汉语程序设计语言

汉语程序设计语言

●沈志斌　著

●電子工業出版社

汉语程序设计语言

沈志斌 著

电子工业出版社

(京)新登字 055 号

内 容 提 要

本书介绍了一种新的计算机程序设计语言——汉语程序设计语言。它完全不同于英文程序设计语言，即不是将一般的程序设计语言中的关键字翻译而得。使用汉语编写计算机程序，只要略知计算机的运行机制就可根据自己的需要进行文字创作，而这一过程也就是程序设计的过程。这样既提高了编程效率，又给用户带来了极大的方便。

全书共分十章，分别介绍了汉语程序设计的意义、汉语编程的数据操作、算术运算、控制结构等，最后简要介绍了汉语编程语言的运行机制。书中各章内容衔接紧密，循序渐进。为了方便读者，本书在附录中还介绍了汉语编程语言的具体产品，并列出了汉语编程语言的词汇表。

本书可作为具有初中以上文化程度的计算机操作者和爱好者学习用汉语编程的入门教材，也可以作为使用汉语编程语言产品用户的参考工具书。

汉语程序设计语言

沈志斌 著

责任编辑 张潇

电子工业出版社出版

电子工业出版社发行 各地新华书店经销

北京怀柔东晓印刷厂印刷

开本：787×1092 毫米 1/16 印张：13.5 字数：340 千字

1994 年 5 月第 1 版 1994 年 5 月第 1 次印刷

印数：1—5000 册 定价：13.80 元

ISBN7-5053-2208-7/TP・580

前　言

我们目前使用的计算机语言、操作系统的版本几乎都是从国外引进的，所以就习惯于用英文编写程序。一般来说学英语比学计算机语言要花费更大的精力。而普通人在学习计算机之前，还必须先学一段时间的英语。如果我们有用汉语编程的计算机语言和操作系统，就可以抛开英语学习计算机。另一方面，英语在语法结构、语言习惯方面与中文差别较大。对于习惯使用中文的人，用汉语编写计算机的程序比用英语编写计算机程序有更多的优势。所以开发汉语计算机语言和操作系统势在必行。

笔者对汉语编程问题进行了多年的研究，认识到汉语程序设计不仅仅是把计算机语言的命令或单词进行汉化，更重要的是应该充分利用汉语的特点。汉语最大的特点就是，汉语是由很多具有独自含义的单字组成的，并由单字再组成词组，这些词组的意义一般都可以从单字的含义中"顾名思义"而来。

我们开发的汉语计算机语言也能象汉语一样，可以从大量的单字中去选择一定的字来组词，这些词既能象传统计算机语言中的语句一样，完成各种各样的操作，又可以象基本单字一样去组成新的单词，以便完成更复杂的操作。如此不断地发展下去，其语句的语义越来越广泛，最终接近于自然语言。程序越往后编就越能体现文字创作的艺术，因此程序员的语文水平决定着他的汉语程序设计水平。

计算机语言本身就是人编造出来的，汉语程序设计语言就是为了让中国人能够根据自己的习惯，使用汉语来设计计算机程序。

汉语程序设计经历了艰难的历程，现在已经进入了实用阶段，这受益于高效率汉字输入方法。掌握了高效率的汉字输入方法就具备了汉语编程的基本功，再稍付出一点努力——仔细阅读本书就可以掌握汉语程序设计。

本书在编写过程中承蒙吴克忠、沈林兴、李新华、刘梅等同志的大力帮助，在此表示衷心的感谢。

由于编者水平有限，时间又很仓促，书中一定存在不少缺点和错误，敬请读者批评指正。

沈　志　斌

一九九三年十月

目　录

第一章 概 述

第一节 引 言

计算机软件工程的发展最终可能要归结到文字表述的方式上，而与计算机本身却只有很少的联系。虽然这方面汉语比英语具有明显的优势。但是，至今还没有一种完全基于汉语的计算机系统和计算机语言，汉语在软件工程上具有的优势无法显露出来。只有设计和开发一种完全汉语化的计算机系统和计算机语言，并用其去发展计算机的软件，这样才能让汉语化的优势充分显示出来。

