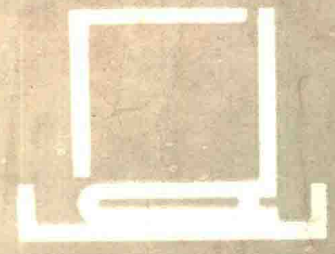
微型计算机常用的十种高级语言及其比较



旅游教育出版社

微型计算机常用的
十种高级语言及其比较

陈敏修
朱德锋 编译

旅游教育出版社

内 容 提 要

本书介绍微型计算机常用的十种高级语言：**BASIC**、**C**、**COBOL**、**FORTRAN**、**LISP**、**Logo**、**Modula-2**、**Pascal**、**Forth**和**PILOT**。采用同样的格式，列举了一些同样的程序例，使读者能比较出在不同应用中采用哪种语言最好。讲述内容深入浅出，力求简明、易懂，每种语言都有一个较大的实用程序例。涉及的语种多，可供不同层次的读者学习使用，也可作为大学计算机专业高年级学生和研究生的参考书，及计算机语言学习班的教科书。

微型计算机常用的
十种高级语言及其比较
陈敏修
朱德锋 编译

旅游教育出版社出版
（北京市朝阳区定福1号）
新时代出版社印刷厂印刷
新华书店北京发行所经销

规格：787×1092毫米1/16　24.25 印张　500 千字
1990年7月第1版　　1990年7月第1次印刷
印数：6000 册　　定价：12.00元

ISBN7-5637-0135-4
TP·002

序

在推广使用微型计算机的过程中，高级语言的作用是不言而喻的。目前，广大计算机应用人员，甚至专业软件工作者一般也只掌握一种、最多二种或三种高级语言。每种语言往往只对解决某些问题是比较好的工具，而对于可能遇到的其它问题就不一定是最佳的了。所以，多掌握一些语言是很有必要的。然而，用一般的方法学习一种高级语言，需要比较长的时间。所以，很多人虽有学习多种计算机语言的愿望，却因花不起足够的时间而不能实现。两位编译者根据多年的实践，在深入研究泰勒教授写的**The Master Handbook of Highlevel Microcomputer Languages** 一书的基础上，从较高的学术水平的角度、结合丰富的实践经验编译了这本书，对解决上述问题是会有帮助的。这是一本比较和学习多种计算机语言的入门书，也是查阅各种常用语言的实用工具书。于此，我们向广大读者推荐这本好书，愿读者通过这本书能够在开阔思路、掌握新的计算机高级语言方面有较大的收益。

吴几康　殷志鹤

1989年于北京

注：

吴几康现任中国科学院计算所研究员、中国计算机学会副理事长。

殷志鹤任北京电子振兴办软件负责人、北京软件行业协会秘书长、高级工程师。

引　言

本书采用统一的方法，描述微型计算机使用的十种高级语言，讲述每种语言的主要特点及其用途，并进行比较。目的是使读者能通过这本书可以对每种语言有个比较全面的了解，并能够根据自己的问题，选用合适的语言。

书中列举的十种高级语言是：BASIC、C、COBOL、Forth、FORTRAN、LISP、Logo、Modula－2、Pascal、PILOT。各章除一些较短的程序外，每章末都有一个比较长的综合程序实例，以便使读者能对每种语言的实用程序可以有个感性认识。

为了便于对各种语言进行比较，在每种语言中都列举了一些相同的程序实例，使读者能看到怎样用不同语言来编写具有相同功能的程序。

书中所举十种语言都可以在APPLE Ⅱ系列微机（有的要求加插Z—80卡）及IBM PC微机上运行。

本书主要参考了Charles F. Taylor〔美〕写的 The Master Handbook of Highlevel Microcomputer Languages 等书籍。

当然，在这么短的篇幅中，也不可能把每种语言的所有特点都讲到。我们的意图是讲出各种语言的主要特点，阐明其本质，并进行比较。对于那些因语言的具体实施而不同的特点，我们尽量少介绍。

