BASIC

通用手册

傅教智 陈月 译编

黑龙江科学技术出版社

BASIC 通用手册

傅教智　陈月　译编

黑龙江科学技术出版社

1987年·哈尔滨

内 容 简 介

本手册共搜集了世界上200多种大中小及微型计算机的420多条BASIC 语句、命令、函数及操作符，并把它们作为单字和基本条目，从类型、功能用途、测试举例、使用技巧、各种写法及缩写形式、不同的用法、使用中的注意事项、参见条目等多方面进行了逐条逐项的介绍。这是一本内容全面，有助于读者改写不同机器的源程序，便于手头查阅的 BASIC 语言百科全书。

本手册适于程序员、学校教师、大中专及中学学生、一般工程技术人员、经济管理人员、各种计算机用户使用和参考。

责 任 编 辑：张丽生

封 面 设 计：张可欣

BASIC 通用手册

傅教智 陈 月 译编

黑龙江科学技术出版社出版

（哈尔滨市南岗区建设街 35 号）

北京财联印刷厂印刷

新华书店首都发行所发行

787 × 1092 毫米 16 开本 18.5 印张 500 千字

1987 年 12 月第一版 · 1987 年 12 月第一次印刷

印数：1 — 10,000 册

书号：15217 · 334 定价：4.90 元

前　言

本手册英文原名叫《The BASIC HANDBOOK — Encyclopedia of the BASIC Computer Language》，是美国八十年代出版的一本 BASIC 语言百科全书。作者 David A · Lien，曾做过教师，讲授电子学、数学和计算机科学。他也是一位计算机和航空航海技术专家，出版过一系列计算机方面的工具书。

本书共搜集了世界上200多种大中小及微型计算机的420多条 BASIC 语句、命令、函数、操作符，并把它们作为单字，从类型、功能用途、测试举例、使用技巧、各种写法及缩写形式、不同的用法、使用中的注意事项、参见条目等多方面做了逐条逐项的介绍，是一本便于查阅的工具书。

不少同志在应用计算机的过程中，往往感到在不同机器上编写出的 BASIC 程序缺乏通用性，移植起来比较困难。本书为解决这方面的问题专门设立了"如果计算机没有本单字怎么办"栏目，给用户提供了模拟完成一些单字功能的具体方法或子程序，以帮助读者解决程序间不能兼容的问题，扩充计算机的潜在功能。

根据近年来 BASIC 语言的发展和读者的需要，本书在后一部分对磁盘 BASIC 作了简要介绍，读者可在处理文件时参考。

我国近年来，有关 BASIC 语言方面的图书出版了不少，有些内容不错。但是归纳起来，这些图书一般只限于三个方面：一类是 BASIC 语言和程序设计方面的教材或读物，这类书由于介绍基本内容的需要，通常只涉及几十个 BASIC 单字；第二类是 BASIC 语言程序库，又叫程序汇编，主要介绍一些典型或专门性的程序；第三类是随一些具体型号机器出厂的 BASIC 用户手册（如《APPLE Ⅱ使用手册》，《IBM PC BASIC》），这类书虽然也称手册，但实际上是某一特定机器的技术说明书或技术资料。而能够对各种机器的每一 BASIC 单字作全面介绍，便于各个层次读者使用的工具书目前在我国还尚为少见。本书就是根据这种情况选题翻译的。

本书除了可供程序员使用外，也可供教师教学参考。对于正在学校学习的大中专甚至高中学生以及具有初步 BASIC 知识的一般读者，本手册也可供他们在深入学习全面掌握 BASIC 时参考。而对于一般工程技术人员、经济管理人员和计算机用户，本手册则可作为他们进一步掌握和正确使用 BASIC 的工具书。

本书在翻译过程中，曾得到王乃林等同志的大力支持，在此向他们表示感谢！由于我们水平较低，错误之处在所难免，望读者批评指正。

本书的编排原则

下述原则为确定本书的编排内容的主次详略提供了基本依据。一本书若不在范围上作出明确规定，就不能称其为是一本好书，读者也不会从中受益。下面简要叙述这些原则，读者可以从中领会本书的编排思路。

1. 本手册是一本 BASIC 语言百科全书，而不是一本字典或教材。本书的着眼点在于对每一 BASIC 单字作出准确的权威性的解释，以便读者在阅读 BASIC 教材时参考。但本书并不想代替教材。

