CBASIC 语 言

四 川 省 地 震 局 印
一九八四年五月

译　序

本书译自CBASIC语言第二版，该语言是在CP/M或CP/M—86操作系统支持下的编译型BASIC语言。该种语言的特点是效率高、速度快，是广大计算机使用人员最常用的一种算法语言。

CBASIC语言第二版在功能上有一些扩展，增加了整型变量、多行函数、链接和公用变量……等，还包括IF…THEN…ELSE和WHILE结构以及磁盘文件存取等，为了给程序设计人员和广大科技工作者使用计算机提供方便，四川省计算机应用研究中心李明、何小钧同志翻译了此本资料，以供大家使用时参考。

由于译者水平有限，译文中会有不妥，以至错误之处，恳请读者批评指正。

编　者

一九八四年三月于成都

目　录

第一章 CBASIC

1.1 引 言

这本手册描述的是CBASIC的第二版，CBASIC第二版在CP／M操作系统支持下可作为编译程序，也可做为解释程序使用。CP／M操作系统是数字研究公司的商标。

CBASIC第二版有一些扩展变化，其特点包括IF…THEN…ELSE和WHILE结构以及磁盘文件存取。CBASIC还允许使用31个字符的变量名，自由使用说明，空格，表。这些都有助于你建立一个可保存的程序文件。

CBASIC第二版增加了整型变量，多行函数，链接和公用变量，并且还增加了预定义函数以及其它方面的功能。还能提供一个交叉参考表。

CBASIC系统由三个程序组成。第一个程序是编译程序，编译程序将用户的源语言程序翻译为放置在磁盘上的中间代码文件。第二个程序是运行监控程序（The run—time monitor），它直接地执行由编译程序产生的中间代码文件。最后一个程序是XREF·COM程序，它产生一个在CBASIC源程序里所有被使用的变量的交叉参考表。

使用CBASIC的微型计算机系统必须使用CP/M操作系统。这本手册中假设你已经具备以下三个CP／M文件的知识：

（a）CP／M特点和性能介绍

（b）ED：CP／M磁盘系统的文本编辑程序

（c）CP／M接口指南

这些手册可取自数字研究公司（Digital Research），PO BOX 759, Pacific Grove, california。

一个新接触计算机领域的人最好应读一本介绍BASIC语言的入门书。

本书的第二章至十章，描述CBASIC语言的功能。十一章至十三章提供一些比较进一步的处理方法。最后是三个附录。

1.2 CBASIC第一版程序员须知

熟悉CBASIC第一版的程序员在回顾第一版CBASIC手册时，特别会注意到整型变量的使用。在第二版手册中的第三章里详细的提供了整型表达式的使用说明。这部分涉及到新语句和函数，应当详细地阅读。十二章和十三章还包括了非常新的信息。

尽管在一般情况下，用CBASIC第一版编译并执行的程序应该适合CBASIC第二版的

操作。但实际由CBASIC第一版编译建立的INT文件不能用第二版的运行监控程序执行。源程序必须重新编译，如果出现的语句不是按严格的操作，系统软件将默认为是注释，其中包括一条语句或由这条语句引起的多条语句。

1.3 程序文本号说明

所有的系统软件在程序名后面都跟着一个文本号。其格式如下：

V.CR

“V”是版本号。本手册描述的CBASIC语言是第二版。

“R”是程序的发行号。比如像对某一版本中的错误进行校正，校正后可按新的发行号标注。

“C”是指程序在内存里的排布。“C”如果为0，则表示程序是在标准CP／M支持下运行；TPA（内存暂存区）是从100H开始。如果为1，则意味着程序运行在Radio Shack TRS—80的CP／M操作系统下。

第二章 概 述

2.1 语 句

一个程序由一条或多条规定格式的CBASIC 语句组成。也称为源语句。如果出现END 语句，则表明程序结束。此时，END 语句后面跟随的语句将不被理睬。

全体ASCII码字符均可以接受，不过仅用其中的64个字符便能写出所有的语句。小写字母除出现在字符串里和备注里的外，其余都由编译程序转换为大写字母。编译开关用来限制上述的小写字母到大写字母的转换。它将在十三章里介绍。

CBASIC 语句基本上采用自由格式，但有下列的要求：

(1)当一条语句在一行里不能结束时，必须使用继续行字符(\)。这样可在下一行继续写这条语句。CBASIC保留字，变量名，字符串常数不能从中间打断继续到下一行。

(2)在同一行中，跟随继续行字符后面的所有字符将被编译程序忽略。

(3)允许在一行中出现多条语句，但其间必须用冒号隔开。DATA、DEF、DIM 和END这四条语句出现的行里不允许其它语句出现。IF语句必须是一行中的第一条语句。以上的规则对于REM语句是例外的（参看2.4节）。