汉语化计算机语言必须具有汉语本身所具备的特点。西方人领先发展了计算机及其软件工程，而我们在开发一种新型的汉语化计算机语言时往往忽略了汉语本身的特点，依然跟随西文计算机语言的习惯，忘记了建立汉语计算机系统和语言的根本点。

经过多年的探索与努力，我们终于开发出了基于汉语习惯的程序设计语言——汉语程序设计语言及其产品。与西文计算机语言相比，汉语计算机语言毕竟是众多计算机语言家族中的一名新秀，随着它的应用与发展，相信会给计算机及其软件工程带来新面貌，并进一步地解放生产力。

第二节 什么是汉语编程

汉语编程是基于汉语习惯的，如果单纯把汉语编程理解为将普通的计算机语言中的关键字翻译成汉字，并可以在字符串中使用汉字，这种计算机汉语编程语言的意义是不大的。汉语编程必须最大限度地利用汉语本身的特点，以达到我们建立一种全新计算机语言的目的。

我们设计计算机程序的主要目的是预先指定计算机去干什么。在程序设计中比较多见的是设计所谓的菜单程序，也就是让用户在屏幕的提示下去选择下一步要操作的内容。假如设计一个程序，当运行这个程序时，就在屏幕上显示一句话，让用户选择输入一键值，然后根据不同的键值，在屏幕上显示不同的内容。在程序设计时为了便于管理，必须为这段程序起一个名称。有时为了便于记忆还需要将一些常用的数字规定成为一个好记忆的名称。如键盘上的数字键值，48 为“0 键”，49 为“1 键”…，依次类推，我们可以给这些键起名，同时为了便于记忆我们还给所编的程序起一个名，并说明此段程序将要完成什么样的工作。

```
48 为 0键        49 为 1键        50 为 2键
……
57 为 9键
编 菜单          （ 编一个其名称为“菜单”的程序 ）
                 回车 显″ 键入选择项目(0～9键)″
                 等键
```

```
如和      0 键   对应就   显″   您键入的是零键″
  而和    1 键   对应就   显″   您键入的是一键″
  而和    2 键   对应就   显″   您键入的是二键″
  而和    3 键   对应就   显″   您键入的是三键″
  而和    4 键   对应就   显″   您键入的是四键″
  而和    5 键   对应就   显″   您键入的是五键″
  而和    6 键   对应就   显″   您键入的是六键″
  而和    7 键   对应就   显″   您键入的是七键″
  而和    8 键   对应就   显″   您键入的是八键″
  而和    9 键   对应就   显″   您键入的是九键″
  其余就                  显″   不在选择范围″
而后
```

上面对我们希望设计的程序进行了简明扼要的描述，为了叙述更加清晰，在格式上做了一定安排。我们平时习惯于说“编程序”，可见计算机程序是“编”出来的。上述程序开始用“编”字做为程序的开始，最后用“。”表示程序的结束。在汉语程序设计中，以上的描述已经是一段完整适用的程序了。汉语程序设计语言很象我们使用的汉语，有很多基本的单字，在单字的基础上可以形成具有新的意义的词，这种新词的意义完全由单字的内涵所确定，可由设计者根据自己所要表达的意思用最恰当的汉语词汇来表达。有趣的是新词可以与原来固有的单字一样去参与组合其它新词。这种组合我们称它为定义。在程序设计中一方面是程序的内涵，另一方面则是程序的外在表现，即程序本身的名称。由于一个好的汉语名称可以精确地表达其实在含义，故在程序设计中只要见到简短几个汉字组成的词就基本上可以知道此段程序中要进行什么样的操作。可以想象，这样的程序设计方法随着程序层次的提高，其定义词的语义也越来越高级，程序设计与计算机之间的联系就越来越小。这样，程序设计的艺术将主要体现在程序员的汉语水平上。从软件工程的角度来看，这无疑是一大进步。但如果用英语来设计这样的系统，其优越性显然不如汉语突出。英语是以字母为基本单元的，字母中只有“a”和“I”表示一定的意思，其它字母并不表示什么含义。要表达一定的含义只有进行一定的排列组合。由于单词的拼法有要求，其字母的排列组合就受到一定的限制，加上英语中常用的单词已占用了很多较少字母的单词，所以英语单词的发展一方面是加长单词的长度，另一方面是以相近的单词来表示另一种意思。这就造成在定义单词时要么其长度过长，要么难以准确地表达。

