IBM PC BASIC 应用——

事务处理 图形学 人工智能

唐培顺 孔繁颖 王豫敏 编著

科海培训中心
一九八七年九月

前　言

BASIC语言现已成为应用最广泛的计算机程序设计语言之一，是进行事务处理、图形处理和人工智能等的有力工具。BASIC语言简单易学、功能较强，对计算机初学者和非计算机专业人员尤为实用。

随着IBM-PC系列机在国内的推广，便形成了众多的用户。为了满足广大用户对BASIC语言应用的要求，我们编著了这本书。其目的是介绍国内外用BASIC语言编制的各种计算机应用程序，从而可形成一个BASIC应用软件包。

本书共分为四章。第一章阐述IBM PC机上的BASIC语言的基础，它详细地介绍了BASIC语言的基本功能和命令以及一些比较简单的程序实例。第二章介绍使用BASIC语言编制事务处理软件的方法和程序，其中包括通用报表软件、统计程序、回归分析和预测技术、线性规划、以及物资管理等。在第三章中，我们比较全面地介绍了实用的图形学知识。其中既有BASIC作图的基本命令，又有象使用键盘和光笔之类的图形交互技术，动画技术和三维图形等比较复杂的技术。第四章主要是介绍使用BASIC编制的人工智能程序，包括通用问题求解、计算机博奕、专家系统、机器视觉、自然语言处理和机器学习等。它也是学习人工智能、知识工程或专家系统的入门教材和参考书。

本书是根据国内外的有关资料，结合我们的实践编著的，取材新颖、内容全面，对于熟练的BASIC程序员和初学BASIC的读者都有一定的参考价值。本书第二章和第四章由唐培顺编写，第三章由孔繁颖编写，第一章由唐培顺和王豫敏合写。

由于著者水平所限，加之成书匆忙，书中难免会有些疏漏和不当之处，敬请读者指正。

编著者

一九八七年九月

目　录

第一章　IBM PC BASIC 程序设计基础

§1.1　BASIC 的版本与启动

IBM PC 的工作原理，和其它的计算机一样，也是在程序的控制下运行的。用户需要计算机做些什么事情，首先应为它准备好一个程序。程序可以买现成的，也可以自己编写。通常是买一些较复杂的程序，而简单的程序则自己编写。

当为计算机编写程序时，必须采用计算机能懂得的语言。可惜计算机还不能很好地懂得人类的自然语言。所以，我们必须采用为编写计算机程序而专门设计的语言，这种语言就叫做程序设计语言。

IBM PC配置有三种版本的BASIC：盒式磁带BASIC，磁盘BASIC和高级 BASIC。盒式磁带BASIC可以用于任何一种IBM PC机。磁盘BASIC则要求计算机至少有一个软盘驱动器和一个32K的存储器。高级BASIC 则要求计算机至少有一个软盘驱动器和一个48K的存储器。

盒式磁带BASIC包含了主要的BASIC语言。磁盘BASIC不仅具有盒式磁带BASIC 的全部特点，而且还具有利用软盘存储程序和数据的功能。高级 BASIC 不仅具有磁盘BASIC的全部功能，而且还具有某些控制计算机的附加命令和语句。

本书前几节的绝大部分内容，对于盒式磁带 BASIC 都是适用的。只有在后面的几节，才介绍磁盘BASIC和高级BASIC的一些语言特点。能使用磁盘BASIC的人，或许一开始就想用磁盘，因为要保存他所写的程序，利用软盘要比用盒式磁带快得多，当然使用硬盘会更快。当需要利用高级BASIC 提供的附加功能时，就只能利用高级 BASIC。编译BASIC对初学者来说有一定的难度，故将在后面介绍。

盒式磁带BASIC 的解释程序已经装入计算机的只读存储器中，只要一开机，它就可以使用。这种解释程序让计算机准备接收BASIC语言的命令和程序，它是用机器语言 编写的一种程序。

