内 容 简 介

本书共分三部分．第一部分是一般性介绍，通俗地说明什么是计算机，什么是程序设计，又什么是编译程序，使读者对计算机的功能与使用有个概括的了解．第二部分是BCY-乙算法语言，这部分浅显地介绍些语言的概貌，使读者能在较短时间内掌握之．第三部分是使用手册，这部分详细地介绍了BX 109-乙上机时所需的准备工作及碰到问题时的各种措施。

109-乙机
算法语言及其编译程序
使用说明

109-乙机算法语言编译小组著

*

科学出版社出版
北京朝阳門内大街137号
中国科学院印刷厂印刷
新华书店北京发行所发行　各地新华书店经售

*

1973年7月第一版　开本：850×1168　1/32
1975年4月第二次印刷　印张：2 9/16
印数：16,051—33,500　字数：63,000

统一书号：13031·120
本社书号：236·13—1

定价：0.34元

目　　录

第一部分　一般介绍

§1. 什么是电子数字计算机

电子计算机大致分为两类. 第一类是模拟计算机，它用电子线路模拟各种连续量的运算，包括微分和积分线路，这种计算机只能解决一些比较简单的数学问题，精确度仅达三位数字. 第二类是电子计算机，也称电子数字计算机. 一台快速电子数字计算机每秒能完成上亿次运算，一天能计算几百个题目，即使是一台速度较慢的计算机，每秒也能完成上万次运算. 手算需要几年的计算量，计算机只要几秒就能完成. 下面所讲的计算机都是指这一类计算机.

计算机的发展日新月异，使用范围越来越广，不但在科学计算上得到广泛的应用，而且在工业、商业、交通、医药、建筑、工程设计等方面也逐步开始使用，几乎渗透到各个领域中.

§2. 什么叫程序和程序设计

计算机运算速度之快，使得依循台式计算机的手工操作方式是不行了. 因此，使用计算机时，必须把计算过程和所用的数据预先通知计算机，由计算机按照规定的计算过程自动地进行计算.

计算机只能执行有限的几种基本运算，如加减乘除四则运算和逻辑运算等. 我们必须把解题算法用这些基本运算所构成的序列描述出来，这种序列称为程序，设计和编制程序的过程称为程序设计. 序列中所表示的每一个基本运算的元素称为指令，指令和数据必须用代码表示，穿在纸带上送入机器.

§3. 109-乙机的部件

109-乙机与其他计算机一样，有五个基本部分：运算器、控制器、存贮器、输入和输出等外部设备.

运算器的功能是完成各种基本运算. 109-乙机的运算速度平均每秒六万次.

运算器每次只能做一个运算，其余计算所用的指令和数据以及中间结果必须保存在存贮器的单元内. 乙机有二个磁心存贮器，各有 4096 个单元，每单元有 32 位二进制. 一个存贮器作存放指令用，称为指存；另一个存放数据和计算中间结果用，称为数存. 存贮器与运算器紧密相联. 另外还有两种外存贮设备，包括两台磁鼓和二台磁带机，用作与磁心存贮器交换信息.

控制器把存贮器中每条指令和指令所用的数据源源不断地送到运算器中. 计算机的操作均由控制器控制，它保证了程序的指令正确地按规定顺序执行.

109-乙机采用五一八单位输入机，把程序和所用数据预先按代码形式穿在纸带上，输入机把纸带上的信息转换成电脉冲送入寄存信息的存贮器中，然后开始计算，计算结果再从打印机上输出.

§4. 解题过程

用计算机解题共分六个阶段. 首先，把问题用数学形成表达；其次，确定计算方案，即找出适合计算机性能的计算方法. 例如计算 $\sqrt{N}$，显然用通常的开方方法或减奇数的方法是不合适的，这里我们采用牛顿迭代法求平方根.

