结构化 BASIC 语言
程序设计基础

姚 政 曙

中国 Apple 微机协会
《苹果园》编辑部

一九八八年十一月

结构化 BASIC 语言程序设计基础

姚　政　曙

中国 Apple 微机协会
《苹果园》编辑部

一九八八年十一月

结构化 BASIC 语言程序设计基础

姚政曙

中国 APPLE 微机协会《苹果园》杂志编辑部出版发行
（山东·潍坊东风大街 60 号）
山东·潍坊计算机公司电脑排版中心制版·印刷
开本 787×1092 毫米 1/16·5 印张
字数:237 千字
1988 年 11 月第一版　　1988 年 11 月第一次印刷
印数:1—5000 册

前　言

程序语言 BASIC 是最普及的语言，与其它语言相比，BASIC 在更多的计算机上运行，且世界上使用 BASIC 的人也最多。它是由美国达特茅斯（Dartmouth）学院的 Tohn Kemeny 和 Thomas Kurtz 两位教授在 60 年代中期提出来的，1971 年形成标准文本。研制 BASIC 语言的目的是想以其作为教学生如何编写程序的一种简单的面向数字计算的语言，因此它不可避免地具有许多“教育式”语言的特点：易学、易写、具有极强的人机交互性。比起其它语言，它能管理更多的任务，既能执行所有的 FORTRAN 计算任务，也能比 Pascal 等语言更好地处理字符串，还能执行绘图、通讯、以及低级存储处理等任务。BASIC 语言在世界电子计算机（尤其是微型计算机）的普及推广和应用中，曾经起过并且仍正在起着极其重要的作用。

随着计算机应用领域的日益广泛与应用程度的日趋深入，对软件的“可修改性、效率、可靠性与可理解性”成了现代软件工程所追求的目标。实现此一目标的基本途径是对程序进行模块化或结构化设计。在这种情况下，BASIC 暴露出了它的致命弱点：它是非结构化的程序语言。BASIC 从来就没有提供过程序结构化设计的最少支撑设施，这导致用 BASIC 作为工具编制的程序其可靠性与可维护性极差，因此，BASIC 成了当今计算机语言领域中的唯一一种集卓越优点与致命弱点于一身的最有争议的语言。

但是，BASIC 语言是有生命力的，这种生命力是自它诞生那一天起就具备了的。今天，一个结构化 BASIC 程序设计语言（以后简称该语言为 S—BASIC 语言）的问世，使 BASIC 掀开了它新的一页：S—BASIC 不仅具有它的已为世界上数百万用户认可并接受了的其它语言无法比拟的优点，而且它具有良好的程序结构化设计的支撑设施，在某些方面，它体现出了比 FORTRAN、Pascal 等语言更强有力的对现代软件工程思想和实施的支撑。

由于近二十年来 BASIC 语言没有为结构化程序设计提供支撑设施，因此，凡涉及 BASIC 语言的资料及书籍中也就从来没有向 BASIC 用户们介绍、讨论结构化设计的概念和语言的支撑设施及实现。

为了适应我国的计算机应用事业的发展，在 S—BASIC 语言解释系统设计的基础上，详细地向用户介绍 S—BASIC 语言所提供的结构化设计支撑设施，以及程序结构化设计方法的基本内容，以期使读者一开始就具有良好的程序设计习惯和风格。

比起其它语言，BASIC 有更多的版本和方言，结构化 BASIC 语言今后必然也如此，因为从一开始对它的设计，目标就是不妨害现有任一版本 BASIC 用户的既有软/硬件资源。实现这一目标的最佳途径是在原有目标机器的原配置非结构化的 ROM BASIC 版本支持下进行扩充。因此，可以期望，今后的结构化 BASIC 的各种版本，除了基本的结构化设计支撑设施语法是相同和类似的以外，各个版本在指令集包含的指令数量上和语法原语上会有所差别。

本书是以第一个结构化 BASIC 语言版本为目标写成的。这个版本名为 STRUC—BASIC<1.0>，它的目标机器是 APPLE II，软件支持环境是 DOS 3.3 操作系统（为支持子程序库的创立/使用而需要）以及 APPLESOFT BASIC（ROM BASIC）。它的指令集为两部

