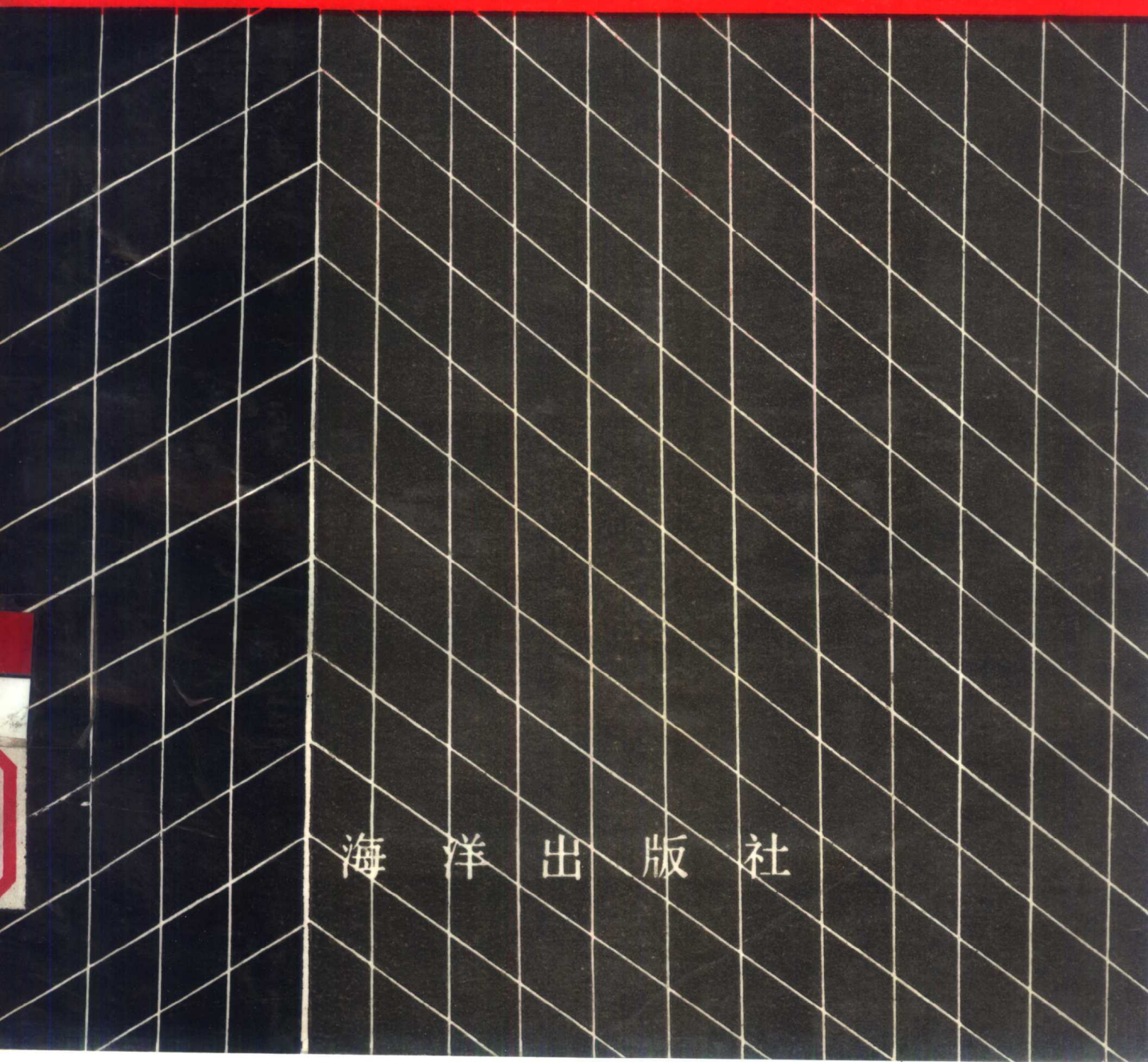
秦笃烈 编译

中国科学院希望高级电脑技术公司程序设计丛书

Quick BASIC
高级软件开发工具包



海洋出版社

中国科学院希望高级电脑技术公司程序设计丛书

Quick BASIC

高级软件开发工具包

秦笃烈　编译

海洋出版社

一九九〇年二月

内　容　摘　要

本书是在QuickBASIC环境下开发的一系列高级软件工具包和程序，内容极为丰富，共包含260个以上例行程序清单，总量在17000行以上，共计四十多个软件工具包和程序，发展的BASIC语言已能满足现代程序设计要求，兼顾专业程序设计和广大非计算机专业用户的需要，书中所列混合语言工具包，开辟了C和QUICKBASIC子程序调用的新途径。本书所含软件有很高实用价值。