第三阶段是程序设计，按照确定的计算方案画出框图，描述实现算法的思想. 例如用牛顿迭代法计算 $\sqrt{N}$，则先用 $N \div 2$ 的值作为第一个近似值 y_0，再从关系式

$$y_1 = y_0 - 0.5 \times (y_0 - N \div y_0)$$

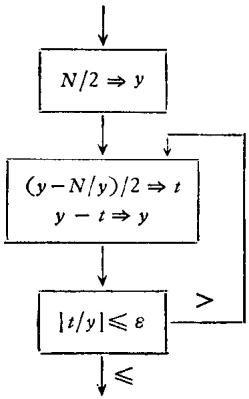
得到改进值 y_1，并从

$$y_{n+1} = y_n - 0.5 \times (y_n - N \div y_n) \qquad n = 1, 2, \cdots$$

求得各近似值. 重复此过程,直到相继的两个近似值满足

$$\left|\frac{y_{n+1} - y_n}{y_{n+1}}\right| \leqslant 10^{-6},$$

即相对误差小于 10^{-6} 的精确度为止. 实际计算时, $y_{n+1} - y_n$ 即为 $0.5 \times (y_n - N \div y_n)$, 框图如下:



	×	N	⟨0.5⟩	y
L:	÷	N	y	t
	−	y	t	t
	×	t	⟨0.5⟩	t
	−	y	t	y
	÷	t	y	t
	|<|	ε	t	L

上面的例子用 BCY 算法语言写出如下:

$N/2 \Rightarrow y;$

L:0.5 * $(y - N/y) \Rightarrow t;$ $y - t \Rightarrow y;$

若 0.000001 < § ABS(t/y) **则转** L **否;**

……

最后一个语句的意思是: 当满足关系时转到L, 重复迭代计算. 然后按照框图编出符号地址程序. 这里为了易读起见, 采用了三个地址的指令形式, 即把第一个地址中的数和第二个地址中的数按照指定的操作进行运算, 把结果送到第三个地址中. 上图右边最后一条指令的意思是: 当第一个地址中的数和第二个地址中的数满足指定关系(上例为按模比较)时, 就按第三地址指定的地址转移, 这就是说, 满足了关系, 下一条等待执行的指令就是前置标号为L的那条指令;否则继续下去.

编出了符号地址，再把它翻译成机器能够接受的信息的代码．首先对每条指令、所用数据和中间结果分配存贮单元，在这个例中，我们假设数据单元分配如下：

N	005	⟨0.5⟩	008
y	006	⟨ε⟩	009
t	007		

并假设指令从 010 单元开始顺序排列，+，−，×，÷ 和 |<| 用代码 1，2，3，4，5 表示，程序可写成如下形式：

010	3	005	008	006	$N/2 \Rightarrow y$
011	4	005	006	007	$(y - N/y)/2 \Rightarrow t$ ←
012	2	006	007	007	
013	3	007	008	007	
014	2	006	007	006	$y - t \Rightarrow y$
015	4	007	006	007	$\lvert t/y \rvert \geqslant \varepsilon$ 则 ←
016	5	009	007	011	

否则 ↓

第四阶段是做好上机前的准备工作，把程序和数据穿在纸带上．

第五阶段是调程序阶段．如果计算方案有问题或者程序编错了，也许穿孔有错，都会引起计算不能进行或计算结果不正确．因此，在正式计算之前，必须用计算机检查程序中的错误，然后修改程序，再次上机．重复此过程，直至程序完全正确为止．

程序调整无误，就能在计算机上正式开始计算了．

§5. 什么叫算法语言和编译程序

使用计算机的过程并不简单，需要许多道手续．首先要把解题算法用机器能够懂得的机器语言（即代码）描述出来，这种语言与自然语言或数学语言差别很大，因此，编制和调整程序需时较长，大大影响了计算机的推广使用．为了克服上述缺点，我们分两个步骤：先把解题算法用近似于数学语言的语言来表示，这种程

序称为源程序；然后再翻译成机器语言表示的程序，称为结果程序．众所周知，计算机能够完成复杂的逻辑运算，因而就能够利用计算机自身的功能来完成这种翻译工作，完成该项工作的程序称为编译程序．

如果源程序是符号程序这一级，这种语言称为汇编语言．它是一种面向机器的语言，与每台具体机器密切相关，各台机器都有各自的汇编语言；如果源程序是类似框图这一级，这种语言称为算法语言，它是一种面向数学的语言．由于在不同领域中处理对象不同，也就发展成各种不同的语言．现在最常用的共有二十多种语言，而每种语言又有不同的变异．在科学计算方面，较著名的有FORTRAN、ALGOL60、PL/1、BASIC 和 APL；作数据处理用，有COBOL 语言；作信息加工用，有 LISP 语言；等等．

FORTRAN（*formula translation* 的缩写）是美国 IBM 公司设计的语言．从 1958 年开始设计并试用，于 1962 年第一次正式发表．与此同时，在 1960 年巴黎国际信息加工会议上，确定了一种国际通用的算法语言 ALGOL 60（*algorithm language*）．我们根据 ALGOL 60 自行设计一种算法语言，名叫做 BCY．

