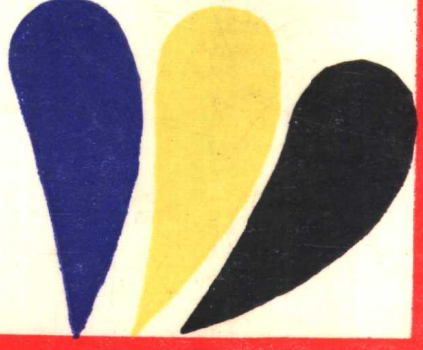
计算机知识普及系列丛书

QuickBASIC4.5

备查手册

（合订本）

秦笃烈
陈捍东 编写



学苑出版社

计算机知识普及系列丛书

QuickBASIC 4.50 版备查手册

陈捍东等　编写

秦笃烈　刘学宗　审校

学苑出版社

1993.

(京)新登字 151 号

内 容 提 要

本书共由三册组成.第一册为 Quick BASIC 4.50 版高级编程指南,第二册为 QuickBASIC 程序开发环境,第三册为 QuickBASIC 4.50 程序设计精要,三册相互独立又彼此参照,便于读者学习和使用 QuickBASIC.是 BASIC 程序员案头必备参考资料。

欲购本书的用户,请直接与北京 8721 信箱联系,电话 2562329,邮码 100080。

计算机知识普及系列丛书

Quick BASIC 4.50 版备查手册

编　　写:陈捍东等
审　　校:张嘉刚　刘学宗
责任编辑:魏国宏
出版发行:学苑出版社　　邮政编码:100032
社　　址:北京市西城区成方街 33 号
印　　刷:东升印刷厂
开　　本:787×1092　1/16
印　　张:42.5　　字　数:969 千字
印　　数:1—3000 册
版　　次:1993 年 12 月北京第 1 版第 1 次印
ISBN7-5077-0821-7/TP·19
本册定价:33.00 元

学苑版图书印、装错误可随时退换

QuickBASIC 4.50 版

高级编程指南

陈捍东　编写

秦笃烈　审校

目 录

附 录

第一章　共同的起点

由于本书读者的计算机经历有很大差异，所以作者作了些假定以达到对这些读者都有益的共同的起点。本书假定你一直在使用较早版本的Quick BASIC，最好是Quick BASIC4.0版本，并因此具有关于它的相当的知识。使用Turbo Basic，MBASIC，ZBASIC，True BASIC，或这种永远流行的BASIC语言的其它一些版本的工作经验也将提供给你必要的背景知识。

如果你没有这种经历，那么作者建议你阅读并使用一本入门的Quick BASIC4.5的书，如，Bob Albrecht、Wenden Wiegand和Dean Brown的《QuickBASIC Made Easy》(Osborne/McGraw-Hill，1989)。或者你可选一本中间的QuickBASIC4.0的书，如Don Inman和Bob Albrecht的《Using QuickBASIC》(Osborne/McGraw-Hill，1988)。

1.1　讲一点Quick BASIC

任何作者把一个讨论划分为初、中、高三级的企图都是以读者的解释为转移的。所以，本章为有经验的高级程序员编写。

QuickBASIC4.5版是Microsoft介绍的最新QuickBASIC版本。它比解释BASIC运行得快得多并且仍保持了该语言的交互性。它集中了解释BASIC的交互作用的长处和编译器的速度及结构化模块化语言的功能。

除作了若干改动以使它更易于初级程序员使用外，QuickBASIC4.5的功能与Quick BASIC4.0的相似。如，现在建立过程提供了一个更简单更易于理解的带有树型结构菜单的SETUP程序。用这个程序，你可以在开始实际装配 QuickBASIC4.5之前作配置选择。这些选择都存在QB•INI文件中，以便SETUP再次运行时能记住并显示所有的配置选择。