2. 本手册的首要任务是帮助 BASIC 用户解决源程序在不同机器间移植时所遇到的无法兼容的问题。不同的机器，往往讲不同的 BASIC "方言"，相互之间缺乏通用性。本书想在"转译方言"方面给读者提供一些帮助。

3. 本书的"主线"是多数计算机所通用的 BASIC 中心字。用较多的篇幅介绍中心字比花费大量的笔墨介绍只用于特定目的的"生僻字"似乎更有必要。当然，在可能的情况下，本书也介绍一些不常用的单字。但是，BASIC 是一种不断发展不断丰富的语言，象探索无限的宇宙空间一样，我们只能不断搜集这些字，但永远也无法穷尽这些字。

4. 无论是袖珍计算机，还是大型主机，都可以使用 BASIC 语言。但是当今，微小型计算机的 BASIC 功能也很强，因此，机型的大小在本书中没有作过多考虑，首先考虑的是语言本身的汇集和整理。

5. 为了便于查阅，本书对每个 BASIC 单字尽可能地都采用统一的格式叙述，对于那些不常用的单字，或只限于在个别机器上使用的在程序移植时又很少遇到的单字，只作为基本单字的其他写法或缩写字来处理。

6. 标识出 BASIC 单字属于哪家机器哪种版本并不重要，重要的是全面搜集每一个 BASIC 中心字。指明哪些厂家的特定 BASIC 单字的功能只是作为第二位的因素来考虑的。

7. 大型计算机通常使用编译 BASIC，小型计算机通常使用解释 BASIC。但就单字本身来说，二者之间并没有原则区别。本书中介绍的解释 BASIC 单字的内容通常也适用于编译 BASIC。

8. 为了减少篇幅，本书对关键的中心字尽可能多作一些介绍，相同的内容一般不作重复，读者参见有关条目即可。

9. 不同 BASIC 在控制磁盘、磁带以及打印机等外部设备方面往往采取不同的方式，使用的控制字也有所不同。这些控制字一般距离 BASIC 的中心字较远，又不适于工具书的格式化叙述方式，因此，本书没有把它们包括在内。

10. 本书不适用于做某一具体型号机器的用户使用手册，只能作为它的补充和参考资料。

11. 某些 BASIC 厂家有时要对自己的 BASIC 版本进行改进，以消除其中的错误和其它各种问题，但本书并不打算反映厂家或软件公司对它们的 BASIC 所作的每一次修改。

12. 电子游戏机上使用的一些 BASIC 单字通常不在正规机器上使用，这些游戏机的生产厂家也不打算遵循 BASIC 语言标准设计自己的解释程序。但是，根据其使用的单字的重要性及今后在其他计算机上的潜在应用可能，本书对一部分"离谱的方言"有时也做些简单介绍。

使 用 说 明

BASIC 单字是构成 BASIC 程序的基本单位。本书的中心内容，就是从不同的角度对各个 BASIC 单字进行介绍。为了便于读者更好地理解和有效地使用本手册，我们下面结合 SEG $ (函数)一例来介绍本书的编排形式。

(1)**本条单字**：即本条目所要解释的 BASIC 字。它是构成 BASIC 程序或起控制作用的基本字。本手册不包括那些在系统监督程序中和在编辑程序语言及其他计算机语言中使用的单字。

(1A)**又作**：本条单字的缩写和其他拼写形式，一般放在{ }内。

(2)**ANSI 标准标记**："ANSI"为美国国家标准协会的缩写，如果所解释的条目带有这一标记，则表明该单字属于美国国家标准协会所规定的 BASIC 最低限度词汇中的字。本书将这一标记放在[ ]内。在这个例子中没有出现这一标记，说明该单字不在 ANSI 之列。

(3)**词类**：BASIC 字分为四类。

命令 令计算机进行某种操作的单字，例如R U N，L I S T 等。有些计算机允许将部分命令放在 BASIC 程序中使用，在这种情况下，该命令也同时为语句。

语句 只出现在程序行内用以构成 BASIC 具体指令序列的一类字，计算机就是根据这些序列进行各种判断并完成某些预定的任务的。如：

```
PRINT A, B, C
```

若出现在程序中就为一条语句。

函数 使用这类字实际上是令计算机向外调用预先编制好的机器级微程序，执行某一相对复杂的"函数"运算。通过使用函数，可以在一条语句中实现诸如对三角函数和平方根的计算等。例如语句：

```
LET X=TAN(Y)                    PRINT LOG(A)
```