Quick BASIC高级软件开发工具包

秦笃烈　编译

责任编辑　刘莉蕾
特约编辑　鞠玉兰

海洋出版社出版（北京市复兴门外大街1号）
北京建华印刷厂印刷
开本：787×1092　1/16　印张：24.5　字数：594千字
1990年2月第一版　1990年2月第一次印刷
印数1—3000
ISBN7—5027—0877—4/TP·16　定价：15.50元

序　言

BASIC语言风靡全世界为计算机的应用和教学起了不可磨灭的作用。计算机科学作为新兴学科的一个显著特征是设备和知识更新十分迅速。BASIC语言在这种背景下本身也在不断发展。1964年5月1日问世时BASIC只有14条语句。在BASIC的发明人G·Kemeny和E.Kurtz的努力下，BASIC已有了美国国家标准。它的微型版本（适用于IBM—PC和Mackintosh）True BASIC吸取了Pascal，C，Ada语言的优点，同时保留了BASIC本身的独特优点。它既便于初学者开发程序也适合于专业程序设计人员编制精巧而复杂的程序。

美国著名的Microsoft公司根据它本身所具的软件实力优势推出QuickBASIC，在环境、功能、效率方面也有不少新意。本书所介绍的大量软件包体现了该公司的最新软件产品水平。本书中的许多程序虽然是在Quick BASIC环境下写出，但是不少内容稍加修改即可用True BASIC或另一个BASIC新品种Turbo BASIC。从这个意义上说，本书的软件包是对今后新型BASIC语言推广应用的有力软件支撑。我们祝愿读者能在实用价值和软件技巧方面获益于本书。

我要感谢希望公司和海洋出版社的合作精神使本书能如此迅速问世。对鞠玉兰和刘莉蕾同志在编辑工作中所付劳动深表感谢。

计算机基础教育研究会谭浩强、刘瑞挺教授对本丛书的出版给予热忱支持，在此谨向他们表示衷心感谢。

本书的发行工作由中科希望高级电脑技术公司负责。需要的读者可直接与该公司联系（地址：北京市海淀区黄庄8721信箱资料部，邮码：100080　电话：2562329）

秦笃烈　1989.12

首都医学院生物医学工程系

北京右安门外，邮码：100054

目　　录

第三部分　混合语言软件工具包

第四部分　附录

第一部分 预 备 知 识

第一章 QuickBASIC和软件工具包

感谢Microsoft QiuckBASIC 4.0的出现，使BASIC终于发展成为一种灵活的，功能齐全的，并且是强有力的程序设计语言。只要信手翻阅本书并且浏览一下程序清单，读者一定会看到BASIC已是今非昔比了。Microsoft QuickBASIC 容易理解，具有快速学习曲线，并且赋予读者很快就能创作复杂程序的能力，这种复杂程序用传统的BASIC是难以实现的。

在传统BASIC和QuickBASIC之间的主要差别在于QuickBASIC允许结构化程序设计，这是能够创作大程序并容易维护的重要特征。

第一节 结构化程序设计的优点

对BASIC的早期版本来说，程序的书写和执行都是针对由程序行组成的单个段块进行。经验不足的程序设计人员在写作较大程序时不自觉地会编出所谓“拉面条代码”（指频繁而不恰当地使用GOTO语句的程序），这样的程序既难以理解也不容易维护。

QuickBASIC的主要特征是它具有创作结构化程序的能力，所谓结构化，主要是指大程序可以分解为若干小的程序模块。用QuickBASIC进行程序设计就不必着眼于构造一个包罗万象的大程序段块，而是构造一些各司其职的程序模块，程序模块又可以通过称为子程序和函数的过程进行构造。每一个这样的过程都是完成一项特定的、定义得很清楚的任务。由于集中精力考虑一个过程的功能，程序设计人员就不必为程序的其它部分操心而把全部注意力放在手头任务上。实际证明，利用这种模块化处理方法，程序设计人员能够更快更准确地开发复杂的程序。