QuickBASIC4.5版提供下面两个级别的菜单：

• 容易菜单

这些菜单提供最简单的用户接口。提供高级的或过多的功能的项目都从全菜单中删除了，可能使初学者混淆的一些不必要的项目也删除了，而且简化了一些对话箱。

• 全菜单

这些菜单提供所有高级程序员用到的功能特性。

SETUP程序允许下面两种级别的装配：

• 级别1　　本级别与容易菜单一起用。使用SETUP程序的缺省设置。你可以在任何愿意的时候从options菜单改变这些选择。

• 级别2　　本级别提供高级编程需要的所有功能特性。

由于本书是为高级程序员写的，所以容易菜单和级别1 SETUP除了在有必要加以区别的地方外都将被忽略。

QuickBASIC4.50版包含下面的功能特性：

• 与BASICA和GW-BASIC的兼容性。

• 一个内部的，全功能编辑器、编译器和调试器。

• 下拉菜单，对话箱，和联机帮助。

• 用键盘或鼠标器从菜单中选择命令。
• 在内存编译以节省时间。
• 程序行输入时检查语法。
• 用字母数字标号，结构化的逻辑语句，子程序，和多行函数支持结构化编程。
• 支持程序模块库。
• 支持．EXE文件的创建。
• 支持图形，BLOAD，BSAVE，声音，音乐，和事件捕捉

正如你所见，QuickBASIC使用了与MicrosofBASIC其它版本的特征的同样有力而友好且易于使用的语言环境。

QuickBASIC有许多高级功能特性，但它仍与IBM的高级 BASIC (BASICA) 和运行于许多IBM PC兼容机上的Microsoft的GW-BASIC兼容。QuickBASIC是给了解这些 解释版本的 Microsoft BASIC 的用户的。当然，如果你用过较早版本的QuickBASIC，则这一最新版本你将很快熟悉。

QuickBASIC4.5新增加的功能使用户从GW-BASIC和其它语言到QuickBASIC的过渡更容易。现在，用户可以从最初级别顺利地进入QuickBASIC的更高级的功能特性。当用户发展为高级用户后。可能就想进一步使用MicrosoftBASIC编译器的强有力的功能。

1.2 BASIC保留字的使用

现在讨论哪些BASIC语句和函数要用而哪些不用。你们中的许多人可能 用BASICA 或GW-BASIC编过程序。由于Microsoft的Quick Basic4.5与BASICA和GW-BASIC兼容，所以许多以ASCII码形式存放的BASICA或GW-BASIC程序 都可用QuickBASIC 装入，编译并运行。然而，某些BASICA和GW-BASIC语句和函数不能用，某些需要修改，而某些是从Microsoft BASIC较早版本中遗留下来的。下面一节讨论这样的语句和函数。

1.2.1 不用的保留字

下面清单中列出的BASICA和GW-BASIC语句和函数不能用于QuickBASIC4.5 程 序中，因为它们在源文件上执行编辑操作，妨碍程序执行，引用卡型盒式磁带机设备，或重复QuickBASIC环境提供的支持。

AUTO	DELETE	LLISL	MOTOR	SAVE
CONT	EDIT	LOAD	NEW	USR
DEF USR	LIST	MERGE	RENUM	

上述清单中的语句或函数在本书中都不用。

1.2.2 可修改的保留字

含有下面语句的BASICA和GW-BASIC程序在用QuickBASIC4.5编译和运行以前需要修改。QuickBASIC4.5怎样使用这些语句的详细描述见QuickBASIC软件包的《 programming in BASIC 》

BLOAD	CHAIN	DIM	RESUME
BSAVE	COMMON	DRAW	RUN
CALL	DEF	PLAY	

本书认为你知道必要的修改。本书讨论并使用上述清单中的语句，正如它们在（programming in BASIC 》使用手册中描述的那样。

附录A给出了QuickBASIC4.5保留字的完整的清单。除了这里列出的语句外，本书将可能使用那个清单中的任一保留字。

1.3 其它考虑

用户应知道怎样使用所有菜单及其相关对话箱。QuickBASIC允许用鼠标器和键盘选择菜单和对话箱。“Learning to Use QuickBASIC”解释了用键盘和鼠标器怎样在菜单和对话箱之间来回移动。因此本书认为用户知道怎样用适于你的系统的方法。本书不给出怎样选择菜单项或对话箱回答的详细情况。

用户也应具有一些使用QuickBASIC编辑器和调试工具的知识和经验，熟悉QkickBASIC4.5版的大部分QuickBASIC关键字。本书在引入新的QuickBASIC4.5关键字时讨论它们。