启动磁盘BASIC或高级BASIC的方法如下步骤：

（1）把DOS软盘插入驱动器A中。这个软盘中有要装入的磁盘BASIC或高级BASIC程序。

（2）打开计算机的电源开关。在计算机自检之后，就自动地把 DOS 软盘中的磁盘操作系统装入机器。

（3）输入日期和时间。

（4）显示出所需BASIC版本的信息和提示符A>。

其中A，代表驱动器A是缺省的驱动器名，当不指明驱动器名时，就表明是指驱动器A。>表示操作系统正在等待接受命令　如果想要使用磁盘 BASIC，就打入 BASIC，并按下Enter 键；如果想要使用高级 BASIC，就打入 BASICA，再按下 Enter 键。打入BASIC或BASICA，可以用大写，也可以用小写。

计算机显示的宽度是指每行所能显示的最多字符数。在IBM PC 上能设置的宽度为40或80个字符。这是通过命令　WIDTH 40↵　或　WIDTH 80↵来实现的。

§1.2 常数、变量和输入输出

1.2.1 数据类型和常数

计算机所处理的数据项，可以是一些数字和文字。我们把这些数据项看作数值，并且把它们按数据的类型分类。属于不同数据类型的数值在计算机中存放的地方是不同的，而且处理时的操作也不同。在BASIC程序中，可以用常数来表示数值，例如100和“北京”。

IBM PC 的 BASIC，有两种主要的数据类型——字符串和数。数可以进一步分为整数，单精度数和双精度数。

字符串就是一串字符。数字、英文字符或汉字皆可，标点符号和一些特殊的符号亦可。在一个字符串中，所允许的字符数最多为 255 个。字符串中也允许包括零字符。没有字符的字符串称为零字符串（或空字符串），它和算术中的零一样，是非常有用的。一个字符串常数，是一个放在引号里面的字符串。引号不属于字符串的一部分，它们只用于告诉计算机，字符串常数的起点和终点。例如：

“Have a nice day。”

“2.71828”

数的表示方法在BASIC中和我们日常所用的是相同的。下面是 BASIC 中数值常数的几个例子：

235　-510　1.2345　6.3333

有一点不同的地方，就是数值常数中不能用逗号。在BASIC 中，逗号常用于把一串数中的项与项分开。例如 5，342，749是表示 5，342和749等三个数。

在IBM PC 的 BASIC 中，数可分为三种类型：整数，单精度数，和双精度数。单精度数是以 7 位数字的精度存储在计算机中的，而双精度数是以17位数字的精度存储的。

用 8 位数字以上表示的常数是双精度数，下面是一些例子：

3.1415927　2.7182818　37980000

在常数的后面加上一个 # 符号，就把它变为双精度数：

25 #　2.618 #　-5.23 #　800 #

如果不加 # 符号，中间两个是单精度数，而前后两个是整数。

1.2.2 BASIC中的浮点数表示

BASIC 可以采用浮点记数法，这种记数法可以省去写出一长串的零。我们先用例子来说明这种记数法。假设有一个数 3.25E4，E 右边的 4 表示把小数点向右移 4 位，并根据需要添上一些零。这样，3.25E 4 就变成 32500。同样，可以 把 7.85E2 变 为 785；把 8.7E 3 变为8700和把2.16 4 E 2 变为216.4。

如果E的右边是一个负数，则小数点应向左移位，而不是向右移位。这样，2.3E - 2 就变为.023；5.61E-3就变为.00561；和6.3E - 1就变成.63。

带有字母E的浮点常数表示一个单精度数，如果用 D代替 E，就是双精度数（整常数不能采用浮点记数法）。因此，6.82E3是单精度常数，而6.82D3 是双精度常数。当用浮点记数法打印一个数时，如果是一个单精度数，计算机就用 E；而如果是一个双精度数，就

用D。

当传统的记数法占用太多的空间时，BASIC 就自动地用浮点记数法来显示数值，例如：

```
PRINT  10000000000000000000
1D+20
OK
```

当列出 BASIC 程序清单时，程序中某些数的表示方法可能与原来所写的格式不同。数字零如果不是有效数字（例如003或 2.50 中的零）的话，那末就会被删掉，也可能在某个数的末端加上一个感叹号或 # 符号；或者把原先以传统记数法输入的数变成浮点数。BASIC 以特殊的代码变为打印的字符，而 BASIC 所选择的数的打印格式，可能与该数原先输入的格式不同。