由于我们的水平所限，领会也还不够深刻，不当之处在所难免，敬请广大读者批评、指正。

本书的读者最好是已初步掌握一种高级语言（例如BASIC 或FORTRAN 等）的编程，并在计算机上运行过的。

本书可供广大计算机应用人员学习、参考，也可作为大专院校学生学习高级语言的参考书籍。

本书承蒙田淑清副教授仔细地阅读了手稿，并提出了不少宝贵建议，在此表示感谢。

IEEE 高级会员　陈敏修
朱德锋
1989年于北京

目　录

第1章　什么是计算机高级语言？

微型计算机的心脏是叫作微处理器的集成电路，它构成微型计算机的所谓中央处理单元CPU。所有的运算都是在CPU中进行的。早在1983年只要花3.95美元就可以买到大拇指那么大小的一片微处理器，它的运算能力相当于20年或25年前最大的计算机的运算能力，而后者价值却是几十万美元，并且体积还很大。可见时代是改变了。

当然，花3.95美元是买不到一台微型计算机的。因为，如果没有电源、输入输出设备、存贮器及各种接口电路的话，CPU也就没有多大用处了。因此，总的价格当然是要远远高于3.95美元的，最简单的微型计算机价格低于100美元，而最复杂的则高于10,000美元。

• 机器语言

目前微处理器已由8位机、16位机发展到32位机。市场上主要的有Intel公司的8080A、8085、8086、8088、80186、80286、80386，Mostek公司的6502A、6502C、65802 65816，Motorola公司的6800、6809、68000、68010、68020，Zilog公司的Z80、Z80A、Z 8000、Z 80000等。每个微处理器都有它自己的语言，即机器能懂得的指令集。这种语言称为机器语言。机器语言由一些0和1组成，即二进制数字所组成。但是，即使是把这些二进制数写成十六进制数（基数是16），使用起来仍然是十分困难的，并且各种微型计算机的机器语言又各不相同（虽然在8080A、8085与Z 80A之间，在8086与8088之间有些共同性，但是机器语言仍不通用）。

• 汇编语言

从语言的结构上来看，机器语言上面的一层是汇编语言。汇编语言是用助记符来代替每个机器语言指令代码。助记符由二个或更多个字母组成，通常是英文单词的缩写，这些单词描述了所要完成的操作。例如，在8086汇编语言中，助记符INC BP意味着使寄存器BP的内容增加1（INCrement BP register），与此等效的机器语言是二进制的01000101或十六进制的45。

由于计算机不认识助记符，所以用汇编语言编写的程序必须要翻译成机器语言，计算机才能识别。这可用称为汇编程序（assembler）的计算机程序来实现“翻译”。

助记符INC BP比写成01000101或45确实是一个改进。不过，机器语言指令与汇编语言指令之间在本质上仍然存在着一一对应的关系。机器语言与汇编语言常统称为低级语言，这意味着要取许多的指令才能实现一小步的处理，并且程序员必须要知道许多机器内部的细节，还要清楚在程序执行过程中这些细节的变化情况。

低级语言的编程是困难的，学习也很困难。为一台计算机编写的低级语言程序，在另一种计算机上不能运行。现在市场上的微型计算机五花八门，如果只有低级语言的话，那它的销售及应用将不知会出现什么情况。

• 高级语言

高级语言是在五十年代后期引入大型计算机的。最早推出的是FORTRAN 和COBOL ，之后出现了许多种其它高级语言，其中有BASIC 、LISP 和Pascal 等。

那么，高级语言与低级语言有什么区别呢？它们的区别表现在几个方面。首先，基本的区别是高级语言的一个语句相当于低级语言的许多语句。因此，程序员只需要花费较少的力气就可以完成许多工作。

高级语言基本上与机器无关。因此，在一种计算机上可以运行的COBOL 程序，通常只需要稍作修改就可以在另一台计算机上运行。此外，高级语言的程序设计员不需要知道他所使用的计算机的更多细节。

高级语言一般比低级语言容易读懂。例如，要把两个数（A 和B ）相加，得到第三个数（C ），用8086汇编语言可以写成：

```
CLC              ；清进位标志
MOV  AX，A       ；把A装入累加器
ADC  AX，B       ；加B（有进位）
MOV  C，AX       ；结果存到C
```

此处，假定问题中的数据及其运算结果A 、B 、C 都是在0到255之间的整数，所以写出来的程序还算短，否则将要长得多。然而，在BASIC 语言中，该例只需要写成：

```
LET C =A +B
```

即可。在COBOL 中，则可写成：

```
ADD A TO B GIVING C
```

即使是这样一个简单的例子，也可以看出高级语言是很容易读的。如果数的位数比较长、又有小数的话，汇编语言写出的程序就会变得复杂得多，而高级语言写的程序却仍旧不变。

象汇编语言一样，高级语言编写的程序，在运行前也要先翻译成机器语言程序。这种“翻译”通常有两种做法：用编译程序或解释程序。编译程序把整个程序一下子都翻译成机器语言程序，便可被执行；而解释程序却是一次翻译程序的一行，译出一行便立即执行，即边解释边执行，直至程序结束。