1.4 QuickBASIC回顾

为了理解QuickBBSIC术语，需要有一个清楚的称为模块的单元的概念。一个模块是一个含有QuickBASIC语句的单独文件。一个模块可以是一个独立的程序，或者它可以是一个含有一个或多个能由其它模块调用的SUB或FUNCTION过程的文件。含有多于一个模块的程序叫做多模块程序。当有一个多模块程序时，每个模块存在一个单独的文件中。

每个QuickBASIC程序都有一个主模块。在一个多模块程序中，主模块含有该程序运行时第一条执行的语句。一个模块可以是许多不同程序的一部分。但一个模块只能成为一个程序的主模块（该程序标有此模块的名字）。在一个单模块程序中，该程序包括任何SUB和FUNCTION过程，是主模块。

过程 SUB…ENDSUB 和FUNCTION…END FUNCTION对QuickBASIC程序特别重要。这些过程与子程序的DEF FN语句相似，但它们比旧的结构有明显优势。用这些过程编程的三个主要好处是：

- 它允许把程序分为分离的逻辑单元。每个单元都比不含这种单元的完整程序更易测试和改正。
- 一旦过程调试好了，它们就可作为建筑块用于同一或不同程序中。
- 总的来说，它们更可靠，因为它们只有一个入口，还因为它们内部说明的变量都由系统设置为这些过程的局部变量。

1.4.1 用户定义的函数

用户可以用DEF FN语句命名并定义自己的函数，象GW-BASIC和BASICA中那样用单行函数定义，或者用GW-BASIC和BASIC中没有的多行函数定义。用户只能在定义了DEF FN函数的模块中使用他们。

程序1—1使用了下面三个单行函数定义：

```
DEF FNcent#    把一个双精度数四舍五入成一个小数点后两位的数。
DEF FNmin!     找出两个单精度数中最小的一个。
DEF FNsrch%    为某一子串寻找一个串。
```

注意，DEF FN函数必须在该函数使用之前定义，即你必须把该函数定义放在源文本中引用该DEF FN函数的任何语句之前。你也许想把函数定义集中在一起放在程序开头，在它们中任一个被使用之前。

当运行程序1—1时，它首先要你输入两个双精度数。你输入的数被赋给变量first#和

esconcl#。这两个变量用于DEF FNcent#中把数四舍五入成小数点后两位的数。

然后，程序1—1要你输入两个单精度值。你输入的值分别被赋给n1! 和n2!。单精度变量n1!和n2!用于DEF FNmin!中。这个函数识别n1!和n2!中的最小值，这个最小值四舍五入后的双精度值first#和second#的值一起，以单精度值显示于屏幕上。

```
REM ** DEMONSTRATION OF DEF FN **
' Program 1-1  File: PR00101.BAS

REM ** Define Function FNcent# **
' Rounds a double precision number to two places
DEF FNcent# (mn#) = SGN(mn#) * INT(100 * ABS(mn#) + .5) / 100

REM ** Define Function FNmin! **
' Finds the minimum of two single precision numbers
DEF FNmin! (n1!, n2!) = n1! * ABS(n1! <= n2!) + n2! * ABS(n1! > n2!)

REM ** Define Function FNsrch% **
' Searches a string for a substring
DEF FNsrch% (st$, sbst$) = INSTR(UCASE$(st$), UCASE$(sbst$))

REM ** Input Numbers **
DEFINT A-Z: CLS
INPUT "First double precision number, please "; first#
INPUT "Second double precision number, please "; secnd#
PRINT : INPUT "First single precision number, please "; n1!
INPUT "Second single precision number, please "; n2!
PRINT

REM ** Call Numeric Functions and Print Results **
PRINT "Rounded value of first number is "; FNcent#(first#)
PRINT "Rounced value of second number is "; FNcent#(secnd#)
PRINT
PRINT "Minimum of single precision numbers is "; FNmin!(n1!, n2!)
PRINT

REM ** Input Strings, Search, and Print **
LINE INPUT "String to be searched, please? "; st$
LINE INPUT "String to search for, please? "; sbst$

REM ** Call String Function and Print Results **
PRINT : found = FNsrch%(st$, sbst$)
IF found = 0 THEN
  PRINT CHR$(34); sbst$; CHR$(34); " not found in the string."
ELSE
  PRINT CHR$(34); sbst$; CHR$(34); " starts at position";
  PRINT found; "in the string."
END IF
END
```