分：原 APPLESOFT BASIC 指令集和扩充后的结构化支撑设施指令集。原 APPLESOFT BASIC 指令集已有相当数量的专门书籍给予介绍，故本书对涉及这部分指令集的介绍作了简化与压缩且图形/游戏部分略，重点介绍涉及结构化支撑设施部分。APPLE 用户可把本书作为基本工具书，而对其它 BASIC 版本的用户，本书内容也有参考价值，相信不用多久，其它版本的 BASIC 用户也会具有相应版本的结构化 BASIC 语言系统以及与其相符合的基本工具书。

本书共分七章，第一章讨论了结构化 BASIC 程序的概念；第二章除 CASE 条件结构外都是原 APPLESOFT BASIC 语言所提供之指令的介绍，因此内容已被尽量的简化；第三章到第七章则为扩充后的结构化支撑设施（指令）的内容介绍。其中，第三章讨论了函数和子程序，第四章介绍子程序库，第五章讨论程序的分节与覆盖，第六章介绍程序的模块化设计方法，第七章专门集中介绍了支持子程序库设施的 DOS 3.3 扩充命令。这几章内容对大多数 BASIC 用户来说是第一次涉及，今后也将逐步熟悉并使用，所以是本书的重点。书末附有四个附录，附录四给出了十个常用算法标准子程序，这些子程序是完全的封闭模块结构的，全部可以入库。希望 BASIC 的用户从此后建立子程序库的概念并习惯使用标准子程序库，也相信广大的计算机算法专家们在 S—BASIC 语言支持下将会为广大用户提供更多的 S—BASIC 语言的标准子程序库以造福于 BASIC 用户们。

借此机会，感谢恩师严仰光教授和钟建业高级工程师，他们自始至终对结构化 BASIC 语言的问世给予了极大的关注和支持。

本书能顺利出版与广大读者见面，是 Apple 微机协会《苹果园》编辑部给予了热情的支持和帮助，在此表示衷心的感谢。

由于这是第一本详细介绍结构化 BASIC 语言的书籍，无参考资料可阅，谬误之处在所难免，恳请阅者赐教。

作者　一九八八年六月

目　　录

第一章　结构化 BASIC 语言概述

如果算上各种非标准版本，BASIC 是目前世界上应用最广泛的一种程序设计语言。但是，以往所有版本的 BASIC 都有一个致命的弱点：缺乏结构化语言的特点。

所谓结构化程序设计，是使程序具有一合理结构以便保证和验证其正确性而规定的一套程序设计方法。采用结构化程序设计方法以及与此相关的一些措施，可使程序的错误大为减少，缩短程序的研制周期，获得具有良好的可靠性、可读性、可维护性和可适应性的高效能软件，更深远的意义还在于：使程序设计从依赖于设计者技巧的活动变成一门科学。

仅依据语言提供结构化设计的支撑设施并不能保证设计的程序是结构化的。一些程序员用结构型语言编写的程序却完全不可理解，这是缺乏结构化程序设计的概念以及训练所致。BASIC 用户中绝大多数程序员是以 BASIC 为唯一基本语言的，因此更缺少结构化设计方面的知识和训练，更多的反而是编程上“随随便便”的坏习惯。本章介绍结构化 BASIC 程序的概貌以及某些基本概念，以期一开始就着意使读者具有良好的程序设计习惯和风格。

第一节　S—BASIC 程序举例

我们从设计和阅读一个简单的 S—BASIC 程序入手，开始结构化 BASIC 程序语言的学习。对于本节所举的程序例子，只要求读者通过阅读从而对结构化 BASIC 程序有一个全面的感性认识，以利于后继各章的学习。程序中所出现的各种 BASIC 语句的作用及其使用规则会在以后各章中分别介绍，这里可暂不深究。

例 1—1.

