个语句的次序不能简单地交换，因为语句一般是按顺序执行的。

例2、上述求根公式亦可写成如下赋值语句：

$b := b/a;$

$c := c/a;$

$x1 := -b/2+*\mathrm{sqrt}(b\uparrow 2/4-c);$

$x2 := -b-x1;$

前二个赋值语句相当于将方程中二次项的系数化为 1，其新的一次项系数与常数项恰为简单变量 b, c 的新值，此例说明赋值号“:=”与数学公式中一般的等号“=”含义是不同的，赋值号是有方向性的等号，它将右边算术表达式的值赋给左边的变量，因此，变量的值在计算中是可以改变的。

例3、$x[1:10]$为数组，i、j，为简单变量，说明执行下列语句后各变量的值：

$i := 1;$ $\qquad$ $j := i+1;$

$x[i] := 1;$ $\qquad$ $x[j] := 2*x[i]+j;$

$x[i+j] := x[i]+x[j];$

$i := j+1;$ $\qquad$ $j := i+1;$

$x[2*i+j] := x[i]+x[2*x[j-3]];$

逐个分析各个语句的执行情况，可知在执行这些语句后，$i=3$，$j=4$，$x_1=1$，$x_2=4$，$x_3=5$，$x_{10}=9$，而其余 x_4，x_5，……x_9 未予赋值（其值一般为不确定）。

该例还说明，下标变量的下标也可以是算术表达式，当然必须取整整值。

第二节　逻辑表达式与条件语句

一、关系式

若干个算术表达式用关系运算符联结起来的式子称为关系式。

关系运算符共 6 个：

$<$，$\leqslant$，$=$，$\neg\leqslant$，$\neg<$，$\neg=$。

例：

$i\leqslant 100$，$*\mathrm{abs}(x)*2+1\neg<2*y/3$，

$x[5]+3\leqslant a$，$\qquad$ $0\leqslant x\leqslant 4$，

$1/4<x\uparrow 2+y\uparrow 2<1$。

注意 719 算法语言允许一个关系式中关系运算符可以多次出现。

从这些例子中可以看到，算法语言中的关系式相当于数学上不等式的概念。关系式的成立与否，我们用 **true**（真）或 **false**（假）来表示，关于 **true** 与 **false** 作为基本符号的用法在数值计算中一般不常用，本讲义不介绍，我们只用它在讲义中作为叙述内容时使用。

一般地说，一个关系式有 **true** 与 **false** 二种可能性，如 (x, y) 表示平面直角坐标系中点的坐标，关系式 $x\uparrow 2+y\uparrow 2<1$，当该点在以原点为圆心的单位圆内时为 **true**，当点在圆周上及圆外时，该关系式为 **false**。

当然，也有某些关系式恒为 **true** 或恒为 **false**，如 $x\uparrow 2+y\uparrow 2\neg<0$ 恒为 **true**，而关系式 $3=1$ 恒为 **false**。

还须注意，关系运算符 =，¬= 应谨慎使用，如 $a=5.1$，$b=2$，$c=10.2$ 时，关系式 $a*b=c$ 似乎应为 **true**，然而，由于机器在计算中有舍入误差，结果却成为 **false**，这样就搞错了。一般当两边的算术表达式均取整数值时，用 = 、¬= 是不会产生错误的。

二、逻辑表达式

若干个关系式用逻辑运算符与括号联结起来的式子称为逻辑表达式。逻辑运算符常用有三个：

¬（非），∧（与），∨（或）。

逻辑表达式的成立与否仍用 **true** 与 **fals** 来表示。逻辑表达式是关系式的扩充，而关系式是逻辑表达式的特例。

如 $-2<x<2\wedge -2<y<2\wedge x\uparrow 2+y\uparrow 2\neg\leqslant 1$ 对坐标平面上位于单位圆外面在一正方形内的点为 **true**。

在规定 i 取整数值时，$i\neg\leqslant 5\vee(i\neg\leqslant -3\wedge i\leqslant 0)$ 为 **true** 的范围是 $i=6, 7, 8, 9\cdots\cdots$ 及 $i=-2, -1, 0$。在其他范围为 **false**。

三个逻辑运算符的优先顺序为：

（1）¬，（2）∧（3）∨.

其他规则与算术运算符的情况类同，如括号处理等。

另外，对逻辑表达式中出现的不同运算符，以算术运算符在先，关系运算符其次，逻辑运算符在后。

三、条件语句：

条件语句有二种形式：

（1）**if**—**then** 语句，其一般形式为：

if *LE* **then** S1；

（2）**if**—**then**—**else** 语句，其一般形式为：

if *LE* **then** S1 **else** S2；

其中 *LE* 为逻辑表达式，S1、S2 是语句——可以是我们已学过的赋值语句，也可以是我们以后学到的其他语句。

条件语句的执行效果可用下列框图来表示：