1.2.3 算术运算

利用计算机的存储器来存放中间结果，就可以进行任何计算。然而，如果只限于使用包括一个运算符和两个操作数的表达式，则计算起来就比较麻烦，不过，要是利用多个运算符的表达式，计算起来往往是比较方便的，例如：

```
PRINT 3+7+9+2
21
OK
PRINT  25-8-2-1
14
OK
```

在这种情况下，操作是从左向右进行的，与我们读表达式时，所遇到的运算符的顺序一样。在上面第一个例子中，无论相加的顺序如何，总数是相同的。第二个例子中，如果减法运算是从右向左，而不是从左向右，则结果就成为18而不是14。

我们还可以写出带有不同运算符的表达式，例如 3 + 5 * 2 。可以认为操作是从左向右进行的。实际上，有一些程序设计语言，就是以这种方法来计算表达式的。绝大多数的程序设计语言，都采用稍为复杂一点的方法，这是数学家设计的一种方法，使常用的表达式写起来容易一些。在这种方法中，运算符各有自己的优先权级别。在计算表达式时，计算机首先执行最高优先权的操作，然后执行较低优先权的操作。而同等优先权的操作，则采取自左向右的顺序进行计算。

在IBM PC 的BASIC中，运算符和操作的优先权顺序如下所示：

∧	取幂
-	负号
*和/	乘和除
\	整除（商）
MOD	整除（余数）
+和-	加和减

如果把表达式中的某一部分用括号括起来，则在括号中的那部分就要比表达式的其余

部分先得到处理。括号使我们在必要的时候，可以超越运算符的优先权。例如，假设我们要使7和3相加，然后再把结果乘4。如果没有括号，就会有困难。

```
PRINT  7 + 3 * 4
19
OK
PRINT  ( 7 + 3 ) * 4
40
OK
```

我们还可以利用括号，来使具有同样优先权的运算符超越自左向右的顺序，例如：

```
PRINT  9 - ( 3 + 2 )
4
OK
PRINT  12 / ( 4 * 3 )
1
OK
```

1.2.4 变量及其名称

我们把在整个程序执行过程中，数值保持不变的项称为常数；而把某一个存储区称为变量，因为在运行程序的过程中，存放在那个区域的数值是可变的（计算机能把不同的数值存入存储区）。这种存储区的名称是一个变量的名称，而存放在它里面的数值就是变量。

BASIC可以相当自由地选择变量的名称，用来表示所存储数值的意义。为了使BASIC能把变量名称与其它的语言要素区分开来，在这里对选择变量名称作了一些限制。

（1）变量名称是由字母组成的，其中可以包括数字0～9和小数点。小数点用于把变量名称中的字和字分隔开。在变量名称中不允许有空格和短划线，例如：

```
AMOUNT.DUE  GROSS.WAGES  INVADERS.DS
WORD.COUNT  SUBTOTAL.5
```

其中的XQ35 B虽然是允许的，但不是一种好的选择，因为它没能表示出所存数值的意义。

（2）一个变量的名称不允许超出40个字符的长度。请注意这一规则在BASIC程序使用手册中的说明有错误。在手册中说变量名称的长度是任意的，而前面的40个字符可用于区别变量的名称；但实际上，当名字的长度大于40个字符时，就算是一种语法上的错误。

（3）变量名称必需以字母开头。数字和小数点只能用于变量名称中的其它地方，而不能作为变量名称的第一个字符。这个规则使计算机能辨别出是变量名称还是数值的常数。没有这一规则，100.05、23E5和5.3D4等就全都成为有效的变量名称，便会引起混乱。

（4）变量名称不应当是BASIC的保留字。因为保留字的最好方法是利用与当前应用有关的专用变量名称。因为保留字都只和BASIC语言的特性有关，而不会与某些专门

的应用有关。

（5）变量名称不能以字母FN起头，因为FN是用户自定义函数名称的前缀，这个问题将在以后讨论。

BASIC的变量是不需要事先予以说明的。所以要注意不能使变量重名，因为这样可能会导致逻辑上的错误。语法错误可由BASIC的解释程序或编译程序发现，而逻辑错误只能通过对程序运行的测试及结果分析才能发现。

