6912机算法语言介绍

北京大学汉中分校计算中心

前　言

本讲义是6912型电子计算机配备的算法语言的一个导引，可供初学编写6912机语言程序的同志参考。这是一本普及性的讲义，使用它不需要初学者有在计算机上工作的经验。

本讲义介绍了6912机算法语言的基本内容。掌握了这些基本内容，便能够了解语言程序的梗概并且可以着手编写一些实际的算题程序。对这些基本内容力求叙述得通俗易懂，便于自学。本讲义未介绍如何上机操作、如何调整程序。为了突出重点，也没介绍算法语言中的带语句、格式语句以及转向语句中的计算转、符合转等内容，并且把其中一些非本质性的细节规定略去了，如数据语句的换行规则及连续若干个相同数据的缩写形式。对计算机和程序设计已经有所了解的同志可以不看第一章。

编写本讲义的主要依据是北京大学电子仪器厂出版的《6912型电子计算机使用说明书》。此外，还参考了北京有线电厂译的《程序语言 FORTRAN》、科学院数学所周龙骧发表的《算法语言ALGOL60导引》、北京619信箱黄仲孚编写的《DJS—6机算法语言介绍》、清华大学电子系计算数学教研组编写的《112电子计算机初学者会话语言》等有关资料。

由于水平所限，本讲义的问题和错误一定很多。特别是由于当前条件所限，讲义中所举的一些例子未能上机调试，更有待实践的检验。衷心希望使用本讲义的同志给予批评和指正。

北京大学汉中分校计算中心

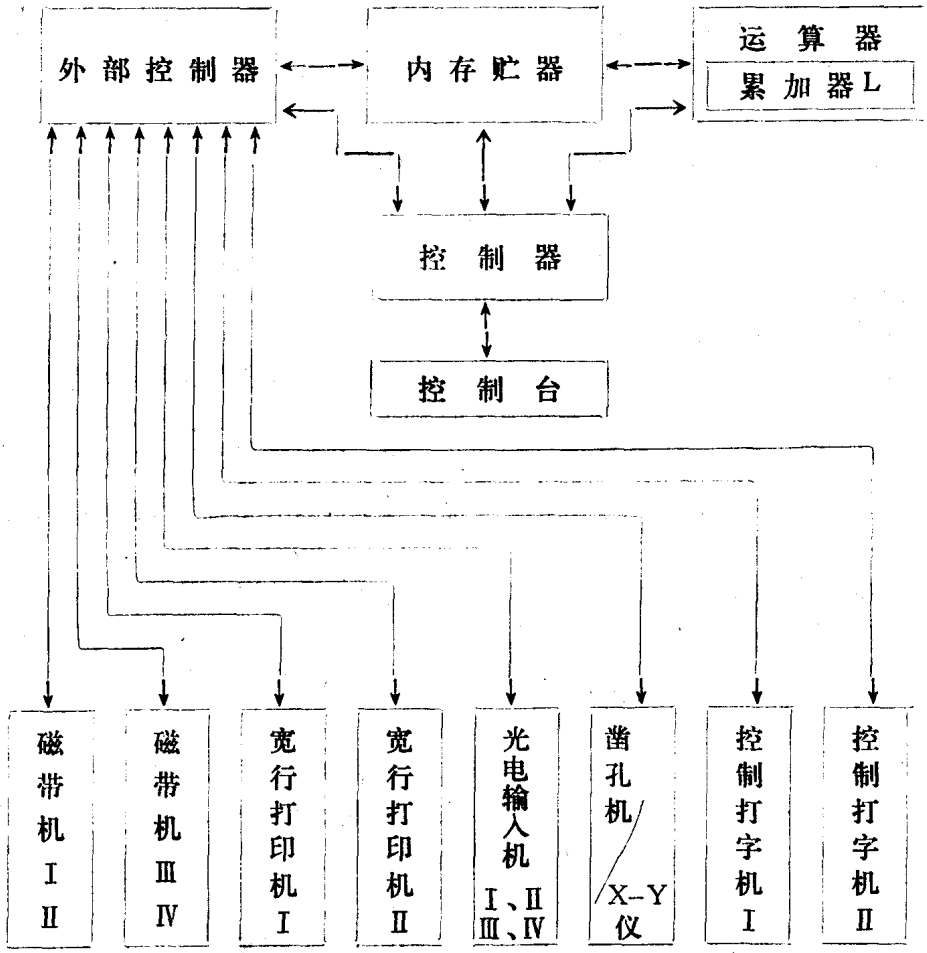
1974年10月

目　录

第一章 基础知识

第一节 6912机的基本结构

6912型电子计算机是一台集成电路中型通用数字机。速度约为每秒10至15万次。内存贮器容量为65536个全字长单元。外部控制器有8个通道，每个通道可接4台外部设备。它的基本部件可分为内存贮器、运算器、控制器、外部控制器、输入输出设备和磁带机等，各基本部件之间的联系可用下面的框图表示：



这些部件各有什么作用呢？为了便于理解，我们先来分析一下人是怎样用算盘进行计算的。首先要把计算公式、计算步骤、初始数据写在纸上，然后按次序用算盘进行加、减、乘、除这些基本的算术运算，必要时把计算所得到的中间结果和最后结果写在纸上，在这个计算过程中用到了纸、笔、算盘、脑和手。计算机开始工作以后，它的内存就相当于纸笔，运算器就相当于算盘，控制器就相当于脑和手。

内存贮器是用来存放初始数据、程序及中间结果的。内存分为两个体，每个体有32768个全字长单元。一个全字长单元占48位，即可存贮48个二进制代码。这些单元都统一编号，这种编号就叫单元的地址。内存具有这样的特点，即从其任一单元中读出代码并不改变该单元的内容，仍可再次读出，但是往某一单元中送入代码，该单元原来存放的代码就被挤掉了。