都使用了函数。

操作符 是用于进行某种特定比较或起限制作用的字符，如逗号、冒号和等号等等。

在本手册中，命令、语句和函数均按字母顺序出现，而一部分无法按字母顺序排列的操作符，则被安排在后面。

(4)**介绍与注释**： 介绍本条 BASIC 单字的作用和功能，必要时还对某些型号计算机的主要或特殊用法作些具体说明。

(5)**测试程序**： 提供给读者的简短程序，以测试所使用的 BASIC 解释或编译程序能否接受和使用本单字。

(6)**运行举例**：即计算机在接受本单字的情况下运行测试程序后可能做出的反应。若计算机不同运行结果可能也会略有不同，但通常情况下不会与运行举例结果相差很远。

(7)**提示**：有时，这里给读者提供一些编写程序的技巧。这些技巧既可使写出的程序可靠，又可使编写工作简化。

(8)**其他写法**：不同的计算机对于同一 BASIC 字可能会有不同的拼写形式。本单字其他一些写法在此一并列出。

(9)**如果计算机没有本单字怎么办**：在可能的情况下，在此介绍一下通过采用其他 BASIC

字而编写的替代子程序，它可以实现与本单字基本相同的功能。当然，并非在任何情况下都能做到这一点。对于函数，子程序可以间接实现尚缺的内部函数的功能；对于语句，一段用其他单字和方法新写的程序段也可以实现与本单字相同或相近的功能。

(10)**不同用法**：即本单字在不同计算机上使用时在用法上的不同之处(注意：不是因为使用不同的单字带来的结果上的差异)。

(11)**参见**：本书不想用过多的篇幅去介绍重复性的内容。BASIC 字常常"成族"出现，每族都有个中心字，本书仅对其中的中心字作详细介绍，而对非中心字只介绍其特定的用途。因此，参见其他条目有助于读者更好地理解本条内容。

SEG $　{SEG}　[　　]　(函数)

(1)　　(1A)　　(2)　　(3)

(4) SEG $ 的功能是从某个串变量所代表的字符中截取某一段作为子字符串。该函数的自变量有三个：字符串变量，子串在字符串变量中的起始位置以及子串的长度。

例如若：A $ ="COMPUTER"，则执行 PRINT SEG $ =(A $, 4, 3) 后，计算机即会打印或显示出 PUT。

(5) 测试程序

```
10 REM * SEG$ TEST PROGRAM *
20 A$ ="CONTESTANT"
30 B$ =SEG$ (A$, 4, 4)
40 IF B$< >"TEST" THEN 70
50 PRINT "SEG$ PASSED THE "; B$
60 GOTO 99
70 PRINT "SEG$ FAILED THE TEST"
99 END
```

(6) 运行举例

```
SEG$ PASSED THE TEST
```

(7) SEG $ 函数可用来模拟 LEFT $ 和 RIGHT $ 函数，SEG $ (A $, 1, 4) 相当于 LEFT(A $, 4)，而 SEG $ (A $, LEN(A $) — 3, 3) 则相当于 RIGHT $ (A $, 3)

(8) 其他写法

个别计算机写做 SEG 。

(9) 如果计算机没有本单字怎么办?

如果读者的 BASIC 中既没有 SEG $ 也没有 SEG，可以在测试程序中用 MID $ 试一次。如果也不奏效，则可能是因为某些计算机规定要对整个字符串作 DIM 说明(如

Hewlett-Packard 机就是这样)，但这类机器在截取第4 到第7 个字符作为子串时一般不写做 SEG $ 或 SEG，而直接采用 A $ (4,7)的形式。

(10) 不同用法

不知。

(11) 参见

MID $，LEFT $，RIGHT $，DIM

ABS {A.} [ANSI] (函数)

ABS 出于绝对值 ABSolute value 一词，该函数用于确定一个数或数值型变量的绝对值。一个实数，在不计其正负号时的值，叫做这个数的绝对值。

例如执行 PRINT ABS(-10)，就会打印或显示出10。

无论多大多小的数，只要在计算机 BASIC 解释程序所能表示的范围之内，ABS 均可进行绝对值处理。

测试程序1

```
10 REM 'ABS' TEST PROGRAM
20 X = 35
30 PRINT "ABS PASSED THE TEST IF";
40 PRINT ABS(-435.28);
50 PRINT ABS(-.03245);
60 PRINT ABS(-X)
70 PRINT "ARE ALL PRINTED AS POSITIVE VALUES."
99 END
```