§6. 为什么要采用 BCY

我们采用 BCY 作为乙机的算法语言，主要是为了适应数学工作者的要求，兼顾实现编译程序上的方便．我们对 ALGOL 60 和 FORTRAN 语言作了分析，并以 ALGOL 为蓝本，删去了不太常用的成分，如递归、固有量等．同时，为了使语言具有表达符号加工和数位运算的能力，在表达式中增加了字运算．另外还引进了控制台变量、已赋量以及调语句等，以适应调整程序之需要．

最后说明一下，BCY-乙 算法语言和 BCY 算法语言也有微小的差别．在这里，我们只介绍 BCY-乙 算法语言．

第二部分　BCY-乙算法语言介绍

§1. 计 算 语 句

我们先从一个简单的例子讲起．若计算 a 和 b 的“和”，结果为 f，按 109 乙机算法语言（以后简称 BCY-乙），则写为

$$a + b \Rightarrow f,$$

此处符号 ⇒ 称为赋值号，是有方向性的，其意义是右边变量的值取自左边计算结果，我们称它为计算语句．在执行前，a 和 b 的值必须给出；执行后，变量 f 的原有值失去意义，而代之以新的值．这个值一直保留下来，直到下一个计算语句给它另一个新值为止．

从上面看出，⇒ 和一般算术公式中的 = 有本质的区别．如 $a \Rightarrow f$ 和 $a = f$，前者表示 f 的值由 a 的值定义，后者表示 a 和 f 的值相等．同样，$-e \Rightarrow e$ 表示改变 e 的符号，而 $e = -e$ 相当于 $e = 0$．

§2. 名 字 和 数

上例中，a，b 和 f 表示简单变量（或称为**简变**），是名字的一种表现形式．在后面会看到名字有许多用处，它不但可用来表示简单变量，也可以用来表示一组变量或程序的位置，甚至可代表整个程序．因此，我们首先介绍什么样的符号串可称为名字．

一个名字是由 26 个拉丁字母和 10 个阿拉伯数字（共 36 个符号）任意组合而成，例如

$$a,\ \mathrm{eps},\ i1,\ B2R.$$

我们规定：第一个符号不能是数字，名字中间不能插入其他符号，例如

K-Point,　5thtime

就不能算是名字.

这里限制第一个符号不能是数字,是为了与数字区分开来,如计算语句

$$\text{Phi} - 15 \Rightarrow \text{rs}$$

的意义显然是变量 Phi 减 15, 而不是减名字 15 中的值.

为了避免穿孔编码时发生误解,规定拉丁字母 0 写作 θ, 以便与数字 0 区分开来.

为了便于实现编译过程,如果两个名字的前四个字符相同,就看作同一个名字,而且大写字母和小写字母也看作同一个符号.因此,编程序时要特别注意同名的问题.

在语句中允许直接写出十进制数,一个数通常包括三部分,即正负号、整数部分和小数部分也可以只有其中一、二部分,例如

$$0 \quad 177 \quad -200.3084 \quad .534 \quad -0.0012.$$

在编译时,这些数将自动转化为二进制数. 109-乙机规定,每个数的有效数位是 25 位二进制数 (相当于 7—8 位十进制), 而且数的 (绝对值) 表示范围是 (0.12×10^{-9}, 0.21×10^{10}), 即小于下限的数看作 0, 大于上限的数将溢出,从而引起停机.

§3. 表 达 式

计算语句的左边称为表达式,其中有名字、数以及四则代数运算 +, −, *(乘号), /(除号), ↑(乘方).

名字与名字,名字与数之间必须用符号隔开,以便区别名字的头尾.

乘法符号 * 不能节省, 也不能用小数点代替. 如 $a*b \Rightarrow x$, 不能写成 $ab \Rightarrow x$ 或 $a \cdot b \Rightarrow x$; 又如 15*5, 不能写为 15·5.

运算规则如下: 先乘除后加减,乘幂最先,按顺序从左到右依次运算. 在表达式中允许使用括弧, 但只允许使用圆括弧. 使用括弧的规则与代数中一样,由最内层往外依次运算.

乘幂运算 $a \uparrow b$ 处理如下: 当 b 是整数时,用连乘或连除来实现,其他情况则用 $e^{b\ln a}$ 来计算.