运算器是用来对数据或代码进行加、减、乘、除等算术运算及逻辑运算的。累加器L是其最主要的一个部件，它可以存放从内存取出的数据及运算结果。

控制器能不断发出信号，以根据人编的程序有条不紊地控制、执行一系列的动作，指挥全机各部件协调工作。控制器附设有控制台。除个别按键外，6912机的控制台不提供给算题人员使用。

6912机采用了运算器与控制器合一的设计，称为运算控制器。

以上三个部件，合起来叫做计算机的主机。

主机还要人来控制，人和主机必须有交换信息的渠道，这就需要给计算机配备外部设备。目前6912机的外部设备有：

光电输入机是将人用穿孔机穿在纸带上的数据和程序输入主机的设备，共有4台，占一个通道。

宽行打印机可把计算结果打印出来给人看，是主要的输出设备，共有2台，各占1个通道。

凿孔输出机可把数据或程序穿在纸带上，供以后输入用。

X—Y绘图仪也是一种输出设备，可将计算结果直接画出曲线。

凿孔输出机和X—Y绘图仪各有一台，它们合占一个通道。

控制打字机是人与计算机联系的主要工具。算题人员通过打字机可向主机输入命令，控制主机及外部设备的工作。当在运算过程中发生问题时，主机也能通过打字机输出信息，把发生的问题及时告诉算题人员，以便算题人员进行分析、处理。所以，控制打字机既能输入，也能输出，但这些信息都是控制性质的，不能用打字机大量地输入或输出。共有2台，各占一个通道。

磁带机是一种可装卸的外存贮器，与内存一样，也是存贮数据或程序的，之所以称它为外存，是因为它不能象内存那样可以直接与运算器、控制器联系，而只能通过外部控制器与内存交换信息，它也是既能输入也能输出的。共有4台，两台合占1个通道，共占2个通道。

所有这些外部设备与主机交换信息都要在外部控制器的统一控制之下进行。有了外部控制器，再与控制器中的中断系统相配合，便可以实现外部设备与主机并行工作以及若干外部设备之间的并行工作，从而提高整个机器的工作效率。

第二节　用计算机解题的基本步骤

使用电子计算机解题，大致可分为如下五个步骤。

一、建立数学模型。这是指把生产和科学实验中需要解决的问题用数学形式描述出来。

二、确定计算方法。数学问题是多种多样的，可以是求解代数方程，也可以是求积分或求解微分方程等等。必须针对不同的情况，采用相应的数值计算方法，把这些问题归结为计

算机能够执行的四则运算及逻辑运算。

三、编制程序。这是指把第二步确定的计算过程用计算机能懂的语言描述出来。

四、上机计算。这包括上机前的准备工作（如穿孔、拟定操作说明书等）、上机调程序与试算直至算出最后结果。

五、分析结果，回到实践中去检验。

当然，这五步也不是截然分开的，它们是相互影响、相互制约的。如在建立数学模型时，就应该考虑到有没有合适的计算方法，确定计算方法也要考虑到如何使编程序来得容易。结果算出来了，如果不能满足需要，则要考虑如何改进计算方法，甚至修改数学模型。