1.2.5 数据类型

我们必须把要存放在存储器中每个区域的数据类型告诉计算机，使计算机能选择一个大小合适的存储区。利用下列的类型说明符加在变量末端，倒是一种说明的方法。

类型说明符	数据类型
$	字符串
%	整　数
!	单精度
#	双精度

这样，EMPLOYEE.NAME$就是一个字符串变量，而且只能用于存放字符串；而PAGE，COUNT%就是一个整型变量，只能用于存放整数值；而MONTHLY.WAGES!只能用于存储单精度数值；NATIONAL.DEBT#只能用于存储双精度数值。

除了特别指明的情况之外，变量的名称如果没有用类型说明符结尾，就被认为是单精度的变量。因为单精度的数一般已能满足需要，所以常把它的类型说明符省略掉。

1.2.6 赋值语句LET

LET语句把一个数值存入存储器中与某一变量相对应的存储区。存入存储器中的数值就变成变量的数值。因此，LET语句能把一个新值赋给变量，我们把LET语句称为赋值语句。

LET语句的一般格式如下：

LET<变量>=<表达式>

赋值给一个变量不是永久性的。利用后续的赋值语句，还可以把新的数值赋给同一个变量。当把一个新的数值赋给变量时，这个新的数值就取代了以前的数值。以前的变量数值就消失了。

```
LET GREETING$  = "早上好"
OK
PRINT GREETING$
早上好
OK
LET GREETING$ = "下午好"
OK
PRINT GREETING$
下午好
```

```
OK
```

由于赋值语句是很常用的，故BASIC允许把语句中的关键字LET省略掉：

```
LABEL$ = "Sales Tax:"
OK
SALES.TAX = 200 * .04
OK
PRINT LABEL$; SALES.TAX
Sales Tax:  8
OK
```

1.2.7 输入语句INPUT

INPUT语句是程序要求使用计算机的人输入数据的语句。当执行INPUT 语句时，程序的要求就显示在屏幕上。用户就打入所要输入的数值作为响应，然后按下Enter键。用户所打入的数值，就赋给INPUT语句中指定的变量。例如：

```
INPUT "请输入姓名："; NAME$
```

利用一个INPUT语句，我们可以把任意多个数值赋给多个变量。在INPUT 语句中这些变量是用逗号隔开的。用户打入的数值也必须用逗号互相隔开。例如：

```
INPUT "长，宽，高："; LENGTH, WIDTH, HEIGHT
```

1.2.8 行输入语句LINE INPUT

LINE INPUT 语句能把一个完整的打印行赋给一个字符串变量。输入的内容 可 以由ASCII 代码表中32—126或128—255范围内的任何字符所组成。这个范围包括了通常打字机上所有的字符。字符串可以带引号，而且能保留字符串开头的空格；但是，字符串末端的空格被删掉。

LINE INPUT 语句不能自动显示出问号。它可以带一个提示用的字符串，而且 这个字符串可以包括一个问号。例如

```
LINE INPUT "请输入一行："; A$
```

1.2.9 输出语句PRINT和LPRINT

PRINT语句与LPRINT语句的区别在于，PRINT是输出打印在屏幕上，而 LPRINT是输出打印到打印机（当然，这时的打印机必须是准备好状态，否则就会出 错）。其它格式一样。涉及到LPRINT输出时，在调试程序时可以使用PRINT 语句，正式运行再换成LPRINT。

PRINT和LPRINT 这两个关键字的后面，都可以跟一个要打印的项目清单。清单中的各个项目可以用分号或逗号隔开。各个打印项目之间的间隔，取决于所用的标点符号。

如果用分号来分隔各个项目，则打印出来的各个项目是连接在一起的，没有用间隔把各个项目隔开来，否则可以用逗号。例如：

```
PRINT "A"; "B"; "C"; 15
ABC  15
```

```
OK
PRINT -1, -2, -3
-1          -2          -3
OK
```