一般来说，编译方式比解释方式的执行速度要快。这是因为不管程序中的某一行要执行多少次，编译程序对每一行都只翻译一次，然而解释执行的程序却是执行一次就要翻译一次。另一方面，解释方式却不需要额外的编译步骤，因此，容易使用。有些语言，例如BASIC ，既有解释方式，也有编译方式。但是，大多数语言只有一种方式，或者是解释方式或者是编译方式。

• 高级语言与低级语言的比较

低级语言的主要优点是执行速度快。同一个程序，如果用汇编语言编写往往要比用高级语言编写的程序运行速度快几倍。

有时速度是很重要的。例如，一个在生产环境中需要多次执行的程序，在这种情况下，

速度效益就是问题的关键。通常，用低级语言比用高级语言编写程序和调试程序所用的时间更长。

低级语言的另一个优点是它很容易利用具体机器的核心操作，这对写编译程序和操作系统等软件工作来说，是很重要的。

高级语言有其本身的优点，比低级语言易学、易用，而且使用者也不需要知道具体计算机的细节情况。

一般来说，程序员用高级语言可以有更高的成效。同样的操作，高级语言只需要写很少的几步，而低级语言却要写出好多步。研究表明，用高级语言编写程序的效率比较高。

低级语言程序都是针对具体计算机的。对于一种高级语言来说，用它编写的程序的移植性更好。用高级语言编写的程序，通常可以在不同类型的计算机上运行，而用低级语言写的程序却只能在一种计算机上运行。

高级语言的另一个优点是易读性，这对于需要维护或改进的程序来说，不管是由编制者本人还是其它人来进行，都是十分重要的。尽管由于编写者的水平问题，有的高级语言程序写得也不容易看懂，但高级语言本身的易读性却是显而易见的。

• 本书的目的

本书从使用者的角度来讨论高级语言。第二章对高级语言作一全面的讨论，从第三章开始讨论目前微型机所具有的各种高级语言的具体特点。重点是介绍每种语言的特色，以使读者对各种语言写出的程序能有一个感性认识，探讨每种语言的优点和弱点。

对于每种语言特征的介绍，侧重点放在各种计算机都通用的那部分。对于与具体机器有关的性能，例如绘图，在可能的情况下尽量少谈（主要的例外是 **Logo** 语言，因为绘图是**Logo** 语言的主要功能）。同样的理由，本书也不过细地讨论有关如何在一台具体微机上建立和运行一个程序的细节。读者如果需要这方面的知识的话，可以查阅有关计算机的语言手册。

我们始终确信，现在没有、也许将来也不会有那样的一种语言，它对各种各样的应用都是最好的。所以，多懂得几种语言是很有好处的。因为，在某种应用场合，某种语言可能是最适宜的，然而对另一种应用，它就不一定适宜，而别的一种语言却是最好的了。用某种语言写的很长的复杂程序，换用另一种语言时，则可能会变成很短的一个小程序。程序设计语言是建立计算机程序的一套工具。如果一个程序员只能用一种编程语言编写程序的话，就有点象只会使用少数几种工具的木匠一样，虽然也能完成任务，但其质量和速度就不如应有的那样好，那样快了。

第2章　高级语言的一般特点

本章将对各种高级语言共同具有的特性作一般性的讨论，以便在讨论各个语言时有个基础，也便于对各种语言进行比较。

§2-1　数据表示

计算机常常也叫作电子数据处理机。各种计算机都是处理数据的，只是这些数据的表示形式各不相同。所以，看看在高级语言中怎样表示数据是有益的。

§2-1-1 常量

数据值不变的数据称为常量。例如，12就是一个常量；它总是表示同样的数值。

计算机用二进制数字系统（基数是2）存贮数据，这是因为二进制的1和0直接与数字电子器件的两种状态相对应。在使用高级语言时，程序员并不需要考虑这么多，但是却应该知道高级语言所采用的数据表示形式。由于用十进制所精确表示的数，有时并不能用二进制精确地表示出来，这样就会产生一些并不希望有的结果（即产生误差），其理由我们将在下面讨论。

机器内二进制系统有两种数据表示形式，一种是整数，一种是实数。整数是没有小数点的数，例如1、2和－5；而实数是带有小数点、有时还有指数，例如1.0、3.1415926、－2.8或1.2E5。其中，最后一个读作1.2乘以10的5次方，这个E表示10的乘方。这种数的表示形式称为科学记数法。