本讲义只集中讨论第三步，即如何编程序的问题。

第三节 程序的概念

电子计算机进行运算和逻辑判断的速度很快，但如果每执行一步以后，就要人来指示下一步的操作及数据，如同使用手摇计算机或电动计算机那样，那么，电子计算机速度再快也是没有意义的，为了适应电子计算机的高速度这一特点，使用它时，必须编排好计算顺序，并把这种计算顺序连同所用的数据预先输入到计算机的内存中去，计算机开始工作以后，控制器便能按照规定的顺序从内存取出数据让运算器执行指定的操作，在这一段时间内计算机相对地摆脱了人的干预，便能实现高速自动地运算。这种预先编排好的计算顺序称谓“程序”。编排程序的工作称谓“程序设计”。

编程序应该使用计算机懂得的语言，每一台计算机都有一套指令系统。每一条指令一般由操作码和地址两部分组成。操作码指明作什么样的操作，如加、减、乘、除、取数到累加器 L、从 L 送数到内存等等。地址所指明的那个单元中的数据即是被操作的对象。如要计算

$$13\times 22.5+94.7,$$

我们假定存放数据13、22.5、94.7的单元的地址分别是 A、B、C，存放结果的单元的地址是 D，我们可以用机器指令来编程序：

$\rightarrow L$ A；　表示把 A 中的数13取到 L 中

$*$ B；　表示数13与 B 中的数22.5相乘

$+$ C；　表示上次相乘的结果再加 C 中的94.7

$L\rightarrow$ D；表示将所得结果送到单元 D

但上面那些表示操作码的符号计算机并不认识，需要把它们换成相应的代码，$\rightarrow L$ 的代码是06，$*$ 的代码是14，$+$ 的代码是10，$L\rightarrow$ 的代码是42。地址也要用数码表示，我们可以认为 A 是1000，B 是1002，C 是1004，D 是1006。这样一来，我们可把上面的程序“代真”为

400:060001000,

401:140001002,

402:100001004,

403:420001006,

冒号前的数码也是地址，冒号后的指令就放在这个单元中。这样的程序，计算机就认识了，另外，还要把常用的十进制数据化成机器能认识的二进制形式。如果我们用穿孔机穿好纸带，

用光电机输入机器，就能让计算机进行计算了。

通过编这样的一个小程序，我们就可以发现用机器语言即指令来编程序是很麻烦的。首先，编程序的人必须较多地懂得有关计算机本身的知识，至少要透彻了解其指令系统；第二，这样的程序很不直观，一般人很难看懂，不用说看别人的程序如啃“天书”，就是自己编的程序隔了一段时间，往往也记不清楚了；第三，这样的程序有错难查，查出来要修改也不方便；第四，这样的程序还缺乏通用性，不同机器的指令系统是不同的，用6912机的指令编的程序，其它机器是不认识的，反之也是如此。

随着计算机的不断发展，必须改变用机器语言来编程序的落后状况。否则，编程序的专业人员不足，工作效率又低，将大大影响计算机的推广使用。在实践的基础上，程序设计技术也是不断发展的，现在人们通常都是用“算法语言”来编程序了。

第四节 算 法 语 言

如果计算机能直接懂得人们习惯使用的数学语言，那是最理想的了。但由于技术上的限制，还没有做到这一步。目前广泛使用的是“算法语言”一类的程序设计语言。

算法语言是独立于具体的计算机的，能够精确描述数值计算的过程。使用算法语言来编程序，不要求对机器本身有多少了解，写出来的程序比较直观，接近通常的数学公式，一般人容易看懂，便于交流，通用性也强。编程序时出错的可能性减少了，有错也容易查找和修改。例如，上节的例子用算法语言就可以直接了当地写成

$$D = 13 * 22.5 + 94.7;$$

但是，计算机并不懂得算法语言。这需要人给它配备一个“翻译”，即用机器语言编一套“编译程序”。这套编译程序事先就存放到机器中去，它懂得用算法语言写的程序（一般称谓“源程序”）。在计算机上实际执行算法语言程序，一般要分两步走。