设任意输入三个实数 A、B、C，要求执行：

当 A＋B＞C 时，计算 (A＋B)！并打印出(A＋B)！＝××；

当 A＋B＜C 时，计算 C！并打印出 C！＝××；

当 A＋B＝C 时，只打印出“ERROR”的信息。

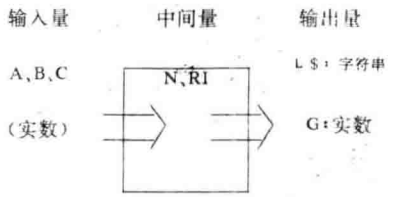
上述三种情况执行后均应回到输入三个实数 A、B、C 处重复执行上述操作。

1. 分析

这个问题的要求是明确的。首先，实数 A、B、C 是输入参数，要计算 A、B 的和并与 C 比较，因此最好有一个存放 A、B 之和的参数(实数)，我们取名为 N，从而变成 N 与 C 的比较；其次，比较结果有三种：当 N＞C 时，计算 N 的阶乘，当 N＜C 时，计算 C 的阶乘，当 N＝C 时，不计算阶乘。在计算完毕后应该根据三种情况分别打印，打印的内容应有字符串与数值两者，因此最好有一个存放字符串的变量，我们取名为 L$，还应该有一个存放阶乘计算结果的变量，我们取名为 G。最后要求打印完毕后返回输入参数 A、B、C 语句处再重复执行，当然，我们还应该有一个返回地址的指标器，取名为 RI。

上述分析中，全部参数有两类：一类是输入参数 (A，B，C)，一类是输出参数 (L$，G)。还有一类是即不输入亦不输出的中间变量(N，RI)，因此，我们把整个程序视为一个有出入口的“体”(想象中是一个方框)，就可以整理出一个数据表。

2. 数据表



有了数据表后，我们就可以用数据表中的变量来设计初始算法了。

3. 初始算法

(1) 读入任意三个实变量 A、B、C 的值；

(2) 计算 A、B 之和并存入 N；

(3) 计算阶乘；

(4) 打印输出 L$、G 并返回到(1)继续执行。

显然，步骤(3)的算法很不明确，因此应该进一步细化。

4. 细化算法

步骤(3)应细化成：

(3)－1. 若 N＞C，则计算 N 的阶乘并将结果存入 G，设定 L$＝“(A＋B)！”；

(3)－2. 若 N＜C，则计算 C 的阶乘并将结果存入 G，设定 L$＝“C！”；

(3)－3. 若 N＝C，只设定 L$＝“ERROR”。

注意到(3)－1、(3)－2、(3)－3 三步中，还有计算阶乘的算法是不明确的，同时(4)步中打印 L$、G 的算法也是不明确的，但是我们把计算阶乘与打印当作程序语言本身提供的诸如标准函数 sin 一样的语言设施来使用的话，那么，实际上没有必要再进一步细化了。假定，我们先承认有这样两个标准设施即计算任意实变量 N 阶乘值的函数子程序 IC(N)和打印输出量 L$、G 的过程子程序 PRINT(A$,X,I/)，于是即可进行程序设计。

5. 程序实现

```
10  REM :EXAMPLE 1－1
15  &. DIS:A,B,C,G,N,RI,L$
20  INPUT A,B,C
30  RI=20:L$="ERROR":N=A+B
40  &.  DO:CASE
50  &. CASE:N>C
60  &.  LET G=IC(N)
70  L$="(A+B)！="
80  &.CASE:N<C
90  &.  LET G=IC(C)
100  L$="C！="
110  &. END CASE
120  &. CALL SUB－R:PRINT (L$,G,RI/)
130  END
10  &. FUNC:IC(N)
20  IC=1
30  FOR I=1 TO N
40  IC =IC * I
50  NEXT I
60  &.  RETURN
10  &. SUBROUTINE:PRINT (A$,X,I /)
20  IF A$  =  "ERROR" THEN 50
30  PRINT A$;X
40  & RETURN I
50  PRINT A$
60  &. RETURN I
```

这个程序不复杂，程序中出现的函数子程序 IC(N)与过程子程序 PRINT(A$, X, I/)读者不必急于追究，承认它们是存在的，而且在主体程序中可以象语言提供的其它标准语句指令一样来使用。我们现在来讨论一下程序中的最前一个语句标号 10 开始至语句标号 130 为止的一段程序，这段程序被称作主程序。

行 10 是注释行，它对程序不起任何作用，是可有可无的。这里的注释行只注释上“例子 1－1”一句，可以理解为这个程序的程序名是“例子 1－1”。

行 15 是为程序中所用到的全体变量(程序中只用到简单变量)预先开辟它们的存储区，其中除 L$是字符型变量以外，其余均是实型变量。

行 20 是输入语句，要求从键盘输入三个数据，按输入的先后顺序依次赋给名字为 A、B、C 的三个变量。

行 30 给名为 RI 的变量赋入返回地址

(语句标号 20)，给名为 L$的字符型变量赋入 ERROR 几个字符，并将 A+B 的值赋给变量 N。

行 40～行 110 为一个条件控制结构。行 40 表示进入条件控制，行 50 给出条件 1 即 N>C? ，如果满足这个条件，就执行由行 60 与 70 组成的语句序列。其中，行 60 是调用函数子程序计算 N! 并赋给变量 G。行 70 是将 (A+B)! =这几个字符赋给字符型变量 L$，然后直接跳至行 120 执行，即跳出控制结构。如果行 50 的条件不满足，则进入行 80 去判断条件 2 即 N<C 是否满足，如果满足，就执行由行 90 与 100 组成的语句序列。其中，行 90 是调用函数子程序计算 C! 并赋给变量 G。行 100 是将 C! =这几个字符赋给 L$，然后直接跳至行 120 执行。行 110 标志条件结构控制的终端。