运行举例

```
ABS PASSED THE TEST IF 435.28 .03245 35
ARE ALL PRINTED AS POSITIVE VALUES.
```

大多数 BASIC 解释程序均允许对算数表达式使用 ABS 函数。当程序中的某个算数表达式值不想取负；而需要转成正值时，这个函数是很有用的。

作绝对值处理的算数表达式需要放在 ABS 中的括号内。

测试程序2

```
10 REM 'ABS' MATH OPERATION TEST PROGRAM
20 A = 18
30 B = 58
40 PRINT "THE ABSOLUTE VALUE OF "; (A-B) / 2; "IS"; ABS( (A-B) / 2)
99 END
```

运行举例

```
THE ABSOLUTE VALUE OF -20 IS 20
```

其他写法

某些计算机用 A.作为 ABS 的缩写。

如果计算机没有本单字怎么办?

如果机器本身不带 ABS 函数，可用下面的程序模拟。

测试程序3

```
10 REM 'ABS' SUBROUTINE TEST PROGRAM
20 PRINT "ENTER A NEGATIVE NUMBER";
```

```
30 INPUT X
40 GOSUB 30010
50 PRINT "THE ABSOLUTE VALUE OF";X;"IS";Y
60 GOTO 20
30010 REM * ABS(X) SUBROUTINE * INPUT X,OUTPUT Y
30012 Y=X
30014 IF X>=0 THEN 30018
30016 Y=-X
30018 RETURN
30999 END
```

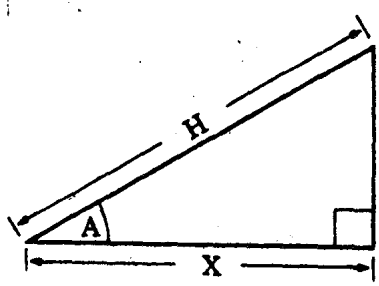
运行举例 (以 *-35.5* 为例)

```
ENTER A NEGATIVE NUMBER ? -35.5
THE ABSOLUTE VALUE OF -35.5 IS 35.5
ENTER A NEGATIVE NUMBER ?
```

ACS {AC.,ACSD,ACSG,ARCOS} (函数)

ACS(n)函数在某些 BASIC 中用于计算比率 n 的反余弦函数 arccos。其计算结果是以弧度制而不是度制形式表示的(一弧度大约等于57度)。arccos(ACS)被定义为直角三角形的一个角(A),这个角的大小由邻边长(X)和斜边长(H)所构成的比率来确定:

A=ACS(X/H)



ACS 的反函数是 COS(余弦),角(弧度)的余弦是指在直角三角形中某角的邻边长与斜边长之比:

COS(A)=X/H

测试程序

```
10 REM 'ACS' TEST PROGRAM
20 PRINT "ENTER A COSINE VALUE (-1 TO 1)";
30 INPUT C
40 W=ACS(C)
50 PRINT "THE ANGLE WITH THE X/H RATIO OF ";C;" IS ";W;
   "RADIANS"
```

```
30999 END
```

运行举例

```
ENTER A COSINE VALUE (-1 TO 1)?  0
THE ANGLE WITH THE X/H RATIO OF 0 IS 1.5708 RADIANS
```

若想将结果由弧度制换成度制，可用57.29578乘以该函数。本例中第40语句如换成度制则应改成：

```
40 W= ACS(C) * 57.29578
```

有些计算机在计算反余弦时以度制或梯度制(grad)表示计算结果(100梯度=90度)，这时该函数分别写做 ACSD 和 ACSG。若在这种机器上运行测试程序，第40语句就要改成 ACSD(度制)和 ACSG(梯度制)。这时在运行举例中仍输入0，则结果就将分别为90度和100梯度。

其他写法

Sinclair ZX 80 机将此函数写做 ARCOS，SHARP 1211 机(TRS-80 袖珍计算机)则写成 AC..。

如果计算机没有本单字怎么办?