第一步先由编译程序把源程序翻译成用机器指令组成的程序（一般称谓“结果程序”或“目标程序”），这一步叫做编译阶段。

第二步再执行结果程序，对初始数据进行加工，算出结果。这一步叫做运行阶段。运行时，还需要有一套辅助的程序（一般称谓“运行程序”）来帮助执行结果程序。

第二章　简单程序

第一节　基本概念

一、基本符号和定义符

正象英语是由26个字母和若干个标点符号构成的一样，6912计算机的算法语言（以下简称本语言）也是由64个基本符号构成。今后用本语言编写的程序只允许出现这些基本符号，除此以外的任何符号计算机都不认识。这些基本符号是：

1. 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

2. 数字

0 1 2 3 4 5 6 7 8 9

3. 算术运算符

+（加）　-（减）　*（乘）　/（除）　↑（乘幂）

4. 逻辑运算符

¬（非）　∧（与）　∨（或）

5. 基本关系符

＜（小于）　＞（大于）　＝（等于）

6. 分隔符

10（小十）　·（点）　，（逗号）　；（分号）　：（冒号）

7. 括号

（　）　［　］

8. 其他

§　*　⇐　▽　←　→　？　△　（空格）。

使用这些基本符号需要注意以下几点：

1. 字母一律用大写的印刷体。这样在穿孔时容易辨认，少出差错，同时与宽行打印机及控制打字机的输出一致，便于查对。

2. 有些基本符号在手写时容易混淆，如字母O与数字0，字母I、数字1与除号/，字母V与逻辑运算符号∨等。因此，一定要求编程序的同志书写工整。对字母O，还可以在中间加一小点，以与数字0相区别。

3. 基本符号中的最后一个是空格。空格就是指在一行中空出一个位置不写任何符号，手写程序时，为了醒目起见，有时用△代替。

除上述64个基本符号外，本语言还要用到如下28个定义符：

*ZU	*ZC	*HD	*JB
*SZ	*SG	*KG	*HS
*DG	*DA	*ZX	*FX
*RU	*ZE	*FZ	*XH
*DY	*FH	*SS	*ZZ
*ZG	*CL	*SJ	*SA
*GA	*GW	*MM	*WA

关于这些定义符的意义及用法以后将分别介绍。在这里需强调指出：　组成定义符的3个符号（*及两个字母）是一个统一体，不可分割，在语言中的地位如同上面的64个基本符号完全一样。

二．数与变量

任何一个程序总是离不了数和变量。对于这两个词，我们按数学上通常的意义去理解。在本语言中，数用十进制形式书写，如：

1974　　+653　　−207　　0

0.618　　+3.14159265　　−2.718　　.53

$0.45_{10}-7$　　$-12.7_{10}14$　　$_{10}+5$　　$-_{10}+6$

以下三点需要提及：

1．正数前面的“+”号、正阶码前面的“+”号、无意义的“0”写与不写皆可，如：

+0.5　　0.5　　0.50　　.5都相等，

$0.1_{10}5$与$+0.1_{10}+5$也是一样的，

但不允许小数点后没有数字，象51.和$3._{10}-3$就是错误的写法，应改写为51.0和$3.0_{10}-3$。

2．数的阶码一定是整数。

3．数的绝对值不得超过$_{10}38$，数的有效数字最多可以有12位。

变量是指在程序中即计算过程中可以取不同的值的量。类似于数学上用字母或字母加数字表示变量，本语言也要给用到的变量起个名字。如计算二次多项式

$$M=AX^2+BX+C$$

的值，当系数A、B、C一定时，X、M是变量。当X取不同的值时，M也取不同的值。X、M也就是变量的名字。

为了与以后要介绍的数组中的元素——下标变量相区别，我们称这些单个的（或者说，具有单个值的）变量叫简单变量（简变）。

三．程序段与语句

一篇文章可以分成若干段。每段有很多句子。用本语言写的程序也是如此。一个完整的程序由若干个程序段构成，其中必须包含且只能包含一个主程序段。每个程序段由若干语句组成。语句是独立的运算单位。每个语句之间一律用“；”隔开。