汉语程序设计语言是一种基于汉语基础的计算机语言，它虽然有一定数量的单词，但为了满足编程的需要，还必须具备扩充的能力，你想说什么，就去定义一个这样词名的程序，程序的定义也是利用原先所存在的词来完成。

思考题

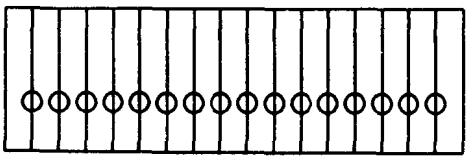
1. 通常情况下一本英文书翻译成中文后篇幅是变长还是变短了？
2. 从当前的录入技术分析，同样多的文字信息量是英文录入快还是中文录入快？
3. 说出五种以上汉字录入的编码方式。
4. 我们见到的程序文本与计算机实际运行的程序有何不同？

第二章　数据及其操作

第一节　数据的表示法

一、计算机中数的概念

我们提到的计算机，一般都是指电子数字计算机。中国人最早发明了数字计算机——算盘，现代的电子数字计算机在很多地方与算盘有相似之处，尤其是数的表达方式上都是采用数字方式。在计算机中用电平的高低来表示 0 或 1，而在算盘中则用算盘珠在上面还是在下面来表示数的大小。由于人们最初习惯于用十进制来表达数字，所以算盘一般都是采用十进制的。如果我们将算盘的表达方式与计算机的表达方式达成统一，就可以把我们用的十进制算盘改为二进制算盘，这样算盘的每个杆上就只有一个珠子，算盘珠拨上去表示为 1，拨下来表示为 0。



一个算盘珠只能表示两种状态，0 或 1，两个算盘珠就可以表示 4 种状态；算盘全部拨下表示 0，最右边的单独拨上去表示 1，左边的单独拨上去表示 2，两个算盘珠都拨上去则表示 3。这四种状态分别 00，01，10，11 表示。如果要表示更大的数则只有将算盘珠增多。三个算盘珠可以表示 8 种状态，那么按 2 的 n 次方的算法依次类推分别为：

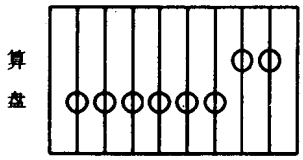
16，32，64，128，256，512，1024，2048，4096，16384，32768，65536...。

算盘的大小是有限制的，计算机内部的表示也是同样，一般将其分为 8 位、16 位、32 位、64 位等等。为了不至于将运算位取得太大，又能满足于实际应用的要求，我们将 16 位作为一个运算基本的单位。一个 16 位的数可以表示 65536 种状态。作为一个 16 位的正整数，可以从 0 一直表达到 65535。但是作为一个有符号的数，则往往用 16 位最高位为 1 来表示这个数为负数，这时这个 16 位的数只能从－32768 表示到＋32767 范围。16 位的数一般又称为单精度数。

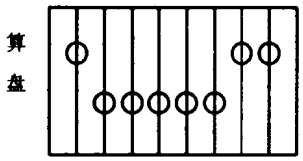
上述一个二进制的数可以表达一定的数值范围，但它是如何表达这个数的实际值呢？对于一个无符号数，我们可以按上述二进制算盘计数的方法依次类推，对 16 位数的每一种状态找到其从 0～65535 的对应值。但对于有符号数来说就不能简单地套用这种办法。在有符号数的二进制的表达方式中，我们分别引入原码、反码和补码三种概念。为了简单起见，我们以 8 位数来说明有符号数三种编码的表示方法。