根据运算规则可以省略不少括弧，但有时难以分辨谁先谁后，我们宁可多用一些括弧．编译程序能够自动检查一个表达式的运算顺序，而编出合乎运算规则质量较好的运算程序．

§4. 语句的顺序

在实际计算中，往往需要许多语句来完成一个计算．例如我们用复数 $0.6+0.8i$ 乘复数 $x+yi$，计算结果仍保留在 $x+yi$ 中．为此引进辅助量 u，执行下面三个语句即可实现计算：

$$0.6*x-0.8*y \Rightarrow u;$$
$$0.8*x+0.6*y \Rightarrow y;$$
$$u \Rightarrow x$$

两个相邻计算语句之间的符号；称为语句分隔符，简称分号．前面讲过名字或数要有其他符号分隔，同样，语句也必须由非语句本身所用的符号来分隔，否则编译程序无法区分每个语句的头和尾．

§5. 语句括弧

在表达式中，可以使用圆括弧把某些名字和运算联系起来．用同样的方法，对语句引入两个语句括弧，开括符号**始**和闭括符号**终**．

语句括弧**始**和**终**分别表示程序的开始和结尾，语句括弧必须成对出现．其次，它们可以把一组语句括起来看作一个语句，称为复合语句．例如，

始
$$5/13 \Rightarrow x;\ 12/13 \Rightarrow y;$$
$$0.6*x-0.8*y \Rightarrow u;$$
$$0.8*x+0.6*y \Rightarrow y;$$
$$u \Rightarrow x$$
终

符号**始**和**终**是基本符号，在 BCY-乙中，类此符号共有 27 个，它们都被看作是单独的基本符号．

§6. 类 型 说 明

上面例子写得更完备一些,则为如下形式:

始　简变　x, y, u; $5/13 \Rightarrow x$; $12/13 \Rightarrow y$;

$0.6 * x - 0.8 * y \Rightarrow u$;

$0.8 * x + 0.6 * y \Rightarrow y$;

$u \Rightarrow x$

终

第一个计算语句之前的一些东西,不是语句,而是说明. 它的功能是指出复合语句中所用到的名字. 本复合语句中的名字 x, y 和 u 称为简单变量,也可以有另外一些说明,不外乎指出每个名字的意义和性质. 特别注意,各种说明必须放在语句的前面,语句之间不能插入说明.

简变是两个中文字组成的一个符号. 在它后面排列着一个或一个以上的名字,彼此用逗号分隔. 在最后一个名字之后用分号,以示与后面的语句分隔.

在语句串前插有说明的,为分程序. 无论是语句和语句,说明和说明以及说明和语句之间都用分号;隔开.

§7. 转 语 句

如果重复执行许多次乘法,我们引进一种新的语句来简化程序,名之曰转语句. 转语句就是中断语句的顺序使之循环执行,并指定另外一个语句作为它的后继语句.为了指出后继语句的位置,在该后继语句前面附一标号. 在转语句中也出现该标号,以便使整个程序衔接起来. 如上述例子,需要连乘,可写成如下形式:

始　简变　x, y, u; $5/13 \Rightarrow x$; $12/13 \Rightarrow y$;

AA: $0.6 * x - 0.8 * y \Rightarrow u$;

$0.8 * x + 0.6 * y \Rightarrow y$;

$u \Rightarrow x$;　　转 AA

终

其中 AA 是标号. 由于标号后面有冒号与语句分隔，所以极易识别，不必预先加以说明.

§8. 条 件 语 句

在实际计算中，不会出现无限次循环，总要求计算一定的次数. 例如要求计算 1000 次，则需执行转语句 999 次，也就是有条件地执行该语句. 这里介绍一种语句，可以实现有条件地转移，称它为条件语句. 条件语句即在程序中加上一个条件，当条件满足时，就执行这个语句；当条件不满足时，就执行另外一个语句. 写成程序，形式如下：

若 B **则** S_1 **否** S_2

这里 B 是条件，S_1 和 S_2 是任意一种语句，也可以是空语句. 如上例，若循环运算 1000 次，则可写成更完备的形式如下：

始 简变 x, y, u, k;
$5/13 \Rightarrow x$; $12/13 \Rightarrow y$; $0 \Rightarrow k$;
AA: $0.6 * x - 0.8 * y \Rightarrow u$;
$0.8 * x + 0.6 * y \Rightarrow y$;
$u \Rightarrow x$; $k + 1 \Rightarrow k$;
若 $k < 1000$ **则转** AA **否**
终

这个例子中 $k < 1000$ 是条件，S_1 是转语句**转** AA，S_2 是空语句. 条件中的 $<$ 称为关系符. BCY-乙中只有三种关系符：

$$<, \leqslant, =.$$

注意，没有 $>$ 和 $\geqslant$. 关系符 $=$ 具有 $\equiv$ 的意义. 在 109-乙中，$=$ 仅对整数才有意义，否则由于舍入误差，很难满足恒等条件.