四．名字

前面已经提到，简变要有名字。同样，程序段也要有名字。有些语句前面要加上标号，

这些标号也是名字。以后要介绍的数组也要有名字。总之，凡程序中出现的量都要有名字。

凡名字都由1个到4个字母和数字组成，第一个符号必须是字母，例如：A，X_1，Y_2，MA5T，A5B3，A_{122}都是名字。而3A，A]，+B，C3-等就不是名字。如果超过了4个符号，只识别前4个，如ABCDE将认为是ABCD。

第二节　算术表达式

通常的算术表达式是语言程序的核心部分，程序的任务一般是离不开计算表达式的值的。

简单的算术表达式是由数和变量通过加、减、乘、除、乘幂和园括号组合而成，例如：

$$AX^3+BX^2-7X+0.5$$

$$\frac{(a+b)\ c^{1.5}-d[e+a]}{b_2}+c_{13}$$

$$d-\frac{a^3+b\ (c^2-d)}{e\cdot f}+2n$$

都是算术表达式。不过，要把这样的表达式送到计算机中去进行计算，还必须作一些变化：

1. 6912机用光电机作为输入设备，它只能把排成一串穿在纸带上的符号一个接一个地送入机器，并且机器（编译程序）分析表达式也是一个接一个地取符号来进行的。所以，首先要把这样的表达式写在一行中。为此，分式$\frac{A}{B}$应改写成A/B、乘幂X^3应改写成X↑3、$(X^y)^z$应改写为X↑Y↑Z。

2. 乘号用*以避免与字母X相混。也不能用基本符号中的“·”代替，“·”只用作表示数中的小数点，别无他用。要注意的是，乘号*不能省略。如2N应写成2*N，A*B不能写成AB。原因是明显的：如果A*B写成AB，则机器（编译程序）认为它是一个名字为AB的变量，而不认为它是变量A与变量B的乘积了，而对2N，机器（编译程序）则认为它是错误的，因为它既不是数，又不符合关于名字的规定。同样，AX^3应写成A*X↑3而不能写成AX↑3。

3. 因为基本符号中没有花括号而方括号另有用处，因此表达式中的括号一律用园括号，而不管套几重括号。括号必须成对出现。

按照这几条规定，上面的三个表达式应写成如下形式：

```
A*X↑3+B*X↑2-7*X+0.5
((A+B)*C↑1.5-D*(E+A))/B2+C13
D-(A↑3+B*(C↑2-D))/(E*F)+2*N
```

表达式的运算有一套优先规则。它们是从左往右算，先乘除后加减，乘幂比乘除优先，如果有括号，则括号中的先算，对于多重括号，则从里往外一层层地进行。

例：　　B+C-D*E/F↑G

其计算次序是

1. B+C　　体现由左往右
2. D*E　　体现*比-优先及由左往右

3. F↑G　　体现↑比/优先

4. D＊E/F↑G　　体现/比－优先

5. 最后得B＋C－D＊E/F↑G

又例如：（A＋B）＊（(C－D)＊E＋F/G)

其计算次序是

1. A＋B　　体现括号中先算及由左往右

2. C－D　　体现内层括号先算

3. (C－D)＊E　　体现由左往右

4. F/G　　体现/比＋优先

5. (C－D)＊E＋F/G　　体现括号内优先

6. 最后得（A＋B）＊（(C－D)＊E＋F/G)

算术表达式是计算数值的规则，它的结果也总是一个数值。表达式中的变量在实际运算进行之前必须具有确定的值。

最简单的表达式是一个常数或变量，如：5，＋3.7，A，－B，＋C13都是表达式。但要注意，表达式中不能允许两个运算符直接相连，例如把A↑（－2），D/（－G），A＋（＋3）写成A↑－2，D/－G，A＋＋3都是错误的。