Program 1-1. Demonstration of DEF FN

最后，它要你输入一个串，然后输一个子串。计算机在输入的串中搜索子串。你可用上档、下当或上下档混合形式输入这个串。搜索与上下档形式无关。如果未找到子串，则计算机在屏幕上显示这一事实。如果找到了子串则计算机显示子串在串中开始的位置。

搜索功能由DEF FNsrch%执行，它把变量st$用于串，把变量sbst$用于子串。这三个函数定义在下表列中：

DEF FNcent#(mn#)=SGN(mn#)* INT(100×ABS.mn#) +.5) /100

DEF FNmin! (n1! , nz!) =n1! *ABS (n1! 〈 =nz!) +nz! *ABS (n1n! 2!)

DEF FNsrch% (st$, sbst$) =INSTR (UCASE$ (st$) 〈 UCASE (sbst$))

程序1—1的典型输出显示于图1—1中。

```
First double precision number, please ? 1234567.890123
Second double precision number, please ? 354.8651234567789

First single precision number, please ? 123.4567
Second single precision number, please ? 198.1234

Rounded value of first number is  1234567.89
Rounded value of second number is  354.87

Minimum of single precision numbers is  123.4567

String to be searched, please? This is the string to be searched.
String to search for, please? Ring

"Ring" starts at position 15 in the string.

Press any key to continue
```

Figure 1-1. Output of Program 1-1

QuickBASIC的多行函数定义比单行函数定义有力得多。你能象下表中显示的那样把一块语句放在函数的名字和参数表行与函数尾之间。竖直的省略号代表定义和函数的一块语句。

```
DFE FNfunctionname (parameterlist)
⋮
END DEF
```

程序1—2使用了两个多行函数来定义一个不能压缩为单行的更复杂的函数。

DEF FN函数定义中使用的变量由系统设置为对当前模块是全程的。但你可把 DEF FN函数中的变量放在一个STATIC语句中而使之成为局部的。在程序1—2的函数定义中没有使用STATIC语句，所以，定义中所用的全部变量都是全程的。

函数定义通常含有一个在函数名后括号里的参数表。参数是一个含有传递给函数的参量的变量名。在DEF FN函数中进来的参数都被保护以防修改。下表说明变量string $在程序1—2中用作参数。

```
DEF FNcount% (strng$)
DEF FNsqueeze$ (strng$)
```

调用函数的语句含有一个参量表。一个参量是一个调用函数时传给和函数的常量，变量或表达式。在QuickBASIC语句中使用DEF FN函数即可调用它们。变量由参量表中的一个值传递给DEF FN函数。从技术上讲，当你使用一个DEF FN函数时，就会创建变量的一个暂时拷贝，而且随后该拷贝的地址被传递给该函数。

正如下表所示，程序1—2调用每个DEF FN 函数，并把你输入的串 teststring $的值传递给每一个函数。

```
PRINT "The word count is", FNcount% (teststring$)
P RINT FNsqueeze$ (teststring$)
```

```
REM ** WORD COUNTER & SPACE SQUEEZER **
' Program 1-2  File: PR00102.BAS

REM ** Define FUNCTIONS **

' FNcountX counts words in a string
DEF FNcountX (strng$)
  words = 0
  FOR num = 1 TO LEN(strng$)
    char$ = MID$(strng$, num, 1)
    IF char$ = " " THEN
      words = words + 1
    END IF
  NEXT num
  FNcountX = words + 1
END DEF

' FNsqueeze$ squeezes spaces from a string
DEF FNsqueeze$ (strng$)
  squeeze$ = ""
  FOR num = 1 TO LEN(strng$)
    char$ = MID$(strng$, num, 1)
    IF char$ <> " " THEN
      squeeze$ = squeeze$ + char$
    END IF
  NEXT num
  FNsqueeze$ = squeeze$
END DEF

REM ** Use the Function Definitions **
DEFINT A-Z: CLS
LOCATE 2, 10: PRINT "Type Q and pres's ENTER to quit."
VIEW PRINT 5 TO 24
DO
  LINE INPUT "String, please ? "; teststring$
  IF UCASE$(teststring$) = "Q" THEN
    EXIT DO
  END IF
  PRINT
  PRINT "The word count is "; FNcountX(teststring$)
  PRINT : PRINT "Your squeezed string is"
  PRINT FNsqueeze$(teststring$)
  PRINT
LOOP
VIEW PRINT: CLS
END
```

