陈立潮　陈桂芳　编

BASIC语言程序调试技术

北京理工大学出版社

BASIC语言程序调试技术

陈立潮　陈桂芳　编

北京理工大学出版社

内 容 提 要

本书是一本关于微型计算机BASIC语言程序调试方面的必备工具书。书中结合大量的实例，对规范化编程、错误信息分析和理解以及程序调试的基本方法与技巧进行了详细的论述。并给出了一个实用的调试软件。各章间既相互联系又自成体系。采用由浅入深、循序渐进的方式编写，以便于自学。

本书可作为程序员的一本有益的参考书，对于用BASIC语言编程的用户和读者是必不可少的工具书，另外也可作为大专院校有关专业师生学习程序语言方面的参考书。

BASIC语言程序调试技术

陈立潮　陈桂芳　编

北京理工大学出版社出版

新华书店北京发行所发行　各地新华书店经售

北京密云华都印刷厂印装

787×1092毫米　32开本　9.25印张　194千字

1988年12月第一版　1988年12月第一次印刷

ISBN 7-81013-083-9/TP·9

印数：1—7500册　　定价：1.85元

前 言

本书的目的是帮助读者在微型计算机上调试BASIC语言程序。

使用过BASIC语言编程的读者，也许都有过在自认为应该正常运行的程序中花费大量的时间和精力来寻找令人困惑的错误的经历。本书不是专门讲授BASIC语言的语法和结构化编程技术。这里假定读者已具备了这两方面的知识。

本书与已出版的BASIC语言和结构化编程技术的教科书并不矛盾，只是作为使用BASIC语言编程的一个补充。因此，其重点仅在于调试技术的讨论。

该书第一部分介绍了BASIC语言调试理论，它由三章组成。其中，第一章是规范化编程技术，作为预防和排除错误的基础。第二章讨论了对BASIC语言的错误信息的理解，这里参考了IBM PC BASIC用户手册。第三章给出了大量的调试策略和技巧以及用于跟踪、查找错误的各种方法。

以上三章采用了由浅入深、循序渐进的方法编写，以便于读者自学。各章之间既相互联系又自成体系，以便于读者直接从自己相关联的问题入手来学习某种特定的方法。然而，若系统、完整地学习这三章，便可获得更加完备的知识。有经验的读者将会看到这三章的内在联系，并从头阅读来解决自己的问题。

本书的第二部分针对以上的策略与方法介绍了一个实用的BASIC语言程序调试软件——DEBUG M.A.S.T.E.R.实用程序。该实用程序主要集中于两个方面的调试讨论。其一讨论了一个单步的实用程序（single-stepping utilities），它允许读者逐行运行程序并可跟踪变量的值；其二讨论了一个相互对照的实用程序，它可按行号列出程序中的变量、保留字和控制分支。

附录A提供了DEBUG M.A.S.T.E.R.实用程序的源程序，以便于读者参考这些实用程序，更好地调试自己编写的程序。所有这些实用程序均用BASIC语言编写。

附录B给出了准备DEBUG M.A.S.T.E.R. Uticities工作盘的操作说明。

附录C提供了一些错误信息参考，以帮助读者方便地找到出现错误的根源。

我们希望本书提出的实用程序对不同程度的程序员，在开发程序时都具有一定的帮助。初级程序员也许会感到其帮助更大。在这些实用程序中，单步程序是最重要的一个调试工具，通过观察逐行执行的不同代码结构，对BASIC语言的语法将会得到进一步的掌握。对于熟练的程序员，也可作为一本很好的参考资料，这些实用程序仍然具有一定的价值。

以上所介绍的调试方法和实用程序，可适用于IBM PC与其兼容的同类型微型机，如长城0520、SUPER、UTEK等。其中的一些调试方法与技巧也适用于对其它高级语言程序调试作参考。

对于使用BASIC语言进行程序设计的读者，本书将是一本较实用的调试工具书，其中涉及的一些问题是借助BASIC

语言参考手册无法解决的。为了方便读者，我们准备了一个实用的调试工作软盘，借助该软盘可用上述介绍的方法直接对你的程序进行调试。有兴趣的读者可与太原机械学院联系。

本书参考了Jerome R.Corsi and William F.Hills编写的《Debugging Techniques for IBM PC - BASIC》和Rorbert C. Bruce编的《SOFTWARED-EBUGGING FOR MICROCOMPUTER》。由太原机械学院陈立潮和海军电子工程学院陈桂芳共同编写而成，其中还包括了笔者多年来教学中的一些体会和经验。全书由海军电子工程学院计算机教研室主任孙玉歧副教授进行了校阅。由于我们水平有限，难免存在不少错误，敬请读者不吝指教。