结构化程序设计的另一个好处是这些模块和过程可以用这样一种方式组织和保存，即它们可以为其它程序重复利用，从而避免了程序设计中的重复性劳动。假如把具有互相补充的功能的模块进行分组，程序设计人员就不难建立有用的例行程序软件工具包，可以使得大型程序设计工作进展迅速，因为程序的主要部分已写出来了。

在模块构造出来以后，也可以进一步将它组织成一个Quick程序库，这是以特殊格式贮存在磁盘上的文件。以后就可以把这个程序库和QuickBASIC程序一起装入，这样就等于把程序库中的例行程序变成QuickBASIC业已存在的例行程序的一部分，使之大为扩充。

第二节 本书的软件工具包简介

对于正在使用或者想用QuickBASIC的读者，本书是一本有价值的参考书。对于刚刚开始作为第一语言学习QuickBASIC的读者，本书可直接作为学习语言必要的实例使用。

对于用QuickBASIC作为软件开发系统富有经验的专业程序设计人员来说，本书中的例行程序无疑是对QuickBASIC语言极有价值的扩充。

第一部分提供了构造一个称为MINICAL的完整工作程序的步骤说明，特别是初步程序设计人员会发现这对于教学极有裨益。第二部分包含全部QuickBASIC软件工具包，并对如何装入和运行作了简要说明。第三部分介绍混合语言工具包的用法并包含几个例子。最后，五个附录包含了有关软件工具包运行要求，函数和子程序的参照的信息以及附加的重要信息。

对于有经验的程序设计人员，可以从第二部分开始并使用某些软件工具包。对于QuickBASIC的初学者，可接着阅读下一章。从中将可用自行建造两个包含函数和子程序的模块并用它们建造一个程序库QuickLibrary，也可用第二和第三部分中的模块以及自己设计的模块制作更多的Quick程序库。读者从本书将可以得到许多强有力的软件工具包使编制程序工作更快更好。

第二章　MINICAL.BAS——一个完整的程序

这一章我们从零开始建造一个完整的工作程序。我们要对程序进行构筑，建立一个Quick程序库扩充QuickBASIC语言并建立一个独立的可执行程序。

在开始工作以前，我们要先对QuickBASIC程序设计环境和涉及到的某些重要概念作一些简要说明。本章的样本程序是由两个不同的模块构成的，每一个都由几个子程序和函数组成。我们要观察Quick BASIC将如何处理这些内容。

第一节　源代码的模块化编辑

Microsoft QuickBASIC4.0的新特征之一是在QuickBASIC环境内复习和编辑程序的方法的新颖性。如果已在用Quick BASIC进行程序设计，大概已注意到一个包含几个子程序和函数的程序不可能在屏幕的一部分观察和编辑全部内容。如果尚未用Quick BASIC编程序，现在只需知道可以从当前装入的子程序和函数清单中选择一个函数或子程序来进行观察和编辑。其它所有例行程序都隐藏起来。初看起来，这似乎令人奇怪，但在对几个程序进行工作以后，就会体会到这种模块化编辑所具有的威力。

QuickBASIC 4.0第二个重大进展是它具有一次能把多个源代码文件装入内存的能力。这就为建立多组子程序和函数打开了大门，这些子程序和函数按照专题贮存在分别的源代码文件中，几个不同的程序可以独立地装入和使用它们。

关于这些新特征的重要概念可以概括如下：一个程序可以由一个或多个源代码文件（模块）组成，而每个源代码文件可以由一个或多个子程序或函数组成。可以把若干个源代码文件同时装入内存，所有模块可以在一起工作构成一个完整程序。虽然一次只能显示和编辑源代码文件的一部分，但很容易在对程序进行编辑时从一部分跳到另一部分。