行 120 为调用一个名为 PRINT 的过程子程序，给这个过程子程序送入参数 $L、G 以及返回地址 RI，它就打印 L$和 G (如果 L$= "ERROR" 就只打印 L$)，然后返回 RI 指令的语句标号 20 的程序行去执行。

行 130 为结束行，它标志一个 S－BASIC 程序的主程序段的结束。在 S－BASIC 程序中，主程序段必须用 END 来作其结束语句。

运行这个程序，其结果列出如下：

```
? 2
? ? 2
? ? 5
C! =120
? 2
? ? 2
? ? 3
(A+B)! =24
? 1
? ? 1
? ? 2
ERROR
```

本例程序中，给出了两种形式的子程序。其一是以

```
&FUNC:IC(N)
```

定义并以

```
&RETURN
```

作结束的子程序，称为函数子程序。就本例来说，它定义函数子程序的名称为 IC。另一个子程序是以

```
& SUBROUTINE:PRINT(A$,X,I/)
```

定义并以

```
& RETURN I
```

作结束的子程序，称作过程子程序。就本例来说，它定义过程子程序的名称是 PRINT。两个子程序定义命令中的圆括号内出现的变量均是形式上的参数，并没有具体的数值，当其它的程序段调用它时，用确定数值的参数去代替形式参数，就可以得到对应的结果。一个子程序段可以多次被调用，从而使得程序编写的工作变得更为灵活。

由上述例子可以看到，结构化 BASIC 语言是一种分段结构的、非常直观的程序设计语言，它可以自上向下，或由下朝上进行分段设计。把本例的程序设计步骤归纳一下为：

(1). 分析问题，了解要做什么；

(2). 为该问题的所有输入、输出数据命名，必要的话，也应该为已确定的中间变量命名，并且列出输入/输出数据表；

(3). 由 (2) 步中的变量，初步写出算法，必要的话，可以进一步增添中间变量；

(4). 对(3)步的算法进一步细化，允许进一步增添中间变量；

(5). 根据算法写成能用 S－BASIC 语言实现的程序。

这种程序设计方法就是所谓的"逐步求精"法，也就是自上向下或由下朝上或这两

种方法的结合。这是提高程序设计效率的一个重要方面，希望读者从现在开始就对这种设计方法的概念有所了解，从而逐步形成良好习惯，以便能写出结构清晰的程序。

第二节　S－BASIC 程序的结构

一个结构化 BASIC 程序是由一个或几个程序段模块所构成。模块结构是 S－BASIC 语言的重要特点。程序段分成主程序段、过程子程序段和函数子程序段三种。

程序段就是一个以结束行 END 或由返回行 &RETURN 终止的 S－BASIC 语句序列。凡是以子程序定义语句开头以返回行终止的程序段称为子程序段；除子程序定义语句以外的其它 S－BASIC 语句开头并以结束行 END 终止的程序段称为主程序段。一个 S－BASIC 程序只能有一个主程序段且必须以 END 来结束，可以有一个或几个子程序段，也可以没有子程序段。程序总是从主程序段开始执行的。如上节例子中的程序就是由一个主程序段和两个子程序段（一个函数子程序段和一个过程子程序段）所构成。

组成一个 S－BASIC 程序的各个程序段之间是相互独立的，也就是说，程序段是一个“黑盒”，只要给它输入，它就有相应的输出。程序段中所使用的语句标号和给变量、数组命名用的标识符只有在本程序段内有意义，所以，虽然在不同程序段中使用了相同的标识符，但它所标识的量一般来说是不相同的。注意到此处论及的程序段“独立”的概念没有套用计算机界必定使用的一个标准即“是一个独立的编译单位”，这是由对解释系统的偏见导致的对程序段的判别标准，这种偏见使得对“模块”的概念混淆，在第三章到第六章中我们会更深入地讨论这个问题。