语句前面可以有空格；在可允许空格的地方可以出现任意多的空格。为加强程序的易读性而对语句位置改变所增加的附加空间，在编译时不予考虑。即不会增加中间代码文件所占用的空间。

2.2 表示法

在这本手册里描述的所有CBASIC 语句。介绍一条语句便给出了语句的一般格式。对于每条语句的一般格式采用下述的表示方法。

保留字和符号

所有特定的字符和以大写字母出现的符号在语言里有特定意义。例如，READ、REM和 PRINT 是 CBASIC 语言的保留字。附录C 是一个 CBASIC 所采用的保留字的详细清单。

尖括号〈 〉

尖括号中包含了一个在程序文本中必须详细定义的项。

方括号［ ］

方括号表示其中的内容为任选项。

花括号{ }

花括号中的部分可以出现零次或许多次。

2.3 行 号

在CBASIC中，行号是可以任选的。除在GOTO，GOSUB，ON，IF等语句给出的行号外，其余的语句中出现的行号编译程序将不予考虑。在用到上述几种语句时，这些语句给出的行号作为其它语句的标号只能是出现在程序中的某一个语句之前，两行中的两个语句不能同用一个行号。使用行号不必按从小到大的顺序排列。

例如：

```
40    INPUT    ITEM
20    PRINT    ITEM
30    GOTO      40
```

在这段程序里，20和30是不要求的；在编译它们时不予考虑。行号40在GOTO语句中出现，则行号40必须在程序中出现一次（作为其它语句的标号），且仅一次。行号可以由若干位数字组成，但仅有前31个数字编译程序给予区分，认为是有意义的。

另外，CBASIC的行号有一个特点，即任何有效的数均可以用做行号。并且允许使用非整型数作为行号。写一个主程序或者子程序，其中所用的行号可以是两个连续整数之间的任意十进制小数，这都是CBASIC语言所允许的。

行号甚至可以是以指数（E）的形式出现。当你在写另一个程序的子过程时，这是一个方便的特性，因为这将有助于确保行号的唯一性。

下面列出的行号在CBASIC中均是有效的：

```
1
0
100
100.0
100.123
100E21
```

100和100.0这两个行号被编译程序处理为不同的行号。换言之，实际上用来决定行号的不同是取决于字符串本身而不是其实际数值。

2.4 REM 语句

[〈行号〉] REM [〈以回车（Carriage Return）结尾的字符串〉]

[〈行号〉] REMARK [〈以回车（CR）结尾的字符串〉]

REM语句是不被编译程序考虑的。REM语句可用来说明一个程序。REM语句不影响可编译或执行的程序所占的存贮空间。一条没有标号的REM 语句可以出现在同一行里的任何其它语句的后面。REM 的行号可以使用在 GOTO，GOSUB，IF，ON 等语句中。

REM 语句举例如下：

```
REM THIS IS A REMARK
remark This is also a remark
tax = 0.15 * income   rem lowest tax rate
```

最后一个例子表示REM语句可以同其它语句在同一行里出现。按照这种方式使用 REM 语句时，两个语句之间不需要用冒号隔开。除此之外，在一行中出现多条语句时，都要用冒号隔开。另外，如果 REM 语句和其它的语句在同一行里时，REM语句必须是这一行里的最后一条语句。

2.5 执行一个 CBASIC 程序

一个CBASIC程序的执行分为三步。第一步是必须将源程序建立在磁盘上。第二步是执行CBASIC的编译程序，将磁盘上建立的源程序文件进行编译。第三步是从磁盘上调出运行监控程序（The run-time montor）执行由编译程序在磁盘上建立的中间代码（INT）文件。此时所用的中间代码程序的文件名与源程序名相同。

源程序可用CP／M的文本编辑程序建立。源程序必须以BAS为扩展名。源程序的每一行以回车换行终止。行的长度可以是任意的，但编译程序仅列出每一行的前80个字符。在打入源程序时，标识符（变量名，保留字，用户定义的函数名）必须正确拼写，可以用非数字、字母字符作为分隔符。一般说来，空格可以用来作为标识符的界限。标识符里的所有字母由编译程序转换为大写字母，除非在编译时采用编译开关对转换进行了限制（见第十三章）。

CBASIC 不允许保留字与其它标识符连接在一起，这是CBASIC与很多BASIC语言的一个区别。例如：

```
READA
```

这条语句编译程序是不认识的。它必须写成：

```
READ A
```

CBASIC的编译程序按如下格式调入内存运行：

CBAS2〈文件名〉［〈磁盘号〉］［$〈开关（toggle）〉｛〈开关〉｝］

这里文件名是源程序的名字。编译程序假定它的文件类型为BAS。编译开关的前面有一个‘$’符号，它可以跟在文件名后面。编译开关将在第十三章讨论。

编译程序产生一个由CBASIC 机器语言构成的中间代码文件，它的文件名仍和源文件相同，不同的只是扩展名为INT。这个 INT 文件一般是放在源程序所在的磁盘上。当程序员希望INT 文件放到指定磁盘上时，编译命令的磁盘号则可以选用。当它出现时格式为A：，B：，等等。