第二节　建造一个程序库Quick Library

如果自己能创造一些新的QuickBASIC语句和函数，而且在每次启动QuickBASIC以后就能使用这该多好啊！程序库QuickLibrary正是要做这件事！

例如，假定在所写的几乎每个程序中都要用双曲函数，就可以用源代码文件形式制作这些函数并和所写的主程序一起装入内存。但还有另一种方法，就是建立一个程序库Quick Library使得这些函数在装入QuickBASIC的同时也装入内存。要建造一个程序库Quick Library是非常简单的，只需把双曲函数源代码文件装入并从Run菜单选择适当的菜单选项就行。

第三节　为MINICAL建立源代码

我们要一步一步在自己参与下经历完成一项完整的程序设计任务的全过程。如果打开计算机边学边做好处最大。程序MINICAL完成五种功能：加，减，乘，除以及平方根。这个程序在范围上是简单的，但是它具备了更大得多的软件项目所具有的重要组成部分。

在开始进行代码设计以前，先看一下程序完成以后它的运行情况。程序MINICAL使用了逆波兰记法（RPN）作为输入。借助RPN可使程序设计工作大大简化，因为它不需要为重新安排隐藏在括弧中的数学命令而进行代码设计工作。

用RPN，要先输入数，其后跟运算符。例如，3加4，可输入

34＋

（在下面我们会说明这些数实际上是如何输入的）。要算1加2然后将结果乘3，可输入：

12＋3＊

MINICAL使用双精度数，所以可以输入任何种类的整数或浮点数。结果最多可显示16位数字。例如，1.2除以－3.45，可输入

1.2 -3.45／

其结果的显示的形式是：

Result...-.3478260869565217

注意：因为计算机键盘上没有×和÷键，星号（＊）用来表示乘号而正斜杠（／）表示除。

MINICAL使用称为存贮栈的结构存放数和运算符同时完成计算（从技术上说，MINICAL所用的结构只是模仿了传统的堆栈，但为讨论起见，可以将它设想为存贮栈），一个存贮栈是开辟的一系列内存位置以存放一些不同的项目，在这里是存放由用户提供的数和运算符。利用RPN表示数和运算符的理由主要是因为存贮栈的存在，因为RPN语法对基于存贮栈的计算是理想的。另外的方法是语法分析，它涉及到“读懂”由用户输入的方程，重新安排和选择各个元素，并对它们施加动作。后一种方法需要花更多的代码设计工作。

MINICAL中的存贮栈能存放20个值之多再加上有关的运算符。可以按上面所说方法输入数值，或者也可以先输入所有的数，然后输入运算符。例如，把1到5加起来，可以用下面两个命令行中的任意一个：

12＋3＋4＋5＋

1 2 3 4 5 ＋＋＋＋

一、模块MINIMATH

这一项目共有两个组成部分：模块MINIMATH和模块MINICAL。我们先从MINIMATH开始。

在系统提示下打入QB并按Enter就可启动QuickBASIC。按照下面格式打入标题段。

```
' ********************************
'   Name:          MINIMATH
'   Type:          Toolbox
'   Module:        MINIMATH.BAS
'   Language:      Microsoft QuickBASIC 4.00
' ********************************
'
' Collection of math subprograms for the MINICAL
' Program.
```

这时，告诉Quick BASIC这个模块的名称是很容易的，下拉File菜单并选择Sale As。打入文件名MINIMATH并按Enter键或将鼠标器指针移到OK框并触发鼠标器左按

钮。然后该文件就存盘，可以准备继续输入程序的其它部分。注意，这里省略扩展名.BAS，Quick BASIC会自动加上。

这第一个模块是由完成数学函数运算的五个子程序组成。先来建立子程序Add。下拉Edit菜单并选择New SUB。当询问子程序名时，响应Add。这时出现QuickBASIC新子程序的第一和最后的一行：

```
SUB Add
END SUB
```

注意，QuickBASIC也会调整编辑窗口使得只显示子程序Add，使用户仅仅把注意力集中在这个子程序上（当程序编得越来越大时，对这一特征就会深有体会了）。下一步是在子程序的第一行前面加一些注释信息并在由QuickBASIC显示的两行之间插入子程序的“五脏六腑”。

先从加注释开始。将光标移到子程序的第一行的第一字符。然后按Enter键。这时，出现一个带有报文Blank lines not allowed before SUB/FUNCTION line.Is remark OK? 的对话框。因为要求注解，所以选OK。然后QuickBASIC插入一个前面带有字符'的空白行提供注释。在打入第一个注释行（内容见下面的程序清单）以后就按Enter，对话框再次出现。触发OK并接着打入下一注释行。