二、带符号数的表示法

一个数可用其最高位来表示符号，正数的最高位为0，负数的最高位为1，其余位仍按正常方法用来表示数值，这种表示方法叫做原码表示法。如：



0 0 0 0 0 0 1 1　表示　3



1 0 0 0 0 0 1 1　表示　−3

原码表示简单易懂，而且与机器的真值转换方便。但若是两个异号数相加(或两个同号数相减)，就要做减法。为了把上述运算转换为加法运算就引进了反码和补码的概念。

正数的反码表示与原码表示相同，最高位为符号位，用“0”表示正，其余位为数值位。如：

	符号位	二进制数值
(＋4)	= 0	0 0 0 0 0 1 0 0
(＋31)	= 0	0 0 1 1 1 1 1
(＋127)	= 0	1 1 1 1 1 1 1

而负数的反码表示，即为它的正数的按位取反(含符号位)。

	符号位	二进制数值
(＋4)	= 0	0 0 0 0 0 1 0 0
(－4)	= 1	1 1 1 1 1 0 1 1
(＋31)	= 0	0 0 1 1 1 1 1
(－31)	= 1	1 1 0 0 0 0 0
(＋127)	= 0	1 1 1 1 1 1 1
(－127)	= 1	0 0 0 0 0 0 0
(＋0)	= 0	0 0 0 0 0 0 0
(－0)	= 1	1 1 1 1 1 1 1

8位二进制数的反码表示有以下特点：

1. “0”有两种表示法。

2. 8位二进制反码所能表示的数值范围为＋127～－128

3. 当一个有符号数用反码表示时，最高位为符号位。当符号位为“0”(即正数)时，后面的七位为数值部分；但当符号位为“1”(即负数)时，后面几位表示的不是此负数的数值，一定要把它们按位取反，才表示它的二进制值。

二进制表示的另一种方式为补码方式。正数的补码表示与原码表示相同，即最高位为符号位，用“0”表示正，其余位为数值位。如：

	符号位	二进制数值
(+4)	= 0	0 0 0 0 1 0 0
(+31)	= 0	0 0 1 1 1 1 1
(+127)	= 0	1 1 1 1 1 1 1

而负数的补码表示即为它的反码，且在最后位(即最低位)加1。如：

	符号位	二进制数值	
(+4)	= 0	0 0 0 0 1 0 0	原码
(−4)	= 1	1 1 1 1 0 1 1	反码
(−4)	= 1	1 1 1 1 1 0 0	补码
(+31)	= 0	0 0 1 1 1 1 1	
(−31)	= 1	1 1 0 0 0 0 0	
(−31)	= 1	1 1 0 0 0 0 1	
(+127)	= 0	1 1 1 1 1 1 1	
(−127)	= 1	0 0 0 0 0 0 0	
(−127)	= 1	0 0 0 0 0 0 1	
(+0)	= 0	0 0 0 0 0 0 0	
(−0)	= 1	1 1 1 1 1 1 1	
(−0)	= 1	0 0 0 0 0 0 0	

8位带符号的补码表示有以下特点：

1. (+0)=(−0)=00000000
2. 8位二进制补码所能表示的数值为：+127～−128
3. 一个用补码表示的二进制数，最高位为符号位，当符号位为“0”(即正数)时，其余七位即为此数的二进制值；但当符号位“1”(即负数)时，其余几位不是此数的二进制值，把它们按位取反，且在最低位加1，才是它的二进制值。当负数采用补码时，就可以把减法转换成加法。例如：

$$64-10=64+(-10)=64+[10]_{补}$$

$$+64=01000000$$

$$10=00001010$$

$$[-10]_{补}=11110110$$

于是

$$\begin{array}{r} 0\ 1\ 0\ 0\ 0\ 0\ 0\ 0 \\ -0\ 0\ 0\ 0\ 1\ 0\ 1\ 0 \\ \hline 0\ 0\ 1\ 1\ 0\ 1\ 1\ 0 \end{array} \qquad \begin{array}{r} 0\ 1\ 0\ 0\ 0\ 0\ 0\ 0 \\ +1\ 1\ 1\ 1\ 0\ 1\ 1\ 0 \\ \hline [1]\ 0\ 0\ 1\ 1\ 0\ 1\ 1\ 0 \end{array}$$