程序模块化结构的优点是：

（1）．可以使程序的编写和调试工作平行进行。例如，对于一个大型程序，可以按问题的具体内容或处理方法分成若干个相对独立或完全独立的部分，分别由几个程序人员同时分头编写与调试，最后进行联合调试。由于各个程序段是相互独立的，除了对程序段之间参数传递作一些必要的协调外，其余在各程序段中使用的资源（包括变量标识符、语句标号、设计策略、数据资源等）都可以由程序人员自己确定，互相不受牵制。

（2）．模块设计使得程序的可读性、可靠性、可维护性以及效率提高，因为模块的组合就象电子电路专家们使用集成器件一样。

（3）．对于一些常用的计算方法或专业性的数据处理方法都可以预先由有经验有水平的程序员编写成子程序并保存在子程序库中供用户选用，不仅可以避免大量的重复性工作，节省人力物力，而且使程序的可靠性与运行效率得到必要的保证。

（4）．可以支持程序的分级分节覆盖结构，从而使大型程序可在有限的硬件资源（RAM 空间）环境下运行。

那么，一个具有若干个互相独立的程序段所组成的 S－BASIC 源程序是怎样组成一个完整的可运行的内码程序的呢？对这个问题，下面作一概括性说明。首先，由词法和语法分析程序将源程序中的所有 S－BASIC“词”都用它的内部码来替换，这样就得到了 S－BASIC 的内部代码程序，然后由解释程序顺序解释执行。如果解释程序解释到一条“调用”子程序指令，那么，解释程序将寻找被调用子程序并使解释器运行指针指到该子程序的首部。如果子程序是库子程序（存放在外存储介质上），解释系统会通过操作系统（OS 系统）将库子程序从外设上向内存储区“载入”，载入内存储区的目标地址是浮动的，由解释系统根据当前内存储

区分配情况动态地决定。解释运行过程中，根据程序或程序员的当前要求进行信息的输入和输出。整个上述过程如图 1－1 所示。

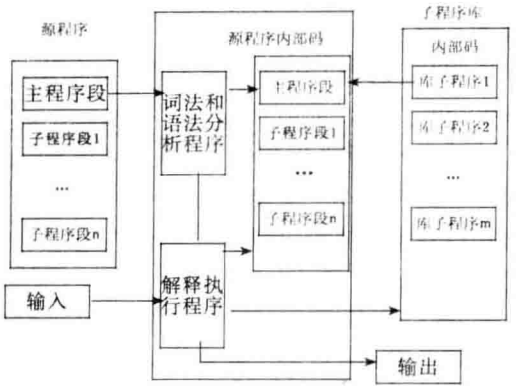


图 1－1　源程序翻译解释过程示意图

构成一个程序段的 S－BASIC 语句分为可执行语句和不可执行语句两类。

可执行语句说明程序的某种操作，计有下列三种：

(1)．赋值语句，它给变量或数组元素赋值。

(2)．控制语句，它控制程序执行的顺序。如转移、调用子程序、返回等语句都是控制语句。

(3)．输入输出语句。

不可执行语句只给解释程序提供一些信息，诸如数据的类型及存储区域的分布等。这类语句的作用是为可执行语句的执行作准备。不可执行语句有下列五种：

①．注释语句。

②．组织语句，如开辟变量区间等。

③．数据初值语句。

④．过程子程序定义语句。

⑤．函数子程序定义语句。

上述各类语句将在以后各章节中详细介绍。

第三节　S－BASIC 基本词法

S－BASIC 语句用英文单词、运算符和运算对象组成，程序由语句序列组成。

3.1　基本符号

APPLE Ⅱ 的 S－BASIC 语言中的基本符号共有以下三种：

1．**数字集**：0|1|2|3|4|5|6|7|8|9

2．**字母集**：A|B|C|D|E|F|G|H|I|J|K|L|M|N|O|P|Q|R|S|T|U|V|W|X|Y|Z

数字是用以组成数、标识符和语句标号的，字母没有单独定义，它们是用来组成标识符和字符串的。

3．**符号集，又分为**：

(1)标点符号子集：·|,|;|:|"|'|?