利用其后的行建立子程序Add。注意SUB Add和END SUB行的位置，并注意要加一些内容到SUB Add行。还要注意，某些行已缩入：虽然并不要求这样的缩入，但把子程序行缩入是好的程序设计作风，可以使程序易读并使各行之间的关系看起来很分明。当做完以后，子程序Add有如下的样子。

```
'************************************
'***   Name:        Add             **
'***   Type:        Subprogram      **
'***   Module:      MINIMATH.BAS    **
'***   Language:    Microsoft QuickBASIC 4.00  **
'************************************
'
' Performs addition for the MINICAL program.
'
  SUB Add(stack#(),ptr%) STATIC
      ptr%=ptr%-1
      IF ptr% THEN
          stack#(ptr%)=stack#(ptr%)+stack#(ptr%+1)
      END IF
  END SUB
```

这时可以把所做工作存起来（经常把已做的工作保存备分以防止停电或其它故障造成的损失是很好的工作习惯）。为此，下拉File菜单选择Save All。

如果以上工作都很顺利，说明到目前为止学得很不错。可用同样的方法建立和编辑其它四个数学计算子程序。

还有一个可以使这个过程加快的诀窍。要注意，五个子程序的开始注释行几乎完全一样。所以不必每一次都重打，可以把对子程序Add打入的行进行拷贝，然后加到其它子程序中去。下面就来做这件事。

首先，建立各个新的子程序，对子程序Subtract，从Edit菜单选择New SUB，输入名称Subtract，并按Enter。在子程序Subtract的两行显示出来以后，对子程序Multiply，Divide和SquareRoot重复New SUB的过程。务必要打入正确的子程序名，每个名称以大写字母开始，在子程序名称SquareRoot中不允许有空格。

其次，对子程序Add中的注释行作一分拷贝。下拉View菜单并选择SUBS。Quick BASIC显示已经输入的全部子程序名称的清单。正是在这里QuickBASIC的威力体现得非常清楚。利用SUBS命令，通过选择所需要的子程序名就可以跳到任何要进行编辑的子程序。方法是用鼠标器在需要的子程序名上按触发键两次或利用光标移动键然后按Enter键。现在，选Add。

为了要拷贝注释行，把光标移动到第一注释行的第一个字符，然后在键盘上按下Shift键并按下箭头键直到所有注释行都加亮为止；也可以利用鼠标器移动鼠标器指针到光标位置，按下鼠标器左按钮，并保持不动直到所有注释行加亮为止。最后，从Edit菜单选择Copy就把加亮的各行拷贝到Clipboard（剪板）。

接着，把这些行拷贝到其它四个子程序。从View菜单选择SUBS再一次显示子程序清单。选择Subtract，将光标移到第一行第一个字符，下拉Edit菜单并选择Paste。从子程序Add拷贝过来的各行这时应该出现（如果不出现或者如果得到一个对话框告诉说在子程序之前不能出现空行，则回到子程序Add并重复Edit-Copy过程。记住，只拷贝注释行，即前面有字符'的行）。

用同样的方法选择其余的子程序名称，重复Edit-Paste过程。不必再重复Copy操作，因为拷贝到剪板上的内容仍然还在，除非有新的内容拷贝上去或退出Quick BASIC。完成以后，回到每个子程序。编辑注释行，和子程序Add一样，输入程序行。在完成每个子程序的输入以后，务必从File菜单选择Save All。还有，要返回来并温习自己做的工作。利用View菜单上的SUBS选项可分别观看每个子程序的能力使这项审核工作更为容易进行，也使程序更加清晰易读。

注意，不要忘记对注释行的编辑！要记住，“粘入”的注释是用于子程序Add的注释。必须对每个注释段中的名称和注释下部行上的信息进行修改以反映出注释内容标识的子程序

以下给出其它四个子程序的源代码清单。下面是子程序Subtract：

```
'*****************************************
'***   Name:          Subtract           ***
'***   Type:          Subprogram         ***
'***   Module:        MINIMATH.BAS       ***
'***   Language:      Microsoft QuickBASIC 4.00  ***
'*****************************************
'
' Performs subtraction for the MINICAL program.
'
SUB Subtract (stack#(), ptr%) STATIC
    ptr% = ptr% - 1
    IF ptr% THEN
        stack#(ptr%) = stack#(ptr%) - stack#(ptr% + 1)
    END IF
END SUB
```