下面的命令将编译一个文件名为INVENTORY.BAS 的 CBASIC 语言源程序。源程序是放在当前驱动器上的，编译后生成的INT文件放在B驱动器上。

```
CBAS2 INVENTORY B:
```

如果选择了列表开关（第 13.2 节），那么跟随在 CBAS2 后面到‘$’字符之间出现的程序名以及其它字符在命令执行时将会出现在每页程序清单的开始。

例如：

```
CBAS2 COST.BAS ON 7 NOVEMBER 1979 $EBF
```

在运行开始时将出现：

```
CBASIC COMPILATION OF COST.BAS ON 7 NOVEMBER 1979
```

这个源程序正常地列在输出设备上，其中包括所有出错信息（看第 13.3 节）。如果在编译期间检查出错误，则源程序必须用文本编辑程序进行修改。附录A中列出了所有的出错信息。然后将修改的程序重新编译。如果在编译时没有错误，则产生的中间代码文件可以按如下命令格式执行：

CRUN2〈文件名〉［TRACE［〈行号1〉［，〈行号2〉］］］［〈命令〉］

跟踪（TRACE）选择在第十三章介绍，命令域是在带有 COMMAND$ 预定义函数

时使用，它们将在第七章讨论。

在执行期间如果发现错误，源程序必须进行修改并重新编译。附录B列出了所有的错误代码。

第三章　表达式格式

这一章讨论表达式格式。首先描述表达式的元素，常数及变量。然后将这些单元联结在一起便形成了表达式。表达式是一种基本的积木块，被用在许多CBASIC语句中。

3.1　字符串

字符串常数定义为0个或多个被引号（"）括住的有效的字母、数字字符。因为继续行标志（\）被认为是字符串的一部分，故定义为常数的字符串在源程序里必须出现在同一行里。回车换行不属于字符串。置入字符串的引号作为两个相邻引号引入。

下面是一些有效常数串的例子：

"123"

"May 24, 1944"

"Enter your name please"

""Look, look,"" said Tom"

最后一个例子是代表：

"Look, look," said Tom

3.2　数

CBASIC提供了两种类型的数值量，整型和实型。实型常数可以是固定格式也可以是指数格式。在这两种情况下，它可以包含1到14个数字，一个符号和一个小数点。在指数表示法，指数按照"Esdd"格式。这里 's' 如果出现，表示有效符号（+，－，或空格）。 'dd' 表示一位或两位有效数字。符号 's' 是指数的符号，不要和数的正负符号相混淆。数值范围从1.0E－64到9.9999999999999E62。虽然CBASIC语言仅给出了14位有效数字，但在一个实常数的内部表示中包含有更多位数字。最后被截取至14位

有效数字。

实型数占了内存8个字节。第一个字节是符号和指数。指数值至多到64。剩余七个字节是规格化的数字部分。最前面那个字节高四位必须代表有意义的十进制数。

如果一个常数里没有小数点，也不是采用指数标位法，范围是－32768到＋32767，那么可将这个常数处理为整型数。整型数占两个存储单元。

整型常数也可以用十六进制和二进制数表示。如果常数的末尾是H，则是一个十六进制常数。如果是以B结尾，则是一个二进制常数。十六进制数的第一个数必须为数字例如：255写成十六进制为0FFH，而不是FFH。

二进制和十六进制常数不包含有小数点。对任意特定的数用十六位二进制数保存。

在这本手册里，实型数与浮点数被认为是相同的。数这个称呼既可以是指实型量也可以是指整型量。有效数字举例：

1，1.0，－99，123456.789

1.993，.01，4E12，1.77E－9

1.5E+3等价于　1500.0

1.5E－3等价于 .0015

1ab0H，　10111110B，0FFFFH

3.3 标识符

标识符的第一个字符必须是字母，随后可跟若干个字母或数字或句点。但在区别不同标识符时，仅前面的31个字符予以考虑。标识符在程序中可用作变量名。若标识符的最末一个字符是‘$’，则这标识符是字符串类型。如果标识符以百分号(%)结尾，则类型为整型变量。除此之外为实型变量。

出现在标识符里的所有小写字母，只要不用编译开关D加以限制(第十三章)，全部将转换为大写字母。标识符里使用句点，可以增加程序的可读性。例如，BAD.DEBT％比BADDEBT％的意义更加明确。