↓

自行丢失

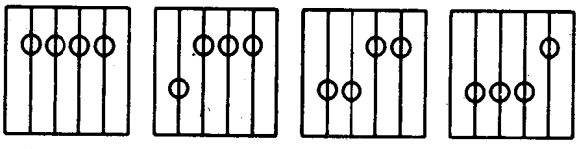
一个算盘的位数是有限制的，在做运算时最高位将自行丢失，故做减法与补码相加的结果

是相同的。同样我们可以将这种表示法扩展到16位以至32位。计算机中在许多情况下都是用二进制补码的方式来表示数值的。汉语编程语言在大部分情况下也用16位的二进制补码来表示数值。

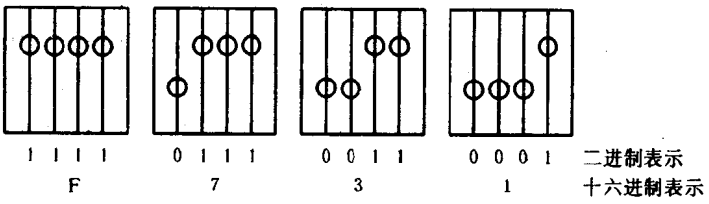
三、数的十六进制表示法

虽然计算机中数是以二进制方式存在的，但表达一个二进制数需用很长的字符串，而直接采用十进制表示，做二进制转换又比较麻烦。用十六进制来表示数，就能克服这些矛盾。

我们将一个16位的算盘分为四等份：

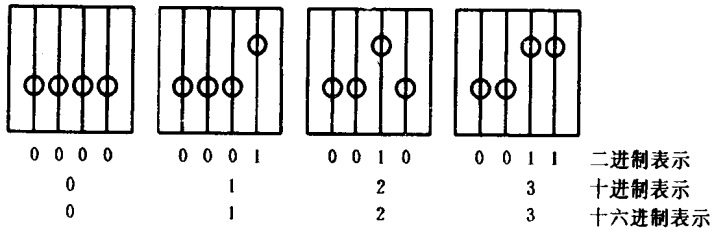


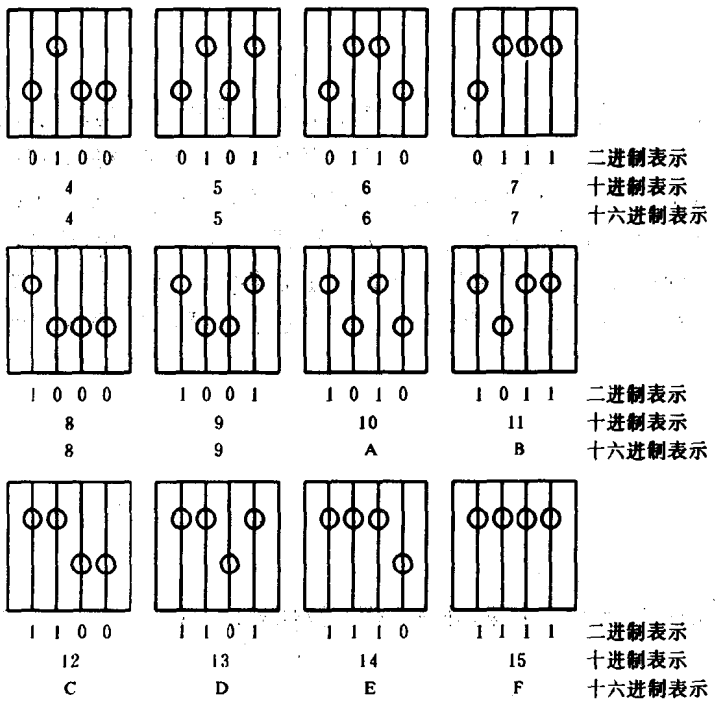
如果把每个小算盘看作一位数，那么它能表示的数的范围是0～15。在我们习惯的十进制表示法中一位数字只能表示0～9，从10开始需用两位以上的数字表示。若想用一位数来表示十以上的数字，则可利用那些最常见的、使用最方便的字符。在十六种状态中，零至九仍用阿拉伯数字0～9表示，而十至十五则用英文字母A～F表示，这就是数的十六进制表示法。这样上面的每一个小算盘的十六进制表示法就是：