通常，当BASIC执行完一个PRINT语句，它就进入另一新行。下一条要执行的PRINT语句的输出，就将出现在这一新的行上。

为了不让计算机进入一新行，可以在PRINT语句的末端加上一个分号或逗号。当使用一个分号作为PRINT语句的结尾时，下一条PRINT语句产生的输出，就会紧跟在当前这条PRINT语句输出的后面。

BASIC还提供了格式化输出语句PRINT USING和LPRINT USING，将在下面讲述。

§1.3 循环与条件语句

我们不仅应该注意程序中的语句，而且还要注意计算机执行语句的顺序。另外，有时候要求计算机重复执行某些语句，和只在某种条件下才去执行别的一些语句。在安排语句的执行过程时，我们有三种组织程序的方法，称为控制结构。这三种控制结构就是顺序结构、循环结构，和条件语句。

在顺序结构中，执行BASIC的语句是按照语句在程序中出现的顺序进行的，即按照它们的行号的顺序进行的。

在循环结构中，语句是重复执行的，它可以执行一定的次数，也可以在某种条件为真时，就一直不断地继续执行。

在条件语句中，可以根据程序运行时所出现的条件，来选择和执行不同的语句。条件语句使程序能够根据不同的输入数据，或对不同命令的响应，使计算机具有判断的能力。

要求不止一次地执行程序的语句，是完全应该的。大家知道，编写一个程序语句就和利用计算器来执行一个程序语句差不多，如果每个程序语句写好之后，只是执行一次，那么利用手算的方法就更简单一些，不需要用计算机来执行所编写的程序了。

当然，我们可以一次又一次地运行我们所编写的程序语句，但我们这里所讨论的循环，是指在某一次运行整个程序时，要重复执行程序中的某些语句。

IBM PC的BASIC中，有两对控制循环的语句，一对是FOR和NEXT语句；而另一对是WHILE和WEND语句，条件语句是IF和THEN。

1·3·1 FOR和NEXT语句

让我们先来看一个例子：

```
10 FOR COUNT=1 TO 3
20 PRINT COUNT
30 NEXT COUNT
RUN
 1
 2
```

```
 3
 OK
```

其中 PRINT COUNT 语句就执行了三次。当第一次执行 PRINT COUNT 时，COUNT数值是1，第二次的数值是2；第三次的数值是3。

跟在 FOR 和 NEXT 后面的变量名(在这里是COUNT)称为控制变量。FOR语句所指定的数值，就是要赋给控制变量的数值——1、2和3。在FOK 语句和NEXT语句之间的所有的语句，对每一个控制变量的数值，都执行一遍。循环语句能利用控制变量的数值，来区别这一次重复和下一次重复，并对每一次重复采取不同的操作。例如，PRINT语句每次重复时，显示出不同的数值。

在FOR语句中，加上一个STEP分句，可用于说明控制变量每重复一次时的数值有什么变化。例如下面的程序：

```
10  FOR I = 1  TO  9  STEP  2
20  PRINT I;
30  NEXT I
```

使计算机每次把数值加2(从1开始)

```
RUN
1   3   5   7   9
```

而程序

```
10  FOR I=10 TO 22 STEP  3
20  PRINT I;
30  NEXT I
```

使计算机每次把数值加3(从10开始)

```
RUN
10  13  16  19  22
```

我们也可以使数值递减

```
10  FOR I=10 TO 0  STEP  - 1
20  PRINT I;
30  NEXT I
RUN
10  9  8  7  6  5  4  3  2  1  0
OK
```

当取消STEP分句，计算机就认为步长(STEP)是1。

控制变量可以是整数，也可以是单精度变量(但不能是双精度变量)。如果赋给控制变量的数值不是整数，就必须是单精度的变量。如果只用整数值来作为控制变量，则既可以用整数，也可以用单精度数。在可能的情况下，最好使用整数变量，这样可以节省计算机处理控制变量所花的时间，从而使循环的过程，运行得快一些。