在计算机中，整数是可以准确表示的，但数值范围有限。微型计算机上整数值的典型范围是－32768～32767。这个范围受存放每个数据所占用的内存大小的限制（一般是16位二进制位或2个字节），也受到其内部表示法的限制。

实数在计算机内部表示成指数和尾数两部分。举例来说，1.234567890 在机内表示成“0.123456E1”。其中，E1是指数部分，意思是“乘以10的一次方”；0.123456是尾数部分，小数点后的最后几位由于允许保留的有效位数受到限制而丢失。有效位数一般是6位或7位，这与分配给存放尾数的存贮空间大小有关。指数的典型范围是±38，它与分配给存放指数部分的存贮空间所占的（二进制）位数有关。

实数往往不能在计算机内准确表示的另一个原因，是因为在机器内采用了二进制系统。十分之一在十进制中可以表示成0.10，只用了有限的几位数字；三分之一在十进制中要准确地表示要用无限多位（0.3333……）。而在二进制中，上述二个数都不能用有限个数位来表示，比如十分之一就被看作是0.09999 ……。

有些微型机上的高级语言支持所谓的双精度实数。对双精度数分配两倍的存贮空间，以便保留较多的有效位数字（一般是14位或17位）。这在统计学及科学应用范围是很有用的。

在大多数应用中，整数、实数或双精度数是足够用的。但是，在金融方面的应用却是

例外。会计员要求他们帐簿中的余额能精确到几分钱，而不能容忍诸如“我的计算机不能把所有数都表示得十分准确”之类的话。解决的办法是采用一种编码的方案，即在机器内部不再使用通常的二进制数方式表示数，而是把十进制数的每一位都译成二进制码，这种方案要占用较多的存贮空间。例如，在二进制中用8个二进制位可以表示0～255之间的数，而在这种方案中只能表示0～99之间的数。这种方案叫作二——十进制的 BCD （binary coded decimal ），即二进制编码的十进制，它可以在机内准确地表示出十进制数。读者将会发现，只有少数几种高级语言具有这种特点。

到目前为止，我们只讨论了数的表示方法，计算机还需要表示字母和其它字符，通常采用美国信息交换标准码（ ASCII ）来进行这项工作。在 ASCII 码中，用65表示字母 A，用55表示数字 7，而用37表示 %，这些都是字符的例子。一串字符称为字符串或简称为串，这种字符串型数据也常称为字母数字数据。

有些高级语言还有一种称为逻辑型或布尔型的数据（取名于十九世纪数学家、逻辑学家乔治·布尔）。布尔型数据只有二个值：“真”（true ）和“假”（false）。

§2-1-2 变量

计算机语言中的变量往往被看成是数学公式之外的一种非常神秘的东西。然而，变量实际上一点也不神秘；我们可以把变量想象成是存贮数据值的一个区域。事实上，变量只是给计算机内存中某个存贮区取的一个名字，在这个区域中将存放一个数据值；而变量所需存贮空间的保持或释放，则由编译程序或解释程序来安排。

除了非常简单的计算机程序外，变量对于所有的程序都是不可缺少的。因为，如果不采用变量的话，就得对所处理的每组不同数据写一个新的程序。因此，每个月就都需要写一个新的程序，来对同一存折进行结算。

变量取一个新的数值的意思，就是对分配给变量的存贮单元存放一个新值，以取代原来的值。

在大部分语言中，一个具体的变量只能存放一种类型的数据（实型数、整型数或字符串）。有些语言要求使用者在使用变量之前，要预先说明你所使用的变量及它们的类型；有些语言在程序员未指定变量类型时，则根据变量名的第 1 个或最后一个字母而给出变量的缺省类型（假定类型）。

有些语言对所使用的变量类型及如何说明、如何使用有详尽的规定，这类语言称为强类型语言，如Pascal 就是这样的语言。而有些语言对变量的类型及说明并不严格，这类语言称为松散型语言，如 BASIC 语言。它们各有优缺点，简单地说，松散类型语言对编程要求比较松，而强类型语言则可以避免一些不必要的错误。

§2-2 赋值语句

赋值语句用于给变量赋以具体值。大部分语言用称之为赋值运算符的符号来表示这种赋值的操作。多数语言用等号“=”或冒号后跟一个等号“:=”作为赋值运算符。举例来说，如果要把5.1 赋给实型变量X，在Pascal 语言中可写成：

```
X :=5.1;
```