下面是子程序Multiply：

```
'*************************************
'**  Name:        Multiply            **
'**  Type:        Subprogram          **
'**  Module:      MINIMATH.BAS        **
'**  Language:    Microsoft QuickBASIC 4.00  **
'*************************************
'
' Performs multiplication for the MINICAL program.
'
  SUB Multiply (stack#(), ptr%) STATIC
      ptr% = ptr% - 1
      IF ptr% THEN
          stack#(ptr%) = stack#(ptr%) * stack#(ptr% + 1)
      END IF
  END SUB
```

下面是子程序Divide：

```
'*************************************
'**  Name:        Divide              **
'**  Type:        Subprogram          **
'**  Module:      MINIMATH.BAS        **
'**  Language:    Microsoft QuickBASIC 4.00  **
'*************************************
' performs division for the MINICAL program.
'
  SUB Divide (stack#(), ptr%) STATIC
      ptr% = ptr% - 1
      IF ptr% THEN
          stack#(ptr%) = stack#(ptr%) / stack#(ptr% + 1)
      END IF
  END SUB
```

下面是子程序SquareRoot：

```
'*************************************
'**  Name:        SquareRoot          **
'**  Type:        Subprogram          **
'**  Module:      MINIMATH.BAS        **
'**  Language:    Microsoft QuickBASIC 4.00  **
'*************************************
'
  Determines square root for the MINICAL program
'
  SUB SquareRoot (stack#(), ptr%) STATIC
      stack#(ptr%) = SQR(stack#(ptr%))
  END SUB
```

至此，程序MINICAL的第一部分已经完成。已经建立的子程序以MINIMATH 为名

存在磁盘上。因为它们完成实际计算，所以是MINICAL的核心。下一部分的工作要建立MINICAL本身。MINICAL完成管家工作——采集用户需要计算的数值，将它们传送给MINIMATH中适当的子程序，并且显示结果。

二、模块MINICAL

从现在起，我们要用两种可能方法的一种继续工作。因为MINIMATH和MINICAL这两个模块都很小，可以通过建立名为MINICAL的第二个模块来构筑程序MINICAL（做法是从File菜单选择Create File接受缺省选择Module，然后选择OK）。或者也可以把MINIMATH转到程序库QuickLibrary并用QuickBASIC系统将它装入，使它成为对该语言的扩充。我们将采用第二种方法以观察利用QuickBASIC的高级功能时这样做是何等容易。

建立Quick程序库的方法是，下拉Run菜单并选择Make Library（建库）。这时会要求给出程序库名称。MINIMATH是好名称，因为QuickBASIC会自动地把一个缺省扩展名.QLB赋给程序库QuickLibrary，而对通常的程序库所给的扩展名是LIB。在打入MINIMATH以后，选择MakeLibrary（这时可不管对话框中的其它选择），完毕以后，QuickBASIC会在当前目录中建立两个新文件：MIINMATH.QLB和MINIMATH.LIB。

现在退出Quick BASIC，重新启动它并用它装入新的Quick程序库。为此，下拉File菜单并选择Exit。从系统提示下打入：

QB␣/L MINIMATH

就重新启动QuickBASIC。虽然看不到有什么明显不同的迹象，实际上一件非常令人激动的事情发生了！QuickBASIC业已扩充，它现在比以前所具有的东西更多了。MINIMATH中的子程序已是QuickBASIC语言的一部分，可以象许多QuickBASIC其它关键字一样利用它们。事实上，因为在调用子程序时，关键字CALL是可省略的，所以这些新的子程序就更像是以新的关键字的形式出现在QuickBASIC中。

随着这个程序其余部分的逐渐引入，可以来试一下新的扩充后的QuickBASIC。务必用前面所说的方式装入QuickBASIC，将Quick程序库MINIMATH作为系统的一部分。然后打入如下程序，MINICAL.BAS：