从单独一个小算盘很容易看出十六进制与二进制的对应关系。巧妙的是四位二进制数正好是一位十六进制数。这样在识别一个十六进制数代表的二进制状态时，可以按位进行二进制变换，且变换过程中不会象十进制转换为二进制那样，转换一位而影响其它位。

为了便于理解我们将一个四位小算盘三种数制的对应关系列出：





以后进行二进制与十六进制之间的相互转换，可以根据上面算盘的对应关系来完成。但要进行多位的十进制转换则必须进行计算。

四、数据的概念及表示方法

假如让你当一名计算员，你的计算工具仅有我们提到的算盘，而没有纸和笔。在进行大量的数据计算时，为了提高计算速度和便于记忆，最好的办法就是将算盘有序存放，比如将算盘摞起来。为了保证在一定的空间可以放足够多的算盘，就将参加运算的数据以及结果都摞放起来。每次总是将参与运算的数先按一定顺序摞在顶上，然后根据这些数进行规定的运算，在得到结果时先将参与计算的算盘拿走然后再将表示结果的算盘摞在顶上。例如进行一个最简单的加法运算，3＋4 我们先将表示 3 的算盘摞在顶上：

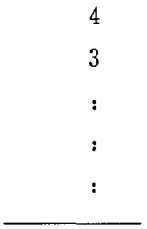
3

：

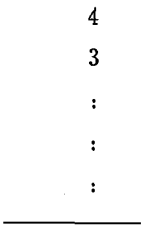
：

：

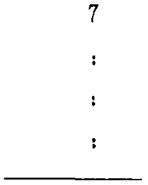
横线表示底线，在 3 和底线之间可能还有以前的摞上去的算盘。然后再将表示 4 的算盘摞在顶上：



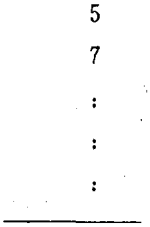
当有了两个数后就可以做加法了，其结果为7。这时参与运算的3和4已经没有用处就将它们移去，再把结果的7摞在顶上。其实我们不必总是考虑用算盘往起摞，而只需考虑其中的数就可以了，实际在计算机中也只需考虑这些数值，而不需考虑这些数值是用什么摞起来的。为了便于表达，我们称这个摞数的地方为“数摞”。① 有了数摞的概念，以后我们都按上面的这种方法来进行所有的计算。计算机的运算往往是连续的，前面计算的结果可能是后面需要的参数。如计算(3+4)*5则按上面的方法先将3摞在数摞顶上，再将4摞在数摞顶上：



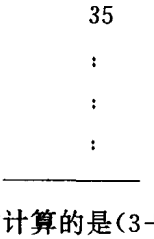
做完加法后将4和3去掉，并将7摞在数摞的顶上：



然后再将5摞在数摞的顶上：



最后做完乘法后将5和7去掉，将乘法的结果35摞在数摞的顶上：



我们刚才计算的是(3+4)*5，但是操作的顺序却没有按这个式子进行，而是3 4 +

① 数摞的概念与堆栈概念不同，堆栈是一种先进后出的数据结构，而数摞在外特性上却具备了对数据进行重排序的特点。

5 * 。这种表示并不奇怪，无论做什么样的计算首先应该有被计算的数据，用(3+4)*5只是一种习惯，叫做中缀表示法。而用 3 4 + 5 * 叫做后缀表示法，或者叫做逆波兰表示法，是由波兰数学家J·卢卡西维兹(J. lukajewicz)于1929年首先提出的一种表达式表示法。采用后缀表示法同样也是一个习惯问题。这种表达式可以省去许多代表优先级的符号。