我们常常要求当条件满足时执行一个以上的语句，这时需要把这几个语句用语句括弧括起来，看作一个语句. 例如根据 tc 1 的符号改变 ta 1 和 te 1 的符号，则写成

若 tc 1 $<$ 0 **则始** $-$ te 1 $\Rightarrow$ te 1; $-$ta 1 $\Rightarrow$ ta 1 **终否**

有时 S_1 和 S_2 又是条件语句，如将 a, b, c 三个简单变量中选

取最大值送 max，则写成

若 $a<b$ **则若** $b<c$ **则** $c \Rightarrow max$ **否** $b \Rightarrow max$

否若 $a<c$ **则** $c \Rightarrow max$ **否** $a \Rightarrow max$； S_3

注意，上例中前三个计算语句后没有分号，仅第四个语句后有分号，这个分号表示整个条件语句结束．这四个计算语句的后继语句都是 S_3。

§9. 循 环 语 句

复数相乘 1000 次的例子，若用循环语句写出，形式更为简单，书写如下：

始 **简变** x, y, u； $5/13 \Rightarrow x$； $12/13 \Rightarrow y$；

对于 $k=1$ **到** 1000 **步长** 1 **执行**

始 $0.6*x-0.8*y \Rightarrow u$；

$0.8*x+0.6*y \Rightarrow y$； $u \Rightarrow x$

终

终

在程序中，“**对于** $k=1$ **到** 1000 **步长** 1 **执行**”称为循环子句，表示 k 的值第一次取 1，以后每次增加 1，直到取值 1000 为止．如果“**执行**”后面的语句是由几个语句组成，那末就要用语句括弧把它们括起来，看作一个复合语句，称这个复合语句为循环部分．k 称为循环参数，在上例中仅起计数作用．在循环语句中可以引用循环参数，但不能改变它的值．例如近似计算

$$e=1+\frac{1}{1!}+\frac{1}{2!}+\frac{1}{3!}+\cdots\cdots$$

到 21 项，其程序如下：

始 **简变** $e, term$； $1 \Rightarrow e$； $1 \Rightarrow term$；

对于 $n=1$ **到** 20 **步长** 1 **执行**

始 $term/n \Rightarrow term$； $e+term \Rightarrow e$ **终**

终

循环子句的一般形式如下：

对于循环参数 = 初值**到**最后限值**步长**增量**执行**

这里的最后限值，意思是当超过限值时，就不执行后面的语句．于是

对于 $k=0$ **到** 25 **步长** 3 **执行**与**对于** $k=0$ **到** 24 **步长** 3 **执行**等价，因为 24 是最后一个值，下一个值是 27，已经超过 25．当步长增量不是整数时，为避免由于舍入误差而造成循环次数不对，最后限值应加上步长增量的 1/2．例如**对于** $k=0$ **到** 1 **步长** 0.1 **执行**应写作**对于** $k=0$ **到** 1.05 **步长** 0.1 **执行**．

初值，最后限值，增量都允许是表达式，写成语句则为

对于 $k=A$ **到** C **步长** B **执行** S_1

上述与以下语句序列相当：

$$A \Rightarrow k;\quad C \Rightarrow m;\quad B \Rightarrow n;$$

L_1：**若** $0<(k-m)*$ §SIGN (n) **则转** L_2 **否**；

$$S_1;\quad k+n \Rightarrow k;\quad \textbf{转}\ L_1;$$

L_2：……

这里相当于第一次 k 值等于 A，以后的新值用公式 $k=k+n$ 计算．每次用新值执行语句 S_1 之前先检查一下 $(k-m)*$ §SIGN $(n)>0$ 是否成立，当满足时就认为执行完了．由于先检查后执行 S_1，因此就有可能一次也不执行 S_1，这时相当于空循环．