第三节　标　准　函　数

在表达式中参加运算的除了数和简变外，还可以有函数。

例如求二次多项式

$$AX^2+BX+C$$

的实根，则要计算如下两个算术表达式：

$$\frac{-B+\sqrt{B^2-4AC}}{2A} \tag{1}$$

$$\frac{-B-\sqrt{B^2-4AC}}{2A} \tag{2}$$

按照第二节中关于表达式写法的规定，我们可把（1）改写成如下形式（以下以（1）为例进行分析，（2）可以照套）：

$$(-B+\sqrt{\ B\uparrow 2-4*A*C\ })\ /\ (2*A) \tag{3}$$

但（3）中开平方的符号在基本符号中是没有的，应进一步处理。数X开平方就是求函数$\sqrt{X}$的值，而这个函数计算机已预先配备好了，并记作§PFG(X)。因此，表达式（3）可进一步写成

$$(-B+\S PFG(B\uparrow 2-4*A*C))\ /(2*A)$$

除求平方根外，还配备了一些其他常用的函数，它们都有专门的记号（特殊的名字），在表达式中要用到时只要写出它们的名字和自变量就可以了，无需作任何说明。这些函数叫做标准函数，下面列出它们的名字。

1. §PFG　　平方根

2. §LFG　　立方根

3. §SIN 正弦
4. §COS 余弦
5. §TG 正切
6. §ASIN 反正弦
7. §ATG 反正切
8. §EXP 指数
9. §LN 对数
10. §ABS 取绝对值
11. §SIGN 取符号
12. §FQZ 取整数部分
13. §FQF 取分数部分

引用这些标准函数，需注意以下几点：

1. 这些标准函数的名字都是由§及若干个字母组成，这些字母有的是函数名称的汉语拼音缩写（如§PFG，§LFG等），有的是通常的数学符号（如§SIN，§LN等）。与定义符一样，构成标准函数名字的§及后面字母是一个整体，不可分割。

2. 这些标准函数可以出现在算术表达式中变量所能出现的任何地方，并且可以把它作为一个变量看待，它的值应首先被计算出来。

3. 这些函数的自变量可以是数，简变，也可以是一个算术表达式。例：

```
§SIN (0.35)
§EXP (A)
§PFG (B↑2-4*A*C)
```

4. 不管是数，简变还是一个复杂的表达式，只要它作为一个标准函数的自变量，则一定要用园括号把它们括起来，否则象§SIN0.35，§cosA，§PFG8都是错误的。

5. 既然标准函数的自变量可以是表达式，那在此表达式中当然也可以出现标准函数，即允许标准函数相套。如：

```
§SIN (§SIN(X))
§LN (§ABS ((-B+§PFG(B↑2-4*A*C))/(2*A)))
```

6. 这些标准函数的意思是明白的，只作如下说明（以下自变量E均作表达式看待）：

§SIN (E) 等三角函数的E值均为弧度值；

$$\S\mathrm{SIGN}\ (E) = \begin{cases} +1 & \text{当}E>0, \\ 0 & \text{当}E=0, \\ -1 & \text{当}E<0, \end{cases}$$

§FQZ (E) 为不超过E值的整数部分，如：

```
§FQZ (3.2) =3,      §FQZ (4.9) =4,
§FQZ (-2.1) =-3,    §FQZ (-9.8) =-10,
§FQZ (1 2) =12;
```

§FQF(E) 为E的值的分数部分，恒为正。如：

```
§FQF(3.2)=0.2,      §FQF (4.9) =0.9,
§FQF (-2.1) =0,9,   §FQF (-9.8) =0.2,
```

§FQF (12) = 0；

所以，恒有§FQZ (E) + §FQF (E) = E。

第四节 计 算 语 句

计算语句是算法语言各种语句中最基本的。在数值计算中，常常需要把一个表达式的计算结果作为一个变量的值，这就要用到计算语句。计算语句的一般形式是：

V = E；

其中左部V是一个变量的名字，等号右边的E是一个算术表达式。当计算机执行此计算语句时，首先计算出E的值，然后把此值赋给V，从而代替V原来的值。例如：

X = 50； 表示将数值50赋给简变X；

I = I + 1； 表示将变量I原有的值加1再重新赋给I；

E = - E； 表示将简变E的值改变符号再赋给E；

F = (375 + §SIN(X)) / (A + B)； 表示将等号右边的表达式计算出的值赋给简变F。

需要注意到：

1. 计算语句中的等号与通常数学中的等号，意义有所不同。 I = I + 1；与E = - E；在数学上是不成立的，而作为计算语句却是正确的，有着明确的意义。计算语句中的=是有方向性的，它的左右是不对称的，它是一种操作而不代表一个简单的相等关系。