因为采用了数据技术，也就适合用后缀表示法。在汉语程序设计中始终利用数据来完成常规的运算。所以我们必须弄懂数据的概念，并熟练地运用于程序设计之中。

第二节 数据操作

在标准的汉语程序设计中所建立的数据均以16位为基本单元。我们可以将数据上的一层想象成为一个具有16个杆和珠的算盘，每一个数都是由这种16位为基本单位的算盘来表示的。在一切的操作与运算过程中其操作数都将从数据上取得，而操作结果也将放在数据的顶上。这样显然带来一个问题，有的时候操作结果并不一定是后面操作所需要的操作数，或者即使是操作数，但按后面操作的要求其顺序不对，或者后续的操作只需要部分的结果，使得我们不得不将数据上的数据进行必要的调整以及复制。在汉语编程语言中提供了一系列数据操作算符，以便在程序设计中对数据上的数据进行适当的调整。

一个人说话是要表达自己的意图，为了达到这一目的，在说话时可能会参杂许多不可缺少的“废话”，舍此，此段话可能就“不像话了”。程序设计也是如此，如汇编语言中要频繁的保护寄存器，高级语言中却要不断的进行变量操作。在汉语编程语言中，出现最多的“废话”就是数据操作，然而正由于这些“废话”才使我们的程序设计变得更为精巧和生动。

数据操作是汉语程序设计中的一个难点，但由于它使用极为频繁，所以一旦掌握，也就得心应手了。

一、数据的交换

现在你仍然充当计算员，但计算过程由指令员来指挥。指令员要做一个算术运算20—3×4，按照算术运算的优先规律应该先乘除后加减。所以指令员先将两个要乘的数告诉你，由你顺序将这两个数放入数据：

4
3

接下来指令员又发出乘法指令，你计算出3×4的结果，并将数据上的3和4去掉，然后将结果12放入数据：

12

指令员又将20告诉你，你随即将20放入数据：

20
12

这时如果指令员发出减法的指令就出现了问题，因为减法没有交换律，按规定必须是前面的数减去后面的数。在我们数据上前面的数就是先进入数据的数，如果直接进行减法操作，就

变成了 12－20，为了解决这类问题，引入了一些数据操作符。象上面的减法问题只要将两个数进行一次交换再执行减法就可以了。我们用“↑↓”表示交换操作。这样指令员在发出减法指令以前就必须先发出“↑↓”指令，你就将数据上数据的顺序交换一下：

12
20
———

然后指令员再发减法指令，你在做完减法后就可以将 12 和 20 去除，并将计算结果 8 放入数据：

8
———

这个过程如果用汉语编程言语的程序来表达就写成了下面的形式：

3 4 * 20 ↑↓ －

以上只是为了说明问题举了一个例子，当然对于这样一种简单的计算完全可以写成另一种简化的形式

20 3 4 * －

我们放物品并不一定都是竖向摞置，有时要横向摞放，如书架上的书基本是横向摞放的。为了表达方便，我们在表示数据操作时常常也是用横向方式来表示数据里的状态。我们写文章时习惯从左至右来，如果写的是数就进入数据保存起来，所以先进入数据的就在左面而后进入的就在右面。用横向法表示数据时则记忆为右边的数在上，左边的数在下。如 1 2 3 则表示：

3
2
1
———

为了表达简便，我们还引入操作符号“———”，并将操作前的数据状态写在“———”的左边，而操作后的数据状态写在右边。数据上的数可以是一些代号，如“n1”、“n2”等等。如上面的“↑↓”操作，就表示为：

↑↓ n1 n2 ——— n2 n1

它代表

n1 n2 ↑↓ n2 n1

而对于“＋”操作就表示为

＋ n1 n2 ——— n3

其中 n3 是 n1 与 n2 之和。

有了以上的基础知识，我们就可以进行数据操作的学习和练习了。在汉语编程系统中，系统的提示符为“★”。不同的汉语编程系统其启动方式会有所差别，具体方法可参照系统版本的说明书。提示符“★”表示系统等待你输入。在提示符“★”的状态下可以直接输入数值，并通过一些数据的显示操作去查看数据。下面是操作过程：

★
1 2 3<Enter>★

其中“<Enter>”表示按“Enter”键

看数据<Enter>