Program 1-2. Word Count and Space Squeezer

注意：程序1—2中使用了块IF…THEN…ELSE语句。这种格式提供了一种清楚可读的结构。在称为"使用函数定义"的程序部分，包括了一个EXIT DO语句，以允许按下Q键时从DO…LOOP中有序地退出。DO…LOOP语句也可在DO或LOOP语句中用UNTIL或WHILE语句。

VIEW PRINT语句用于指定文本屏幕的一个"卷动"区域。首先在第二行打出指令"Type Q and press ENTNR to quit"打Q并按ENTER退出）。然后VIEW PRINT语句指定行5到24作为用于未来显示打印的卷动区域。所以，当这个区域填满后，该指令不会被卷走。就在程序结束之前，另一未指定任何行的VIEW PRINT语句把卷动区域扩展为全屏幕。

```
      Type Q and press ENTER to quit
String please? This is a test string to count words and squeeze spaces,
The word count is 11
Your squeezed string is
This is ateststring to countwords and squeeze spaces,
String, Please?
```

Figure1-2 Program2 : first string

程序1-2的典型输出显示于图1-2和1-3中。

1.4.2 FUNCTION…END FUNCTION过程

多行DEF FN函数改进了单行 DEF FN 函数定义。然而FUNCTION…END FUNCTION过程提供同样的功能且有其它长处。

所有在FUNCTION过程内使用的变量对该过程都是局部的。然而，你可用SHARED语句或其它语句的SHARED属性来使用全局变量。

```
      Type Q and press ENTER to quit,
String, please?This is a test sentence to count words and squeeze spaces
The word count is 11
Your squeezed tring is
This is a test sentence to countwords and squeezes paces,
String, please? Now, here is a second sentence.
The word count is 6
Your squeezed string is
NOW, here is a scond sentence,
String, please?q
```

Figure 1-3. Program 1-2 ready to quit

程序1-3使用了两个FUNCTION过程。除了开始和结束行外，FUNCTINN 中的语句块与程序1-2的 DEF FNsqueeze$ 函数定义中的一样。下表表明了开始行和两个结束行间的差别：

```
FUNCTION Squeezer$ (String$)          DEF FNsqueeze$ (Strng$)
    ⋮                                     ⋮
 Squeezers=Squeeze$                    FNsqueeze$=squeeze$
END FUNCTION                          END DEF
```

FUNCTION过程由后随过程名的关键字FUNCTION开始（前面的例子中是FUNCTION Squeezer$)。第一行也可包含一个含有传向或传自FUNCTION的变量的参数。在这个例子中只传递了一个变量string$。

FUNCTION过程在它的名字中返回一个单个值，所以FUCTION的名字的类型必须与其返回一致，在前面的例子中，串Squeezer$被返回。在这个例子中，返回的值（Squeezer$）在最后一行的前一行被赋值。FUNCTION名字(Squeezer$）必须与返回值的名字（Squeezer$）匹配。

FUNCTION过程尾必须用END FUNCTION语句说明。

通过在一个表达式中使用其名字的办法可以访问一个FUNCTION过程。下表表明了三种访问名为"Squeezer$"的FUNCTION和传递变量Array$（num）的方法：

```
After$ (num) =Squeezer$ (Array$ (num))
```