目 录

第一部分
BASIC语言程序调试理论

第一章　规范化编程技术

1.1 引　　言

BASIC 语言是微型计算机上最常用的程序设计语言，也是各类院校普遍开设的计算机语言课程之一。随着计算机语言的发展和计算机应用领域的扩大，BASIC语言的语句、函数等越来越丰富，语言本身的功能也越来越强。从而，使BASIC语言的版本逐步升级。由于功能的扩充，使得这种语言能够解决各个领域的实际问题。为了使在实际工作中编写出的程序出错机会较少，以及出现了错误也能很快地找到错误的性质和根源，我们在本章提出了一些程序书写方面的技巧。这些技巧的提出与以往的调试技术的出发点有所不同，在此我们不想被动地纠正运行程序时给出的一个个错误，而是积极主动地在编写程序的过程中避免出现错误，即使出现了错误，也能容易地进行修改，这对缩短程序开发周期是很有用的。

在编写BASIC语言程序时，尽管标准的书写程序代码也未必不会在编程中发生错误，但研制开发一套合理的程序代码书写技巧，可以更可靠地避免一些不该发生的错误。对各类水平的编程人员而言，粗心大意的书写程序将会增加出错机会，并为寻找这些错误带来困难。简单地说，规范化的书写程序代码可使程序便于调试，一旦发现了错误，也能容

易地找到错误所在的位置。

进一步讲，所谓规范化编程也有多种解释，这要依程序员的目标而定。若程序员的目标是写出尽可能有效的代码，那么，程序中除了基本的语句之外，其余的（如注释、说明等）都不需要。若程序员的目的是希望设计出的程序代码能经济地利用存贮器，以提高程序的执行速度，则对“规范化”又有不同的理解。所以，对每一个目标而言，其代码书写的标准也不尽相同。

本章的目的在于研究一种便于调试的程序代码书写技术。对于已能写出十分有效的程序代码的读者，也可略去这部分内容。不过，这里提出的所有方法都是一些最基本的方法，即希望能为程序员提供一种规范化的程序代码书写技巧，以增加程序代码按预期设计而工作的可能性。一旦程序代码能按预期的设计运行时，其它的目标（内存的经济利用和/或最小执行时间）便可通过对已写出的程序代码进行修改来达到。

不管程序员的目标如何，都要努力写出规范化的、易于阅读的程序代码，并与目标的要求相一致。

于是，对于书写BASIC语言程序而言，我们在这里提出的方法是着重于完成调试的两个目标：减少错误的产生和快速地消除错误。减少了编程中出现的错误个数，也就减少了调试的影响范围。同样重要的是，以调试为目的所写出的程序代码，可使我们相当容易地寻找和快速修改这些错误。

当你第一次阅读其他人写出的规范化程序时，会很容易地赞赏程序员使程序代码易于理解而所做的努力。随后再看自己没按任何规则写出的程序代码，就会感到十分别扭，并

且要理解自己做了些什么，以及为什么要这样做，将会花费很长的时间。

本章所提出的一些方法和规则，仅是一个有指导意义的参考。我们在本书中所列出的程序清单，包括附录A中给出的调试程序源代码清单，都是按本章所介绍的书写程序的建议给出的。然而，当阅读这些源程序时，就会发现不同的作者都有自己的独特的风格。

对于完成同一个任务而言，两个程序员所写出的程序也不会完全一样。在这里，我们的观点是要求所有的程序员写出规范化的程序，其规范化的重要标准之一是程序的可读性。

作为开始，先考虑一段较简单的程序：

```
10  A = 5
20  B = 10
30  print a * b/10 + 3 * A
```

这是一个可完全被计算机接受的程序，在IBM PC微机上运行该程序后会打印出“20”作为表达式“a * b/10 + 3 * A”的值。然而，这一程序不能算是规范化程序。为了规范化编程，下面分几节来讨论建立构造程序代码的几个核心原则。

值得注意的是，任何版本的BASIC语言，都有自己内部的一些规定。如果不按这些规定去做，系统就会提示出错误信息。关于IBM PC BASIC语言的这些规定，请参考“IBM PC BASIC语言手册”。我们在这里指出的是在这些约定之下，提出几点规范化编程的原则。

1.2 程序代码书写技术

1.2.1 用大写字母书写BASIC语句

IBM PC支持用大写或小写字母，或者两者一起来书写BASIC程序。因此，表达式“PRINT”、“Print”“print”，甚至于“prINT”、“PriNT”以及“PRiNT”，对计算机而言都是等价的。BASIC解释程序可忽略大写或小写字母间的不同。不管原始程序是怎样写出来的，经BASIC解释程序所列出的程序清单都自动地把保留字大写化。但用文本编辑程序列出的程序清单保持原始程序的面貌。