(2)运算符号子集：+|－|＊|/|∧|(|)

(3)关系符子集：=|>|<|>=|<=|<>

(4)专用符子集：$|%|⌣

3.2 语句标号和标识符

语句标号和标识符是 S－BASIC 中最常用的两个基本概念，虽然简单，但较重要。

1．**语句标号**

在一个程序段中，为了使 S－BASIC 能顺序控制语句的执行，给每一个语句一个编号以标识这一语句，这个编号称语句标号(亦常称为程序行号)。如第一节中的实例程序所示的那样，每一语句前的无符号整型数就是语句标号。

语句标号范围为 1－63999。在 S－BASIC 中，语句标号不仅仅是一个起标识符作用的号码，而且控制语句在一个程序段中的排列顺序，因此，它本身就包含了数值大小的概念。语句在程序段中是以其语句标号的大小来决定其在程序段中的位置的。程序员可以在语句标号的规定范围内任意选用语句标号，但对语句标号的按排有严格

的次序上的要求。

语句标号只在它所在的程序段内有意义，所以在同一个程序段中，不允许用同一个语句标号同时去标识两个不同的语句，但是，可以用同一个语句标号去标识不同程序段中的不同语句，例如

```
        ⋮
20   A=B+C      } 第K个程序段
        ⋮

        ⋮
20   INPUT  A   } 第K+1个程序段
        ⋮
```

是合法的。如果上述所示的第 K 与第 K＋1 这两个程序段被组织成一个程序段，那么这种语句标号的使用是不合法的，前一个 20 行语句将被后一个 20 行语句所替代。

2. **标识符**

S－BASIC 语言规定用标识符来给各种对象命名，这些对象包括变量、数组或者函数、函数子程序与过程子程序。所谓标识符就是以字母开头的字母、数字序列。其中，字母是指英文的二十六个字母，数字是 0，1，…，9 之一。据此，

H,S,X1,Y2,RUKT

等都是 S－BASIC 中合法的标识符。但对于不同的对象，使用标识符时有不同的要求：

(1)．给变量或数组这两种对象命名时，标识符长度不要超过两个字符，尽管标识符长度最长可达 256 个字符，但 APPLE II 的 S－BASIC 程序中只有最前面两个字符是有效的（有些 BASIC 版本能识别超过两个字符的标识符命名的变量），如变量 GOOD 和 GOAL 在 APPLE II S－BASIC 看来是同一个变量标识符。

(2)．给函数子程序命名时，标识符长度最长允许达 250 个字符且 S－BASIC 能唯一辨识这个标识符。但是，由于函数子程序名必须在这个子程序体中作为一个变量被赋值，因此，当它作为变量名在自己这个子程序体中出现时，只有最前面二个字符是有效的。而在这个子程序体外，它是作为函数名出现的，因此，它的全部字符都是有效的。

(3)．给过程子程序体命名时，标识符长度最长允许达 250 个字符，且不论在这个子程序体内还是体外，标识符的全部字符都是有效的。

在书写标识符时应当注意，在 S－BASIC 中有些词具有专门的作用和意义，通常将这些词称为关键字或系统保留字，例如 INPUT、READ、&DIS 等，这些关键字不能拿来当作一般的标识符使用（在过程子程序命名中除外，例如 PRINT 是关键字，但第一节的实例程序中仍用它来命名一个过程子程序）。

标识符本身无独立的意义，主要用来给程序段中要引用的变量、数组、函数或子程序命名，以示区别。如在第一节的程序举例中，我们用 A、B、C 等为某些数据量命名，用 IC 作为一个函数子程序的名字。用 PRINT 作为过程子程序的名字。一个标识符究竟作为哪个量或者哪个子程序的名字，这完全由程序人员决定，但建议读者选择与变量的含义与作用相适应的标识符，例如在示例程序中用 PRINT 来命名一个功能为“打印”的子程序就是一个很贴切的命名。对一般变量，由于受制于有效字符（2 个），建议在为这些变量作了命名后立即用注释语句注释它们的全称，这样做是一个好习惯，将会使你的程序更容易让你自己或其他人理解。

3.3 常量、变量、函数和表达式

3.3.1 常量

常量有三种，分别是常数、字符串和逻

辑值。

1. **常数**

常数一律用十进制表示，根据它们在内存中存储方式的不同又分为整型数和实型数两种。

实型数同数学中的实数有所不同，它只是实数集合的一个有限子集。因为计算机只能表示一定范围内的数，其确切范围则随具体机器而定，而且，计算机所表示的实型值只能认为是一个近似值，其有效数字的个数也随具体机器而定（取决于机器内部字长）。它有两种书写格式：