PRINT Squeezer$(Array$(num))
IF Squeezer$(Array$(num))=END THEN PRINT

程序1-3使用上表中第一种形式调用 FUNCTION Squeezer$。Squeezer$过程的值被赋给变量After$(num)。被传递到该过程的值是串Array$(num)。它刚被输入。

```
DECLARE FUNCTION Squeezer$ (strng$)
DECLARE FUNCTION Count% (strng$)
REM ** WORD COUNT & SQUEEZER WITH FUNCTIONS **
' Program 1-3  File: PRO01Q3.BAS

REM ** Get Array Size and Dimension Array **
DEFINT A-Z: CLS
LOCATE 2, 2: INPUT "How many strings"; size
REDIM Array$(1 TO size)
REDIM After$(1 TO size)

REM ** Print Instructions, Set Viewport, Get Strings **
CLS : LOCATE 2, 10
PRINT "Type Q and press ENTER to quit."
VIEW PRINT 5 TO 24
FOR num = 1 TO UBOUND(Array$)
  LINE INPUT "String, please ? "; Array$(num)
  IF UCASE$(Array$(num)) = "Q" THEN
    EXIT FOR
  END IF
  After$(num) = Squeezer$(Array$(num))
  WordCount = Count%(Array$(num))
NEXT num
VIEW PRINT: CLS : num = num - 1
PRINT "The total word count is"; WordCount
PRINT
FOR num = 1 TO UBOUND(Array$)
  PRINT After$(num)
NEXT num
END

FUNCTION Count% (strng$) STATIC
  words = 0
  FOR num = 1 TO LEN(strng$)
    char$ = MID$(strng$, num, 1)
    IF char$ = " " THEN
      words = words + 1
    END IF
  NEXT num
  RunCnt = RunCnt + words + 1
  Count = RunCnt
END FUNCTION

FUNCTION Squeezer$ (strng$)
  squeeze$ = ""
  FOR num = 1 TO LEN(strng$)
    char$ = MID$(strng$, num, 1)
    IF char$ <> " " THEN
      squeeze$ = squeeze$ + char$
    END IF
  NEXT num
  Squeezer$ = squeeze$
END FUNCTION
```

Program 1-3. Word Count and Space Squeezer with FUNCTIONS

这说明一个DEF FN函数定义和一个FUNCTION过程的实现十分不同。把一个DEF FN定义作为一主程序的一部分输入，把一个FUNCTION过程作为主程序的一部分输入，但它们是单独列出且从View菜单看。

FUNCTION过程是由从Edit菜单中选择New FUNCTION选择项而输入的。然后一个对话箱出现在你输入即将使用的FUNCTION…FND FUNCTION名外。在你的输入之后，QuickBASIC提供该过程的第一行和最后一行的一部分。对Squeezer$FUNCTION，QuickBASIC自动提供下面的行：

```
FUNCTION Squeezer$·
END FUNCTION
```

然后你就可完成第一行并输入此FUNCTION过程的剩余部分。

程序1-3的 Count%FUNCTION过程说明了一个选择功能特性。注意此过程第一行尾的STATIC这个字，它显示于下面：

```
FUNCTION Count% (strng$)  (STATIC)
```

这个选择的STATIC属性指明FUNCTION的局部变量将在对FUNCTION的各次调用之间被保存。各串的字计数器在Count%过程中计算并赋给变量RunCnt。

```
RunCnt=Runcnt+words+1
```

RunCnt的值在调用之间被保存，以便附加串的计数能被加上以保持一个当前的总数。在对比函数的是后调用之后，RunCnt的值是所输入的全部串的总字计数器。

Squeezer$ FUNCTION中不需要STATIC 属性。在那个过程中，每一串都被赋给一个不同的变量。

在程序1-3中你输入的串被作为元素赋给数组 （Array$），该数组由能按照你的输入动态地赋给数组大小的REDIM语句说明其大小。另一数组（After$）用于最后压缩了的串。下面的表说明这个数组怎样被动态确定大小的：

```
REDIM Arrays (1To size)
REDIM Arfte $ (1To size)
```

含有数组语句的FOR…EXIT循环用QuickBASIC的UBOUND函数来定义循环计数器的上限。下面给出了它的一个例子：

```
,FOR num=1 to UBOUND (Array$)
```

输入并运行程序1-3。图1-4显示该程序的典型输入的一个例子，而图1-5显示结果输出。

```
    Type Q and press ENIER to quit,
String, please? This is a test demonstration of program1-3,
string, please? It accepts stnings entered from the keyboard,
String, please? It counts the total number of words e ntered
String, please? In the strings and prints the strings as
String, please? squeezed strings,
```

Figure1-4. Five strings for program1-3

```
The total word count is  33
This is a test demenstration of Program1-3,
It accepts strings entered from the keyboard,
It  counts the total  number of words entered
In  the strings and prints the numstrings  as
Squeezedstrings,
Press any key to contiune
```

Figurc1-5. Output of program1-3

1.4.8 SUB…END SUB过程

SUB过程的开始行和结束行与FUNCTION过程的类似。下一个程序中将使用的SqzCount SUB过程有下面的开始和结束行：

```
SUB SqzCount (Array$ ())
    :
END SUB
```