2. 出现在计算语句右边的算术表达式中的变量必须预先赋值。计算语句被执行后，这些变量一般说来都保持原值不变，除了又出现在左部的以外。如

Z = X + Y + Z + 1；

在执行前，X，Y，Z都应具有确定的值。执行后，Z的值变了，X，Y的值都不变。

3. 以上所举的计算语句的左部都是简变，本语言规定左部还可以是下标变量和函数段名（这将在后面介绍），但不能又是一个其他的表达式。如：

- A = A；

A + 1 = A；

A + B + C = X + Y + 1；

都是错误的。

4. 把一个表达式E的值算出来赋给变量V，这有着“缩写”的意义，当一个表达式很庞大复杂而在程序中又要多次用到它时，我们可以用计算语句把它的值赋给变量V，以后每次再用此表达式的值时就不必计算而只需引用V就可以了。这种“缩写”的功能是计算语句的重要功能之一，它不仅可以使程序大大简化，更重要的是减少机器的运行时间。

5. 如果要把表达式E的值同时赋给几个变量，要一个语句一个语句地写。如

X = E； Y = E； Z = E；

若连写成 X = Y = Z = E；则是错误的。

第五节 简 单 的 程 序 示 例

有了计算语句，我们就可以来编一些简单的程序。

例1. 求二次方程 $AX^2+BX+C=0$ 当A=2，B=-18，C=28时的两个实根。

我们不难写出称之为语句串的如下一系列计算语句。

```
A=2;
B=-18;
C=28;
X1=(-B+§PFG(B↑2-4*A*C))/(2*A);
X2=(-B-§PFG(B↑2-4*A*C))/(2*A);
```

但这不算是一个完整的程序。在第一节中我们就指出：用本语言写的程序是分段的，它必须包含而且只能包含一个主程序段。因此，我们必须把上面的语句串包括到一个主程序段中去即要在其前面加上“主程序段首部”在其后面加上“末”。“主程序段首部”是指定义符*ZU及主程序段名，若我们给这个主程序段取名 QG，则首部为 *ZU QG；“末”即是定义符*MM。

另外，在整个程序最后还要加上 *WA；表示“完”。

这样，我们便可以写出一个程序如下：

```
*ZU QG;
  A=2;
  B=-18;
  C=28;
  X1=(-B+§PFG(B↑2-4*A*C))/(2*A);
  X2=(-B-§PFG(B↑2-4*A*C))/(2*A);
*MM;
*WA;
```

应该注意到，*ZU 与 *MM在程序中必须成对出现，它们及由它们括起来的语句串组成主程序段。

例2. 计算复数X+Yj和0.6+0.8j（当X=5/13，Y=12/13时）的乘积。

我们编出程序如下：

```
*ZU FCJ;
    X=5/13;      Y=12/13;
    U=0.6*X-0.8*Y;
    Y=0.8*X+0.6*Y;
    X=U;
*MM;
*WA;
```

这里，X、Y是简变，开始分别具有确定的值 5/13、12/13。我们的目标是把乘积的实部、虚部又赋给它们。但由第一个求实部的表达式计算出来的值不能直接赋给X，否则求虚部时便得不出正确结果。为此我们引进了一个起辅助作用的中间变量U以暂存乘积的实部。适当地引进中间变量是程序设计的技巧之一。

如果把上面两个程序用穿孔机穿成纸带，便可以装入计算机，让机器执行，算出结果。

第六节 如何输出计算结果

由计算机得到的中间结果或最后结果，经常需要通过宽行打印机打印出来以告诉算题人员或者作为计算成果保留起来。如第五节例1的计算结果保存在简变 X_1、X_2 中，要把它们打印出来，可用下面两个语句来实现：

```
*DY  §KHS (0, 40) X1;
*DY  §KHS (0, 40) X2;
```

也可以用一个语句来实现：

```
*DY  §KHS (0, 40) X1, X2;
```

显然，这些语句应放在算出 X_1、X_2 的值的计算语句的后面，*MM；的前面。

在前面加了定义符 *DY 的语句称为调用语句，以后还要详细介绍。§KHS (0, 40) 表示用哪一台宽行输出及打印的格式，对它的详细剖析也留到以后进行。紧接着园括号应写打印对象，可以只写一个，也可以连写几个。