如果计算机未能通过第40语句的测试，但可使用 ATN(arctg)和 SQR(平方根)两函数，则也可用下面的方法求反余弦：

```
40 W= 1.5708-2*ATN(C/(1+SQR(1-C*C)))
```

如果 ATN 和 SQR 也没有，那么还可使用如下子程序计算 arccos。但这时本书 ASN 中的子程序也要放在这里，同该子程序一起使用。

```
30000 GOTO 30999
30500 REM * ARCCOS SUBROUTINE * INPUT C, OUTPUT W
30502 REM ALSO USES VARIABLES S, X, Y AND Z INTERNALLY
30504 S= C
30506 GOSUB 30520
30508 W= 1.570796-W
30510 RETURN
```

然后，再将测试程序第40语句修改成：

```
40 GOSUB 30500
```

若以度制形式表示 arccos 的计算结果，可再增加一句：

```
30509 W= W*57.29578
```

参见

COS, ASN, ATN, SQR, SIN, TAN

AND {A.} (操作符，语句)

AND 为"与"的意思，相当于人们日常所说的"并且"。它作为"逻辑代数"操作符，用于

IF–THEN 语句中。例如，IF A= 8 AND B= 6 THEN 80 。这一语句的意思是：如果 A 的值等于8 并且(AND)B 的值等于6，那么 IF–THEN 条件满足，计算机转到第8 0 语句继续执行。

测试程序1

```
10 REM LOGICAL 'AND' TEST PROGRAM
20 A= 8
30 B= 6
40 IF A= 8 AND B= 6 THEN 70
50 PRINT "AND FAILED THE TEST AS LOGICAL OPERATOR"
60 GOTO 99
70 PRINT "AND PASSED THE LOGICAL OPERATOR TEST"
99 END
```

运行举例

```
AND PASSED THE LOGICAL OPERATOR TEST
```

AND 操作符在个别计算机上可以用来对字符串的比较进行"逻辑与"操作。如语句 IF A$ = "A" AND B$ = "B" THEN 80 的意思就是：若字符串变量 A $ 等于字母 A 并且(AND)字符串变量 B $ 等于字母 B，则 IF–THEN 条件成立，程序转到第8 0 语句去继续执行。详细介绍请见本书的+ 和 * 操作符。

测试程序2

```
10 REM STRING LOGICAL 'AND' TEST PROGRAM
20 A$ = "A"
30 B$ = "F"
40 IF A$ = "A" AND B$>"B" THEN 70
50 PRINT " 'AND' FAILED THE TEST AS A LOGICAL OPERATOR"
60 GOTO 99
70 PRINT " 'AND' PASSED THE STRING LOGICAL OPERATOR TEST"
99 END
```

运行举例

```
'AND' PASSED THE STRING LOGICAL OPERATOR TEST
```

某些计算机用逻辑操作符 AND 来判别两个关系操作条件是否均成立。如果两个条件同时成立，则 AND 结果回送 －1；反之，若其中某一条件不成立或两个条件都不成立，则 AND 回送0。

比如语句 PRINT A= 4 AND B= 8 的意思就是：如果 A 等于4 并且(AND)B 等于8，则计算机打印或显示出 －1；如果其中有一个条件不成立或两个条件均不成立，则计算机打印或显示出0。

测试程序3

```
10 REM 'AND' LOGICAL TEST PROGRAM
20 PRINT "ENTER A NUMBER FROM 1 TO 10";
30 INPUT A
40 B= A >4 AND A <11
```

```
50 IF B= -1 THEN 80
60 PRINT A;"IS NOT GREATER THAN 4 AND LESS THAN 11"
70 GOTO 20
80 PRINT A;"IS GREATER THAN 4 AND LESS THAN 11"
99 END
```

运行举例 (特例)

```
ENTER A NUMBER FROM 1 TO 10? 2
 2 IS NOT GREATER THAN 4 AND LESS THAN 11
ENTER A NUMBER FROM 1 TO 10? 8
 8 IS GREATER THAN 4 AND LESS THAN 11
```

个别计算机还可以根据布尔代数,使用 AND 操作符对两个数进行辑逻与(AND)操作。对布尔代数我们在此不作过多介绍。在这种代数中,AND 通常用于比较两个二进制位,看两者是否都是二进制的"一"。当被"与"(AND)的两个二进制位均为一时,与的结果才为一。即:

```
1  AND  0  =  0
0  AND  1  =  0
1  AND  1  =  1
```

因此,一个数与另一个数相与(AND),是用其二进制值的各位与另一数的二进制值相应各位进行逻辑与,并相应产生第三个数。例如:

十进制		二进制
3		0 0 1 1
	(逻辑与)AND	
5		0 1 0 1
= 1		0 0 0 1

本例中,两个数只各自的末位才均为二进制的一,故与的最后结果为十进制的1(二进制的0 0 0 1)。

测试程序4

```
10 REM 'AND'BINARY LOGIC TEST PROGRAM
20 PRINT "ENTER A VALUE FOR X";
30 INPUT X
40 PRINT "ENTER A VALUE FOR Y";
50 INPUT Y
60 A= X AND Y
70 PRINT "THE LOGICAL 'AND' VALUE OF";X;"AND";Y;"IS";A
80 GOTO 20
99 END
```

运行举例 (*以6和10为例*)

```
ENTER A VALUE FOR X? 6
ENTER A VALUE FOR Y? 10
THE LOGICAL 'AND' VALUE OF 6 AND 10 IS 2
ENTER A VALUE FOR X?
```

其他写法

个别计算机(如 Britain Acorn 机)可以用 A.代替 AND。

不同用法

在 WANG 2200 B 型计算机上,AND 还可以作为语句使用。其功能是对两个字符串变量或十六进制数值进行二进制逻辑与,其格式为 AND(p,q),如果两个数值一个是串变量,另一个是十六进制常数,它们也可以相与,但第一个数p 必须为串变量,第二个数q 只能是一个两位的十六进制常数。与的结果将取代第一个变量。

例:P$ =十六进制 5A,Q$ =十六进制的3C,AND(P$,Q$)为 P$ =十六进制1 8:

	十六进制		二进制
	P$ =5A		01011010
		AND	
	Q$ =3C		00111100
则	P$ =18		00011000

如果 P$ 为字符串而 Q 为十六进制常数,那么字符串中的每个字符需先转换成 ASCII 值,之后分别与十六进制常数进行逻辑与,与的结果值再转回字符形式并存到 P$ 中。例如字符串"EFG"与十六进制4 3 相与便得到字符串"ABC"。

	E	F	G
ASCII 码	69	70	71
十六进制	45	46	47
二进制	0100 0101	0100 0110	0100 0111
十六进制	AND	AND	AND
十六进制 43	0100 0011	0100 0011	0100 0011
=	0100 0001	0100 0010	0100 0011
ASCII 码	65	66	67
P$ =	A	B	C

测试程序5

```
10 REM 'AND' TEST PROGRAM
20 DIM A$3 (注:该语句的作用是定义 A$ 为三个字节长.)
30 A$ = "CCC"
40 AND(A$,F1)
```

```
50 PRINT "AND PASSED THE TEST IF  ";A$;"= AAA"
99 END
```

运行举例

```
AND PASSED THE TEST IF AAA= AAA
```

如果计算机没有本单字怎么办?

逻辑操作符 AND 的作用可通过使用 IF–THEN 语句间接实现。请将检验程序1 的第4 0、第5 0 语句改成:

```
40 IF A < >8 THEN 50
45 IF B = 6 THEN 70
```

WANG 2200 B型计算机的 AND 语句也可以在其他计算机上间接实现,但需要将字符串处理函数与逻辑操作符的 AND 连在一起用。如对上面测试程序作如下修改就可以实现 AND 语句的功能:

```
40 REM 'AND' REPLACEMENT
41 N= LEN(A$)
42 FOR I= 1 TO N
43 C$ = C$+ CHR$ (ASC(MID$ (A$,I,1)) AND 241)
44 NEXT I
45 A$ = C$
```

参见

OR, XOR, NOT, *, +, =, <, >, < >, < =, > =

APPEND (命令)

APPEND 命令的功能是,从外存储器(如磁盘、磁带)上调入某一程序,使之同内存储器中的现有程序合并到一起。例如打入命令:APPEND PROG 2,就会将外存中名为 PROG 2 的程序调到内存,并使其连接在内存中常驻程序之后。"外面"调入的程序行号必须大于内存中现有程序的最大行号。

测试程序

(本例假定外存储器为磁带)。为了测试计算机是否可以接受 APPEND 命令,请先将下面名为 PROG 2 的短程序保存在磁带上(参见 CSAVE)。

```
1000 PRINT "THESE LINES ARE"
1010 PRINT "FROM PROG2"
1020 END
```

存好后打入 NEW 或 SCRATCH 命令将其从内存中清除,接着输入程序 PROG 1:

```
10 REM 'APPEND' TEST PROGRAM PROG1
20 PRINT "THESE LINES ARE"
30 PRINT "FROM PROG1"
40 PRINT "        BUT..."
```