Sqzcount中没有用STATIC属性，因为该SUB过程只被调用了一次。这样，字计数器就不必为产生一个现行使用字总数而保存。

当你在决定是否用SUB和FUNCTION 过程使用STATIC选择时，一定要考虑到下面几点：

1. 使用STATIC后，变量访问快得多。

2. 访问一个非STATIC SUB或FUNCTION过程要求花更多的时间，因为变量空间必须分配在堆栈上。

3. 有许多变量的大的SUB和FUNCTION过程使用许多堆栈空间，而且常常要求使用CLEAR语句。这使用了在其它情况下将分给串的空间。

注意：字符的压缩和字的计数是在同一个IF…THEN…ELSE块中进行。这两个操作在程序1-3中执行于不同的FUNCTION过程。

SUB过程中的任何变量或数组都被认为是局部于该子程序的，除非在SHARED语句中明确说明它们是共享的变量。数组After$说明于该子程序内部一个SHARED语句中。变量words被主程序中DIM语句里的SHARED属性说明为共享的。

```
    :
DIM SHARED words As INTEGER 'in main program
    :
SHARED After$ ()                'in SUB
```

由于words和After$ ( ) 两者都被说明为共享的了，所以它们不必出现在CALL语句的参量中或SUB过程的参数表中。然而，数组Array$并未被说明为共享的。所以它在CALL语句的参量表和SUB过程的参数表中都出现了。

程序1—4用一个单个SUB…END SUB过程作程序1-3中两个 FUNCTION 过程的工作。

从Edit菜单选择NewSUB选择项，你就可以输入一个SUB过程。随后一个对话箱出现在你输入将要使用的SUB…END SUB名处。QuickBASIC提供新SUB过程的第一行和最后一行，就象它对FUNCTION过程所作的一样。

SUB过程的调用不同于FUNCTION过程调用。你不能够用单一表达式中使用其名字的办法来调用SUB过程。对SUB的调用是个独立的语句。但是，你能有两种调用SUB过程的办法。第一种是用一个含有该SUB过程名的CALL语句。第二种是把该SUB过程名的本身作为一个语句。在下面对SUB过程Sqzcount (Array$ ( )) 的列表中两种方法的说明：

```
CALL SqzCount (Array$ ())    ' Array$ () enclosed in parentheses
SqzCount Array$ ()           ' Array$ () not enclosed
```

注意：在第二种形式中，数组没有被括在括号里。如果你略掉了关键字CALL(如在第

种形式中那样）而你又不是在用QuickBASIC写程序，那么你必须在你的程序中引用过的用DECLARE语句说明该过程。

本书用CALL语句（例子中的第一种形式）告诉你当你浏览程序清单时一个SUB过程正被调用。

```
DECLARE SUB SqzCount (Array$())
REM ** WORD COUNT & SQUEEZER WITH SUB **
' Program 1-4   File: PR00104.BAS

REM ** Initialize **
DEFINT A-Z
DIM SHARED words AS INTEGER
CLS : words = 0
LOCATE 2, 2: INPUT "How many strings"; size
REDIM Array$(1 TO size)
REDIM After$(1 TO size)

REM ** Print Instructions, Set Viewport, Get Strings **
CLS : LOCATE 2, 10
PRINT "Type Q and press ENTER to quit."
VIEW PRINT 5 TO 24
FOR num = 1 TO UB UND(Array$)
  LINE INPUT "String, please ? "; Array$(num)
  IF UCASE$(Array$(num)) = "Q" THEN
    EXIT FOR
  END IF
NEXT num
CALL SqzCount(Array$())
VIEW PRINT: CLS : num = num - 1
PRINT "The total word count is"; words
PRINT
FOR num = 1 TO UBOUND(Array$)
  PRINT After$(num)
NEXT num
END

SUB SqzCount (Array$())
  SHARED After$()
  FOR str = LBOUND(Array$) TO UBOUND(Array$)
    squeeze$ = ""
    FOR num = 1 TO LEN(Array$(str))
      char$ = MID$(Array$(str), num, 1)
      IF char$ <> " " THEN
        squeeze$ = squeeze$ + char$
      ELSE
        words = words + 1
      END IF
    NEXT num
    After$(str) = squeeze$
    words = words + 1
  NEXT str
END SUB
```

Program 1-4. Word Count and Space Squeezer with SUB