程序中增加了上述的语句，宽行打印机就会打印出

$$X_1 = \pm 0.SS\cdots S_{10} \pm JJ$$

$$X_2 = \pm 0.SS\cdots S_{10} \pm JJ$$

其中尾数有12位数字，阶码有2位数字。

第七节 暂挂语句与撤离语句

需要计算机做的工作（如运算，打印结果等）全部完成以后，应该让机器停止执行本程序，否则计算机将会盲目地继续运行下去直至造成出错为止。使用暂挂语句与撤离语句，就可使程序停止执行。

暂挂语句的形式是

```
*ZG〈名字〉;
```

其中定义符 *ZG 是“暂挂”的汉语拼音缩写，后面跟的名字可以随便取。当执行此语句时，控制打字机输出信息（包括该语句所在的程序段名及所带的名字）告诉算题人员程序已不再继续运行（简称“挂起”），等待处理。

撤离语句的形式是

```
*CL;
```

定义符 *CL 是“撤离”的汉语拼音缩写。当执行此语句时，控制打字机也输出信息，告诉算题人员程序已经从内存中抹去，称之为“撤离”。

暂挂语句与撤离语句虽然都能使程序停止执行，但也有区别。执行暂挂语句后，程序仍在内存中，可由算题人员命令继续运行或从任何一个合理的地方开始运行（如从头重做一遍），而执行撤离语句就不能这样了，要想重做只有把程序重新装入机器才行。

第五节所写的两个程序还有欠缺之处，现在把第六节、第七节介绍的语句补充进去。

求二次方程的两个实根的程序：

```
*ZU  QG;
     A=2; B=-18; C=28;
     X1= (-B+§PFG(B↑2-4*A*C))/(2*A);
     *DY  §KHS (0, 40) X1;
     X2= (-B-§PFG(B↑2-4*A*C))/(2*A);
     *DY  §KHS (0, 40) X2;
     *ZG  ABCD;
*MM;
*WA;
```

求复数乘积的程序：

```
*ZU  FCJ;
     X=5/13; Y=12/13;
     U=0.6*X-0.8*Y;
     Y=0.8*X+0.6*Y;
     X=U;
     *DY  §KHS (0, 40) X, Y;
     *CL;
*MM;
*WA;
```

第三章　分支程序

第一节　语句标号与转向语句

第二章的例子表明，计算机执行程序是根据语句在程序段中出现先后的自然顺序一个接一个地做下去的，但有时也需要改变这种自然顺序，从某个语句向下或向上跳过几个语句去执行另一个语句。这就需要给语句带上标号并使用转向语句。

第二章第一节已指出，标号也是名字，它应遵守有关规定。标号应放在语句的前面，并用冒号与语句隔开。如：

```
        N = 0;
        SUM = 0;
BH1:    N = N + 1;
        SUM = SUM + N↑2;
```

其中第三个语句就带有标号 BH_1。标号只起标记作用，语句带不带标号，其执行效果完全一样，任何一个语句可以带标号，也可以不带，看需要而定。

转向语句的形式是

```
        *ZX  L;
```

*ZX 是“转向”的汉语拼音缩写，L 是同一程序段内某个语句的标号，执行 *ZX L; 就是转去执行标号为 L 的语句。如在上例后面加上一个转向语句，得到：

```
        N = 0;
        SUM = 0;
BH1:    N = N + 1;
        SUM = SUM + N↑2;
        *ZX  BH1;
```

当执行 *ZX BH1; 时就是又转去执行第三个语句，往下又按自然顺序继续执行，直到再次碰到转向语句。不难看出，执行上面语句串的结果是在简变 SUM 中形成自然数的平方和

$$1^2 + 2^2 + \cdots\cdots$$

并循环不已。

利用语句标号与转向语句我们可以把求二次方程的实根的程序与求复数乘积的程序合并在一个程序中，由此进一步体会它们在程序中的作用。

```
        *ZU  HB;
             A = 2; B = -18; C = 28;
             X1 = (-B + §PFG(B↑2 - 4 * A * C)) /(2 * A);
             X2 = (-B - §PFG(B↑2 - 4 * A * C)) /(2 * A);
             *ZX  SC;
QD2:    X1 = 5/13;      X2 = 12/13;
```