FOR 语句的一般格式为：

FOR〈变量〉=〈初值〉TO〈终值〉STEP〈步长值〉

其中的初值，终值，和步长值，全都可以是变量，表达式，或者常数，例如：

```
10  L = 6
20  M = 2
30  FOR I=L-M  TO  L+M  STEP  M
40  PRINT I;
50  NEXT I
RUN
 4    6    8
OK
```

其中I的初值是4（即L－M的值）；I的终值是8（即L＋M的值），而步长值是2）即M的值）

在NEXT语句中，可以删掉控制变量的名称，这样，就可以把前面的

```
50  NEXT I
```

改写为：

```
50  NEXT
```

1.3.2 WHILE和WEND语句

利用FOR和NEXT语句，我们可以告诉计算机所要循环的次数。但是，有时候我们事先不知道应该循环的次数，只要求在某种条件保持为真时，就继续循环。WHILE和WEND语句就可以用于控制这种循环。

WHILE和WEND语句的结构如下：

```
WHILE〈条件〉
        〈循环语句〉
WEND
```

其中的〈条件〉是一个说明语句，这个说明语句所说明的是程序中所用的一个变量的数值。这个说明语句是真还是假，就取决于变量的数值。其中的〈循环语句〉是指WHILE语句和WEND语句当中的所有的语句。在执行循环语句之前，计算机要检测一下条件为真还是假。如果条件是假的，则不执行循环语句，且从此结束循环的执行。如果条件为真，则执行循环语句，然后计算机再次检测条件，看看是否必需再执行循环语句，如此继续下去。

为了比较常数、变量和表达式的值，我们采用六种关系运算符来构成条件。其表示方法与意义如下：

关系运算符	意义
=	等于
<	小于
>	大于
<=	小于或等于
>=	大于或等于
<>	不等

关系运算符也可以用于字符串的情形，但意义稍有不同。其大小顺序是按字典顺序由小到大排列的。

使用 WHILE 和 WEND 语句，要比用 FOR 和 NEXT 语句稍为复杂一些。所以，当已知循环次数时，就选用 FOR 和 NEXT 语句，只有在事先不知道循环次数的情况下，才采用 WHILE 和 WEND 语句。例如，要计算其五次方大于 10000 的最小整数，程序如下：

```
10  M = 1 : N = 1
20  WHILE  N<=10000
30  M = M + 1
40  N = M^5
50  WEND
60  PRINT  M
```

1.3.3 READ 和 DATA 语句

READ 语句可从 BASIC 程序中的 DATA 语句读取数据。DATA 语句有两个主要的用途。

首先，有些程序常常要求输入常数值的表格，并且在程序的执行过程中，要访问这些表格。在程序执行过程中要求用户打入一个表格不是我们现在要讨论的问题，但是从盒式磁带或软盘中读出一个表格还是比较花时间的。把必要的表格放在 DATA 语句中，作为程序中的一个永久性部分是个好方法。

其次，在数据处理时，READ 和 DATA 语句是一种常用的语句，例如用于读记录。记录通常是属于某个学生，公司的职员，或者存货清单上的项目（一个记录是个有关的数据项的集合，譬如说属于某个职员或学生的全部数据）。记录通常以文件的形式存在盒式磁带或软盘中。但是，如果为了试验一个短的程序，而去建立盒式磁带和软盘的文件，是不太方便的。作为数据处理的初次尝试，我们把要试验的文件，放在 DATA 语句中。

与 INPUT 语句一样，READ 语句可以包括一个要赋值的变量清单。变量的数值可以从 DATA 语句中读得

```
10  READ A, B$
20  DATA 25, Computer
30  PRINT A; B$
RUN
25  Computer
OK
```

当计算机执行 READ 语句时，它从 DATA 语句中读出两个数值。第一个数值是 25，它被赋给A，第二个数值是Computer，它被赋值给B$。

与响应 INPUT 语句时，打入的字符串一样来处理 DATA 语句中的字符串所加的引号。如果字符串不加引号，则字符串前后的空格就被忽略掉，并且字符串中不允许带有逗号或分号；如果不容许有这些限制条件，则字符串必须放在引号当中。

DATA 语句可以在程序中的任何地方出现。计算机在执行 READ 语句时，就去