而在 BASIC 中可写成：

LET X=5.1

或简单地写成：

X=5.1

Pascal 和其它某些语言不采用简单的等号“=”来作为赋值运算符，其理由是因为赋值运算符并不表示真正的相等关系，例如，如果把X加1，其结果仍存放在X中，在 BASIC 和 FORTRAN 语言中可写成

X=X+1

但是，X怎么会等于X+1呢?当然是不会的。所以，不论采用什么符号，都最好把赋值运算符念成“被赋于…的值”。因此，上式可以读作“X被赋于（具有）X+1的值”，或者读作“X取X+1的值”。

§2-3 算术表达式

描述算术表达式最简便的办法就是举例来说明。首先，一个常量比如1.2就是一个算术表达式，数值变量（比如X）也可以是一个算术表达式。大多数高级语言中，算术表达式后面跟一个运算符（+、-等）、再跟上另一个算术表达式，仍然是一个算术表达式。例如，X+1、X-Y和2*X都是算术表达式（其中，*表示乘，/是除的运算符）。

算术表达式也可以包含有括号。重复使用上述规则，可以构成比较复杂的算术表达式。例如

(X-Y)*(X+Y)

2*(3*(X+4)/(Y-5))

算术表达式的主要特点是，按照一套给定的规则可以算出一个数值。

在算术表达式

X+2

中，X和2称作操作数，“+”称作运算符。大多数高级语言中，运算符在操作数之间，常常把这称作中缀表示法。少数高级语言例如 LISP 和Forth 采用另外的表示法。LISP 用前缀表示法（也称为剑桥波兰表示法），它是把运算符放在两个操作数之前。上面表达式在 LISP 中应该写成

(+X 2)，或（PLUS X2）

Forth 语言采用后缀表示法（也称作逆波兰表示法），这种表示法是两个操作数在前，后跟一个运算符。例如上表达式在Froth 中应写成

X2+.

“波兰”这个词是为纪念波兰数学家Jan Lukasiewicz （1878～1956），是他最先提出了前缀表示法的。

表达式是高级语言与低级语言的主要区别之一。低级语言通常一次只能做一件事，也就是说，一个语句只表示一个操作；而在高级语言中，程序员在一个语句中使用了算术表达式就可以完成许多计算。

§2－4 逻辑表达式

算术表达式是计算一个算术值，而逻辑表达式是计算逻辑值，即“真”与“假”。最普

通的逻辑表达式的形式含有关系运算符，如＞、＜、＝、＜＝、＞＝和＜＞（最后三个算符读作“小于或等于”、“大于或等于”和“不等于”）。因此

$$3 > 2$$

是逻辑表达式，其值是“真”。

如果变量X具有值2，变量Y有值3，则表达式

$$X + 2 = Y - 1$$

的结果是“假”。

高级语言还有逻辑运算符，如AND、OR和NOT。NOT算符把逻辑表达式的值从“真”变为“假”，或从“假”变为“真”。因此

NOT（3＞2）

的值为“假”，而

NOT（2＞3）

的值为“真”。

另二个逻辑运算符较难直接理解。首先，它们都有两个操作数（而NOT只有一个操作数），每个操作数都是一个逻辑表达式，一个操作数在运算符之前，另一个操作数则跟在运算符之后。这些运算符的作用可用下面的真值表来加以说明。表中A和B代表逻辑表达式。

A	B	A AND B
TRUE	TRUE	TRUE
TRUE	FALSE	FALSE
FALSE	TRUE	FALSE
FALSE	FALSE	FALSE

可看出，逻辑表达式A AND B的值，只有在A和B都为“真”时才为“真”。

下面的真值表是OR运算符的作用：

A	B	A OR B
TRUE	TRUE	TRUE
TRUE	FALSE	TRUE
FALSE	TRUE	TRUE
FALSE	FALSE	FALSE

因此，逻辑表达式A OR B的值，只在A和B都是“假”时才为“假”。

可以通过一些简单逻辑表达式的组合而得到比较复杂的逻辑表达式（建议采用括号以避免含混的可能性。）

§2-5 输入和输出

如果运算的结果既不能在显示屏上显示、也不能在打印机上打印出来的话，那么，你的处理工作并没有什么意义。这种功能对大多数语言来说，可以用WRITE或PRINT或其它类似的语句来实现，不管输出设备是显示屏还是打印机。