使用长于两个字符的标识符可以增加程序的易读性，并不会增加由编译程序产生的中间代码文件的体积。

有效标识符举例：

A，B$，C1，C1234%

```
Payroll.Record, NEW.SUM.AMT

INDEX%, FLAG.3%, counter%

ANSWER$, file.name$, CUSTOMER.ADDRESS$
```

3.4 变量与下标变量

变量的一般格式：

〈标识符〉［（〈下标〉）］

下标的一般格式为：

〈表达式〉{，〈表达式〉}

下标表达式类型必须为数值型。如果下标表达式是整型，则存取数组元素将会非常的快。如果下标表达式是实型，则按它最接近的整型值作为下标值。如果下标表达式是字符串类型，则会发生错误。“下标”既指定了对应的下标变量，同时也指定了该数组中相应的一个元素。

在程序执行期间，每一个变量都具体与一个值相联系。在初始状态下，数值应为0，字符串应为空串（null string）。一个字符串变量没有一个固定长度预配给它。由此，不同的字符串赋值给字符串变量时，存储是动态分配的。可以赋值给字符串变量的最大长度为255个字符。

一个变量不要以 FN 开头。因为这样的标识符专门用于用户定义函数的。见第八章。

CBASLC语言中可以出现的变量类型有整型，实型，字符串型。

例如下列变量：

```
X$
PAYMENT
day·of·deposit%
```

下面例子为下标变量：

```
Y$(I%, J%)
```

```
COST(3,5)

POS%(XAXIS% , YAXIS%)

INCOME(AMT(CLIENT%) , CURRENT.MONTH%)
```

当计算下标时，检查下标是否已超出数组定义的范围。如果超出了数组的范围，则出现运行错误。一个程序里的下标变量在使用之前，必须用DIM语句定义它的维数。

DIM语句是一条可执行语句；每次执行时都将分配一个新的数组。若这个数组以前用来攻过数据，则在重新为新数组分配空间之前，原来的那个数组要先被删去。若原数组是字符串型，为了收回最大空间，在重新执行DIM语句之前，每个元素须置为空串。DIM语句的一般格式为：

[〈行号〉] DIM 〈标识符〉（〈下标〉）{，〈标识符〉（〈下标〉）}

DIM 语句动态地为数值或者字符串数组分配空间。字符串数组的元素可以是1到255个字节长度，这个长度随不同的值而改变。初始时，数值数组被设置为 0，字符串数 组设置为空。数组必须明确地定出它的维数。CBASIC数组是按行存放的。

下标是用来规定维数的大小的，数组维数的大小要先声明。下标若是不出现，则所有的下标意味着是0。

DIM 语句举例：

```
DIM  A(10)

DIM  ACCOUNT$(100),ADDRESS$(100),NAME$(100)

DIM  B%(2,5,10) ,SALES.PERSON%(STAFF.SIZE%)

DIM  X(A%(I%) , M%, N%)
```

相同的标识符在同一程序中可同时作为变量和下标变量。

3.5 表达式

函数，变量，常数，操作符的代数组合构成了表达式。它们的取值可以是整型、实型

或字符串型。函数在第八章讨论。操作符的优先级如下：

1) 嵌套在内的圆括号（ ）优先级最高。
2) ^ 乘幂运算
3) *，/
4) +，-，字符串联结（+），一目运算符+，-。
5) 关系运算符<，<=，>，>=，=，<>
   LT，LE，GT，GE，EQ，NE
6) NOT
7) AND
8) OR，XOR

算术和关系运算符可以完成整型或实型数的运算。如果一个算术和关系运算符联结的数值是整型和实型，整型数首先被转换成实型数。然后进行两个实型数的运算，结果也是实型数。当进行混合运算方式时，会涉及到这个问题。混合方式操作，会增加执行的时间并且编译程序会产生更多的中间代码。混合方式的表达式总是实型值。

如果使用实型值，乘幂运算操作将通过计算幂底数的对数来实现。因为负数的对数是无定义的，当一个数的操作结果为负时，将给出一个警告。结果由计算负数的绝对值来得到。指数可以是正值也可以是负值。如果两者或者是整型常数，或者是整型变量，则结果可以用连续乘法得到。这里允许负整数做整型乘方运算。在整型的情况下，如果指数是负的，结果为0。在一般情况下，0^0是1，0^X（当X不等于0时）是0。如果指数是整型而底是实型，先将整型转换为实型，然后计算结果。同样，如果指数是实型而底是整型，结果仍用实型数计算。

字符串变量仅能在关系运算符和联接运算符上操作。串和数值的混合操作是不允许的。关系运算符（LT，LE，等）可用相应的代数运算符（<，<=等等）代替。

表达式举例：

```
amount * tax

cost + overhead * percent

a * b/c(1.2+xyz)

last.name$+", "+first.name$

index%+1
```