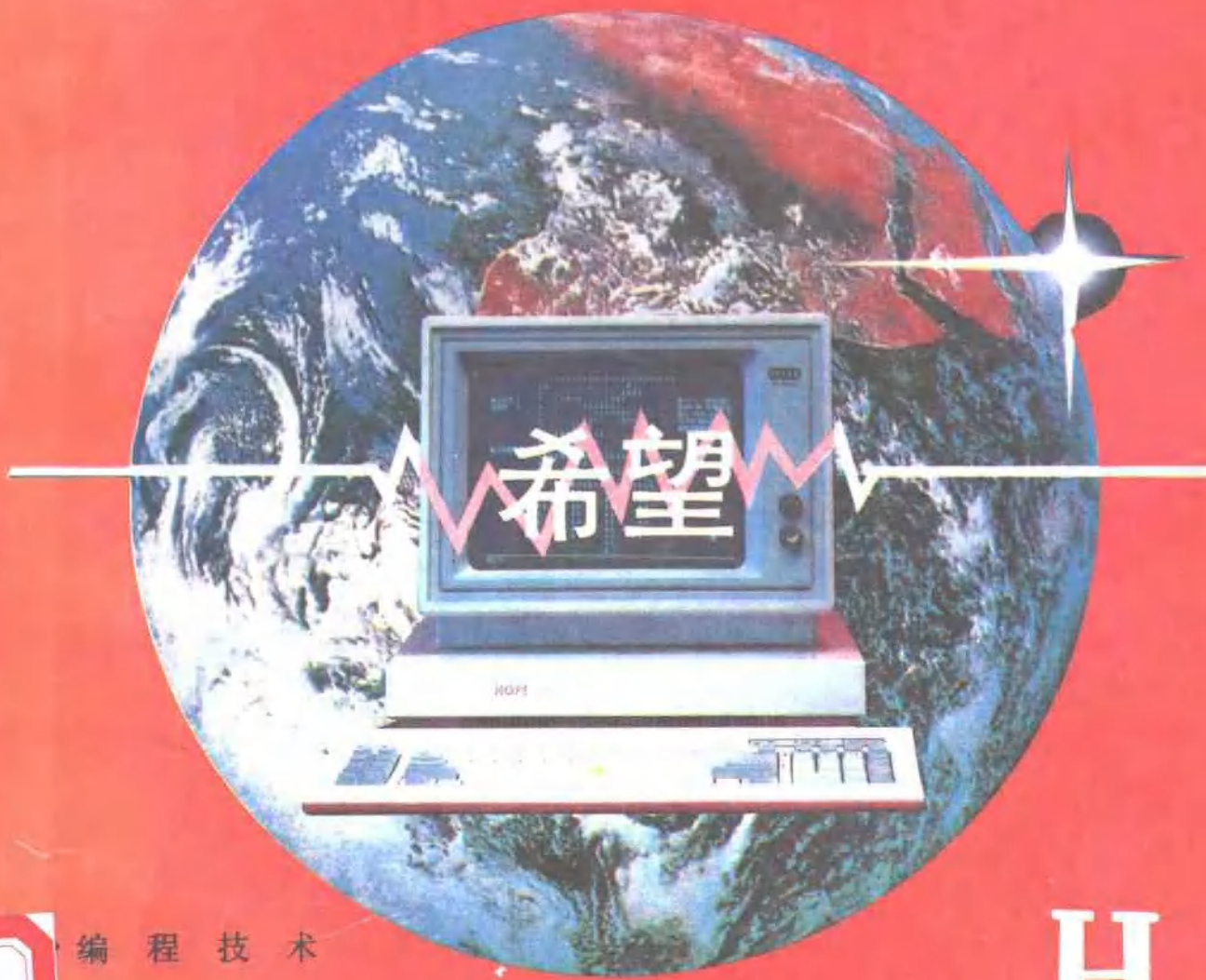
Microsoft Quick BASIC 4.0（共三册）

编 程 专 题

（中 册）



编 程 技 术
应 用 实 例

H

中国科学院希望高级电脑技术公司
一九九〇年八月

Micros ft Quick BASIC 4.0（共三册）

编 程 专 题

（中 册）

·编 程 技 术

·应 用 实 例

中国科学院希望高级电脑技术公司
一九九〇年八月

目　录

前　言

自从Microsoft公司在1975年首次介绍BASIC系统之后，用户已提出了许多关于如何使用这种通俗编程语言的问题。本书就是回答这些问题的。《BASIC编程专题》讨论了用户询问最多的问题。同时，告诉你怎样使用新的强有力的语言特点。例如，用户定义的数据类型、先进的循环结构，判断结构及过程。

本书着重讲解了在BASIC编程中的普遍问题，并且给出了编程实例。

本书的主要内容有：

·第一章，"控制流结构"

告诉你如何使用循环和判断结构来改变程序控制流程。如果你没用过BASIC，否将会对下述新控制流程结构的优越性感到满意：

——DO_LOOP循环

——If_THEN—ELSE语句块

——SELECT CASE语句

·第二章，"过程：子程序和函数"

告诉你如何写SUB和FUNCTION过程。过程使你可以建立独立的程序，并象使用BASIC自己的语句和函数一样地使用它们。

·第三章，"文件和设备I/O"

告诉你如何编程去存取磁盘文件与外围设备通讯。读完这章后，你可以知道如何用TYPE_END TYPE和DIM语句定义一个记录，并用于随机存取文件I/O。这一章还讲叙了数据文件的新的二进制存取方式。

·第四章，"字符串处理"

解释了如何用BASIC去操作字符序列。通过对本章的学习，可以对下述问题有更多的了解：

——定长度字符串

——从串的任意端清除空格以及大小写字母的相互转换的新功能。