循环参数极易辨认，无须用说明来定义．循环参数只能在该循环语句中有定义，越出此循环，循环参数就无定义．不能在循环语句之外使用转语句转入循环语句，相反可以从循环之内转到循环之外．

因为循环部分是一个语句，若需要执行几个语句，就要用语句括弧把它括起来．

§10. 场

计算语句中还有一种有力工具，叫作场．它可以表示矩阵、向量和它的分量．例如，我们用李勃门迭代解调和方程要用下述公式：

$$u_{i,k}^{(r+1)} = \frac{1}{4}(u_{i+1,k}^{(r)} + u_{i-1,k}^{(r)} + u_{i,k+1}^{(r)} + u_{i,k-1}^{(r)}),$$

其中 u 是二维场，而 $u_{i,k}$, $u_{i+1,k}$, $\cdots$ 是其分量. 在算法语言中，分量可写为 $u[i, k]$, $u[i+1, k]$, $\cdots$. 当 i, k 取某一固定值，分量也就确定了. 其中 i, k, $i+1$, $\cdots$ 称为下标，$u[i, k]$ 称为下标变量.

在引用场之前必须给出场的说明，指出那些名字是场，同时说明该场的维数和大小. 举例如下：

场 $A, B[1:5], C[-4:4],$

$R1[-10:10, 0:7], CEB[1:10, 1:7, 1:4].$

上例共有五个场 A, B, C, $R1$, CEB，其中 A 和 B 是一维向量，下标都是从 1 到 5 各有五个分量；C 也是向量，共有九个元素；$R1$ 和 CEB 分别为二维和三维的场，它们的元素个数各为 $21 \times 8 = 168$ 和 $10 \times 7 \times 4 = 280$. 下标变量的下标规定是整数，也可以是表达式. 当表达式的值不是整数时，取该数的整数部分(只舍不入).

下面以矩阵和向量乘法的程序为例，说明下标变量的使用方法：

```
始 场  A[1:6, 1:20], B[1:20], C[1:6];
   输十 A, B;
   对于 i = 1 到 6 步长 1 执行
   始   0 ⇒ C[i];
      对于 j = 1 到 20 步长 1 执行
           C[i] + A[i, j] * B[j] ⇒ C[i]
   终; 印十 C; 停 555
终
```

注意，下标变量的下标值必须在场的说明范围内，否则没有定义. 在场的说明中，要特别注意分隔符的使用方法.

在例中引进了三个语句：**停**，**输十**和**印十**. **输十**是根据场 A, B 的说明中指出的元素个数输入十进制数；**印十**是将场 C 以十进

制数的形式印出来；**停**是让计算机停止运算，表示计算机工作告一段落或发生意外（是程序设计时预料到的）．**停**后面的号码，称为停机号码，表示停机位置．在停机时，停机号码可以从控制台的指示灯中看到．停机号码为一到三位数字，也可以不写．

§11. 输入语句和印刷语句

输入语句和印刷语句都有两种形式，分别介绍如下：

（1）十进制数输入：**输十** $A1, A2, \cdots, An$

（2）十六进制数输入：**输字** $A1, A2, \cdots, An$

（3）十进制数输出：**印十** $B1, B2, \cdots, Bn$

（4）十六进制数输出：**印字** $B1, B2, \cdots, Bn$

其中 Ai 可以是简变和场，从鼓中逐个取出送到内存；若是**输十**语句，还需将十进制数翻译成机器表示的十六进制数．Bi 可以是简变和场，也可以是某些作标记用的十进制数．(3) 是十进制数的输出格式，(4) 是十六进制数的输出格式．在印刷语句中，Bi 还可以是表达式，其含意是将表达式的值计算出来并输出．例如，

印十 111，$x + A[1], y, B$；

例中 x，y 是简变，A 和 B 是一维场．若按十进制数格式输出，第一个数是 111，第二个数是 $x + A[1]$ 的值，第三个数是 y，第四个，第五个，……是 $B[1], B[2], \cdots, B[m]$．

输入语句和印刷语句中 n 必须小于 46，Bi 中出现表达式的个数应小于 8．当 Ai 是场及 Bi 是场和下标变量时，场的说明中界对必须已有定义．

输入的具体办法见第二部分．

§12. 成 组 传 送

计算语句中也有如下二种形式者：

（1）$E \Rightarrow V$　　E 是表达式，V 是场；

（2）$A \Rightarrow B$　　A，B 都是场．

(1) 是将表达式的值赋于右边场的每个分量，(2) 是将 A 中各