（1） 普通书写格式

采用普通书写格式时，实数由数字，小数点和正负号组成，例如：

+123，25.67，−10.52，0.00073

等都是合法的实数。

（2） 指数书写格式

采用这种格式时，实数被写成如下形式：

±X.XXXXXXXXE±mm

其中，“+”或“−”是正、负号，E 表示幂的底 10，X 和 m 是数字，X 组成的部分称为尾数，最多可以有 9 位有效数字，例如：

2.345678E+3，1E−1，73.96E−2

等都是合法的实数。

当输入一个指数格式表示的实数时，尾数中的小数点可以任意放置，例如输入数据为实型数 8765.43，写成：

87.6543E + 2， 8.76543E + 3， 876543E−2

等都是相同的。但计算机输出指数格式表示的实数时，一定输出规格化的指数格式，即尾数的小数点必定在第 1 位数字后面。如上述表示的三种指数形式的输入，输出时均为 8.76543E+3。

在 APPLE Ⅱ 的 S−BASIC 中，实型数取值范围为

$$2^{-128} \leq |X| \leq 2^{127}$$

整型数是一个可以为正亦可以为负的十进制整数。同实型数相似，它的取值范围也只是整数集合的一个有限子集，取值范围视具体机器而定。在 APPLE Ⅱ 的 S−BASIC 中，整型数的取值范围为−32767～+32767。整数的书写同日常习惯相同。下列整数都是合法的：

9，+28，−07，9999，−256

程序中的整型量常用来计算或表示下标。

2. **字符串**

一般亦常被称为行常量，是一组由双引号括起来的任意字符串。需要注意的是在字符串内的所有空格都是有意义的。例如：

“HOW ARE YOU”，“A，B，C =”，“DATE：18/4/89”等都是合法的字符串。

字符串的取值范围同具体机器规定的字符集有关，因为除了该机能够输入输出的字符以外，我们很难使用其它字符。但一般，要求这个字符集至少包括：

(1)按字母顺序排列的二十六个英文字母；

(2) 按数字顺序排列的十个十进位数字；

(3)空格符及一些专用符号如+、−、*…等。

此外，还要求每个字符对应一个非负整数序号，根据这个序号的大小，字符间就能进行比较，这个顺序也随计算机的不同而异，但根据对字符集的要求，应保证：

“A”<“B”<…<“Z”

“0”<“1”<…<“9”

而且字符 0 到 9 的序号是相继排列的。

除关系运算符与字符串运算符外，对字符数据没有其它运算符可用。

3. **逻辑值**

逻辑值只有两个值可供选择：1（表示

真）和 0（表示假），主要用于反映逻辑条件的真和假。在 APPLE Ⅱ S－BASIC 中，逻辑值仅由关系式确定。

对于逻辑值，允许使用三种逻辑运算符：AND、OR 和 NOT。运算结果仍为逻辑值。表 1－1 示出了这三个逻辑运算符的取值情况。

表 1－1

逻辑值 a	逻辑值 b	aANDb	aORb	NOTa
1	1	1	1	0
1	0	0	1	0
0	1	0	1	1
0	0	0	0	1

与逻辑值密切相关的还有关系运算符，即＜、＜＝、＞、＞＝、＝、＜＞等六个运算符。它们的含义就是通常所理解的相等、不等、小于、小于等于、大于和大于等于。运算结果按运算符的含义产生一个逻辑型值。例如：

2＜3　　值为 1（真）

“A”＞“B”　　值为 0（假）

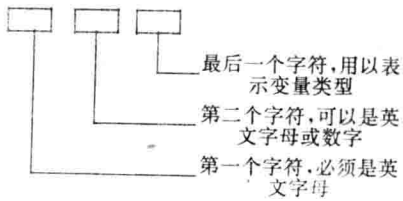
3.3.2 变量

变量就是可以取不同值的量。它的值是在程序执行过程中通过某些语句来获得或改变的。S－BASIC 中，变量分为数值变量和字符串变量两类，此两类变量同时又可分为简单变量和数组变量。除字符串变量外，数值变量还可分为实型变量和整型变量，如表 1－2 所示。