用文本编辑程序编辑的程序和没有经过BASIC编译程序而编译的用户程序，如果在解释程序下运行时，解释程序也会自动地把BASIC的保留字和变量名大写化。

学过英语的读者都知道怎样区别大写和小写。根据英语的习惯，单词“PRiNT”是错的。或者作为一个特例，但要给出说明，只有在这种说明意义之下才是正确的。尽管BASIC程序员可使用英语单词，但编写程序可以不遵循英语的习惯。然而，BASIC程序是程序员对用BASIC语言编写的程序意图的唯一记录。因此，习惯于英语的程序员对程序按英语方式理解与计算机按程序语言对程序理解之间的谐同是十分重要的。

我们建议：对BASIC的所有命令、语句、函数和变量名，都使用大写字母。这样，“PRINT”可理解为BASIC

的保留字。这与程序员希望在程序的某处表示字符串的英语单词“print”就区分开了。

这样，上节中的示范程序改进后如下：

```
10  A = 5
20  B = 10
30  PRINT A * B/10 + 3 * A
```

然而，为使该程序更加规范化，还需做进一步的改进。

1.2.2 数学表达式中尽量使用括号

尽管代数运算规则都有其明确的意义，但表达式“A * B/10 + 3 * A”咋一看似乎有些含糊。若表达式的意义是计算（A *（B/10））+（3 * A），则当变量A = 5且变量B = 10时，这个表达式的值为20。另一方面，若表达式是计算((A * B) /（10 + 3）) * A，当A = 5且B = 10时，表达式的值是19.23077。

在计算代数表达式时，BASIC语言根据代数运算的优先级规则来确定执行运算的先后次序。根据这些规则，代数运算是按照下面的降次优先排列次序来完成的：首先是乘方，然后取负，再乘除、取商整除、模运算，最后是加减运算。

依这种排列顺序，在上例表达式中，乘除法优先于加减法运算。就该表达式而言，BASIC解释程序能够明确其意义，给定A = 5和B = 10后，计算的结果为20。然而，就是这样简单的表达式也可完全把读者搞糊涂。对于更复杂的表达式，如果再书写不清楚，即便是精通代数运算规则的读者，若要明确它们的意义必须要做一番仔细的捉摸。

为使程序中表达式的执行次序更加明显且易于阅读，我

们建议，在表达式计划执行的次序上尽量使用括号来组合运算。

提出这个规则的另一个原因是，使用括号可使程序员象计算机执行那样，构造一个算术表达式。为了说明这个问题，考虑下面给出的因运算次序不同而得出不同结果的例子：

```
10  A = 1.09E + 28
20  B = 0.000050000023#
30  PRINT (A * B) /2.777E + 15
RUN
1.962551E + 08
```

而

```
10  A = 1.09E + 28
20  B = 0.000050000023#
30  PRINT A * (B/2.777E + 15)
RUN
1.96255E + 08
```

可见，最后所得结果的精确度依赖于乘、除法的运算次序。因此，在优先级相同的情况下，若要求先做除法后做乘法以及先做减法后做加法运算，或反之，这时括号不仅是希望的，而且也是必需的。这样可避免出现象上例中的情况。

在数学表达式中所用的括号明确地描述了所希望的运算次序。现把1.2.1节中的例子加进括号后重写如下：

```
10  A = 5
20  B = 10
30  PRINT ((A * B) /10) + (3 * A)
```

1.3 易读的程序代码规范

1.3.1 选用简短且有说明性的变量名

BASIC语言在程序员选择变量名时提供了相当宽广的范围。IBM PC BASIC允许变量名的长度可扩展到40个字符。字母表中的所有字母、数字以及小数点都可作为变量名中的字符。但变量名的第一字符必须是字母，不能是数字或小数点。保留字（用作BASIC的语句、命令或函数的词）不能用作变量名，但可作为变量名的一部分嵌套在名字中。如：PRINT本身不能用作变量名，但PRINTOUT就可以了。不允许用FN作为变量名的起始符，因为这两个字母的这种组合被用来调用用户自己定义的函数。BASIC语言所使用的关键字如下表所示：

ABSI	AND	ASC	ATN	AUTO
BEEP	BLOAD	BSAVE	CALL	CDBL
CHAIN	CHR$	CINT	CIRCLE	CLEAR
CLOSE	CLS	COLOR	COM	CVD
CVI	CVS	DATA	DATES	DEF
DEFDBL	DEFINT	DEFSNG	DEFSTR	DEFETE
DIM	DRAW	EDIT	ELSE	END
EOF	EQV	ERASE	COMMON	CONT
COS	CSNG	CSRLIN	FILES	FIX
FN++++	FOR	FRE	GET	GOSUB
GOTO	HEXS	IF	IMP	INKEY$