这是程序MINICAL的主部，一切动作从这里开始。注意在DO-LOOP结构中的头两行，内容是parm$=NextParameter$(cmd$)和Process parm$,stack#(),ptr%。第一行调用用户定义的名叫NextParameter$的函数而第二行调用名叫Process的用户定义的子程序（不过，目前尚未对它们进行定义，这是要完成的任务清单上的下一项）。注意，调用子程序Process时并没有使用关键字CALL。如果愿意可以用CALL，但并不是非用不可。由于QuickBASIC处理子程序的方式，马上要定义的子程序Process更象是QuickBASIC系统的一部分而不象是程序的一部分，因为不能在程序已显示在屏幕上的同时将它列表或进行修改。也不必去想它或使它的重新编译！用户的创造力可以自由地去处理更高层次的程序复杂性。

一旦将主程序的全部内容输入以后，又该将这个模块存盘了。从File菜单选择Save As并输入文件名MINICAL。

```
' ********************************
' **  Name:       MINICAL       **
' **  Type:       Program       **
' **  Module:     MINICAL.BAS   **
' **  Language:   Microsoft QuickBASIC 4.00  **
' ********************************
' Functions
DECLARE FUNCTION Nextparameter$ (cmd$)
Subprogram $
DECLARE SUB Process (cmd$, stack# ( ), ptr% )
DECLARE SUB DisplayStack (stack# ( ), ptr% )
DECLARE SUB Add (stack# ( ), ptr% )
DECLARE SUB Subtract (stack# ( ), ptr% )
DECLARE SUB Multiply (stack# ( ), ptr% )
DECLARE SUB Divide (stack# ( ), ptr% )
DECLARE SUB SquareRoot (stack# ( ), ptr% )
' Get the command line
cmd$=COMMAND$
Create a pseudo stack
DIM stack# (1 TO 20)
ptr%=0
' Process each part of the command line
DO UNTIL cmd$=""
    parm$=NextParameter$ (cmd$)
    Process parm$, stack# ( ), ptr%
    IF ptr%<1 THEN
        PRINT"Not enough stack values"
        SYSTEM
    END IF
LOOP
' Display results
DisplayStack stack# ( ), ptr%
' All done
END
```

在能够对程序进行试验之前还有一些代码设计工作要做。为了要建这个程序的函数，从Edit菜单选择New Function。打入函数名NextParameter$，和按Enter键。建立和编辑函数实际上和建立和编辑子程序没有什么不同。事实上，在函数和子程序之间的唯一主要差别是函数返回计算中要使用的值或者是要赋给QuickBASIC变量的值，子程序仅仅通过传送的变量返回值。按照用于建立MINIMATH中子程序的同样步骤建立函数Next Parameter$：

```
'************************************
'***  Name:        NextParameter$    **
'***  Type:        Function          **
'***  Module:      MINICAL.BAS       **
'***  Language:    Microsoft QuickBASIC 4.00  **
'************************************
'
' Extracts parameters from the front of the
' command line. Parameters are groups of any
' characters separated by spaces.
'
  FUNCTION NextParameter$ (cmd$) STATIC
       parm$ = ""
       DO WHILE LEFT$ (cmd$, 1) <> " " AND cmd$ <> ""
            parm$ = parm$ + LEFT$ (cmd$, 1)
            cmd$ = MID$ (cmd$, 2)
       LOOP
       DO WHILE LEFT$ (cmd$, 1) = " " AND cmd$ <> ""
            cmd$ = MID$ (cmd$, 2)
       LOOP
       NextParameter$ = parm$
  END FUNCTION
```

现在将以下两个子程序作为模块MINICAL的一部分进行建立和修改。

建立子程序DisplayStack：

```
'************************************
'***  Name:        DisplayStack      **
'***  Type:        Subprogram        **
'***  Module:      MINICAL.BAS       **
'***  Language:    Microsoft QuickBASIC 4.00  **
'************************************
'
' Displays anything left on the stack when MINICAL
' finishes processing the command line.
  SUB DisplayStack (stack#( ), ptr%) STATIC
       PRINT
       IF ptr% > 1 THEN
          PRINT "Stack...",
       ELSE
          PRINT "Result...",
       END IF
       FOR i% = 1 TO ptr%
            PRINT stack#(i%),
       NEXT i%
       PRINT
  END SUB
```

接着建立子程序Process：