表 1－2 变量类型

变　量					
数值变量				字符串变量	
简单变量		数组变量		简单变量	数组变量
实型变量	整型变量	实型变量	整型变量		

变量用标识符来命名，为了指明某个变量是何种类型，必须在每一个变量标识符后用专用字符加以注明，若不在标识符后注明变量类型，则隐含说明此变量类型为实型。通常，在 APPLE Ⅱ S－BASIC 中，变量名采用下列表示式：



其中，最后一个字符若为 $，表示此变量是字符串变量，若为％，表示整型变量，若空缺，表示实型变量。

要注意，S－BASIC 的任何关键字不能拿来作为变量名或作为变量名的一部分。例如 GIF 就不能作为变量名，因为它含有关键字“IF”。下面给出一些合法和不合法的变量名作为示范。

合法：A，B，A1，AA，B％，B$，LM％，

不合法：1，7B，A＃，B$％，IF$，55％

BASIC 允许使用任一个未经定义的简单变量，这实际上鼓励了程序员编程中“随随便便”的作风，因为这种“默认”功能的存在，使程序员编制程序前无需对整个程序的变量分配有一个统一明确的按排，这使得程序员往往无整体观念，导致程序难读，隐含的错误（不被语法视作为错误的错误）概率极高。APPLE Ⅱ S－BASIC 尽管仍保留了 BASIC 的此一功能，但为用户提供了一个预先定义简单变量的命令，该命令语法为：

&DIS：简单变量组

简单变量组是一串用逗号分隔开的简单变量，变量有类型（字符串型、整数型和实

数型)。

该命令的语义是:为命令中的简单变量预先分配好相应类型的变量数值存储空间。此命令是不可执行的命令,仅为其后的可执行命令作准备,因此,它可位于程序的任何位置。一般应位于程序首部,使阅读程序、添加变量方便。

建议读者在编制一个程序时,一定要将输入、输出、中间这三类变量在使用前就用&DIS命令(数组变量用DIM命令)作好定义,这不仅是良好的习惯,而且在子程序的使用上是一种约束性要求。

数组变量是算法语言中很有用的概念,所以专门在本章第四节中详细讨论。

3.3.3 函数

此类函数属于内部函数,由S－BASIC配备,只需写出它的函数名和变量就可以直接引用。APPLE Ⅱ的S－BASIC配备了19种标准函数,其中算术函数11种,字符串函数8种,列出如下:

算术函数:

SIN(X)	求X的正弦值sinx
COS(X)	求X的余弦值cosx
TAN(X)	求X的正切值tgx
ATN(X)	求X的反正切值arctgx
SQR(X)	求X的平方根$\sqrt{X}$
EXP(X)	求X的以e为底的指数e^x
LOG(X)	求X的自然对数lnx
ABS(X)	求X的绝对值|x|
INT(X)	求不大于X的最大整数
SGN(X)	求X的符号函数值
RND(X)	产生一个随机数

字符串函数:

ASC(X$)	将字符串X$的第1个字符变成相应的ASCII码
LEN(X$)	求字符串X$的长度
STR$(n)	将数值n转换为字符串
VAL(X$)	将数字字符串X$转换为数值
CHR$(n)	将数字的ASCII码转换为相对应的字符
LEFT$(X$,n)	截取字符串X$左边的n个字符
RIGHT$(X$,n)	截取字符串X$右边的n个字符
MID$(X$,n1,n2)	将字符串X$从第n1个字符开始向右一共n2个字符取出

利用上述标准函数的组合,可以得到工程上常用的初等函数。给出一些常用函数的组合公式如下:

反正弦: arcsinx＝ATN(X/SQR(1－X＊X))

反余弦: arccosx＝1.5707633－ATN(X/SQR(1－X＊X))

余切: ctgx＝1/TAN(X)

双曲正弦: shx＝0.5＊(EXP(X)－EXP(－X))

双曲余弦: chx＝0.5＊(EXP(X)＋EXP(－X))

反双曲余弦: arcchx＝LOG(X＋SQR(X＊X－1))

反双曲正弦: arcshx＝LOG(X＋SQR(1＋X＊X))

以a为底的对数: $log_a x$＝LOG(X)/LOG(a)

以10为底的对数: $log_{10} x$＝LOG(X)/2.30258509

3.3.4 表达式

表达式被用来表示某个求值规则,是由运算符与运算对象以合理的形式组合而成。APPLE Ⅱ的S－BASIC中有四类运算符(算术运算符、关系运算符、逻辑运算符和字符串运算符),因而具有四类表达式即算术表达式、关系表达式、布尔表达式和字符串表