这时再打入 APPEND PROG2 。

当 PROG 2 调入内存并与 PROG 1 连接后，使用 RUN 命令运行这个连接程序。(在使用 SAVE PROG 2 和 APPEND PROG 2 时请注意，不同计算机在对程序文件命名时，对引号的使用，是否可用单个字母命名等等，都有着不同的规定。)

运行举例

```
THESE LINES ARE
FROM PROG1
        BUT...
THESE LINES ARE
FROM PROG2
```

APPEND 通常用来向内存装入较大的数据文件，或给内存中的常驻程序"追加"常用子程序(这也是本手册中主要子程序之所以都使用30000以上的行号且各子程序间行号并不重叠的原因)。

如果计算机没有本单字怎么办?

若某些计算机对 APPEND 不能作出有效响应，可用 TAPPEND、MERGE 或 WEAVE 等命令另试一次。有时这些命令也不奏效，但如果位于当前程序上的"指针"能进行进行定位的话，程序连接也不是不可能的。不过这样做多少要费点事，读者在连接前可查阅一下自己的计算机手册。下面是使用 TRS-80 机进行连接的步骤:

定位指针于内存中当前程序的末端地址。在 TRS-80 I 型计算机中，该地址存储在16333和16334中，可用 PEEK 命令找到存储在这里的数值。

从第一个数(存在16333单元)中减去2，如结果为负，则加上256。并且从第二个数减去1。

在上面任意一种情况下，均用 POKE 将两个结果存到16548和16549地址中。

现在，可以用 CLOAD 从磁带上装入指定的连接程序了。之后再用 POKE 恢复16548-16549地址中的内容，即将233和66分别存到16548和16549地址单元中。

这样，第二个程序就连接到第一个程序上了。

参见

TAPPEND, PEEK, POKE, CSAVE, CLOAD, DATA

ASC {ASCII} (函数)

ASC 函数的功能是将某一字符或字符串变量转换成相应的 ASCII 十进制数码。例如执行 PRINT ASC("A")，机器就会打印或显示出字母 A 的 ASCII 代码65；而 PRINT ASC(A$) 则会打印或显示出串变量 A$ 所代表的字符串的第一个字符的 ASCII 码值。

测试程序1

```
10 REM 'ASC(CHARACTER)' TEST PROGRAM
20 PRINT "THE ASCII CODE FOR LETTER A IS";
```

```
30 PRINT ASC("A")
40 IF ASC("A") = 65 THEN 70
50 PRINT "ASC FAILED THE TEST"
60 GOTO 99
70 PRINT "ASC PASSED THE TEST"
99 END
```

运行举例

```
THE ASCII CODE FOR LETTER A IS 65
ASC PASSED THE TEST
```

下面是使用串变量测试 ASC 函数的程序。

测试程序2

```
10 REM 'ASC(STRING VARIABLE)' TEST PROGRAM
20 PRINT "TYPE ANY LETTER, NUMBER, OR CHARACTER";
30 INPUT A$
40 PRINT "THE ASCII CODE FOR  ";A$;"  IS";ASC(A$)
99 END
```

运行举例 (*以 H 为例*)

```
TYPE ANY LETTER, NUMBER, OR CHARACTER?  H
THE ASCII CODE FOR H IS 72
```

某些计算机的 ASC 函数可以接受由多个字符组成的字符串，但仅该串中的头一个字符才能转成 ASCII 代码。若在这种计算机上测试 ASC 函数最多可接受多少个字符，可在执行测试程序2 时依次输入一个长字符串直至机器发出出错信息为止。

其他写法

某些计算机(如 DEC-10)直接将该单字写成 ASCII。

不同用法

有些 BASIC 解释程序(如 MAX BASIC)使用 ASC(A$,X)的形式打印 A$ 串中自左边开始往后第一个出现的 X 的 ASCII 代码。

参见

CHR$, CODE, 附录中的 ASCII 码。

ASN {ASNG, ASND, ARCSIN} (函数)

ASN(n)为 TEKTRONIX 4050 系列计算机 BASIC 所使用的一个函数，其功能是计算比率 n 的反正弦值 arcsin(注意：计算结果为弧度而不是度，一弧度大约等于57度)。arcsin 被定义为直角三角形的一个角(A)，这个角的大小由对边长(Y)与斜边长(H)之间的比率确定：