·第五章，"图形"

告诉你如何使用BASIC语句在屏幕上画图，并改变这些图形的颜色和在屏幕上的位置，以及用颜色和图符来填充，或旋转它们。

·第六章，"错误和事件自陷"

告诉你如何编写程序陷落和处理运行中的错误和事件。

这套书的其它两本的内容如下：

手册	内容
《学用快速BASIC》	Mictosoft快速BASIC程序发展的完整指导：包括编辑、调试、使用Quick BASIC菜单和对话单元、建立

手册	内容
	快速BASIC库以及在DOS下的编译和连接。 书中的附录还介绍了如何转换用早期BASIC版本编制的程序和如何在BASIC程序中使用C语言和汇编程序。 书中的词汇表列出了所有三本书中新的和陌生的术语。
《BASIC语言参考手册》	按字母顺序排列的200多条BASIC语句和函数 书中还讨论了BASIC的元素：如数据类型、常数、表达式和运算符。 书中的附录包括有ASCⅡ代码表，BASIC保留字清单和在程序中可能遇到的全部错误信息清单。

关于BASIC的这个版本

如果你用解释BASIC，如BASIC或GW—BASIC编写过程序，你会发现快速BASIC增加了新的语句、函数和特点，使程序更结构化，更快和更容易修改。同时，它仍然保留了你所熟悉的BASIC版本中程序发展的优点。但程序使用中的优点并不是以牺牲执行速度为代价的。装入一个老的BASIC程序到快速BASIC中，编译并运行它，你会发现根据所选择的程序类型不同，执行速度比BASIA快30～40倍。

使用这个新版本，BASIC变成了一个严谨程序员的真正开发工具，以及对那些不是专业程序员但需要用快速有效的编程语言来解决问题的人们，它也是一种理想的语言。

提示：

在书中，术语BASICA指的是通常的BASICA解释版本。

学习BASIC和DOS的附加资料

这本书假设读者已具有一定的基本概念，例加变数等。同时，也假设你知道DOS的基本命令，例如COPY或DIR命令。

BASIC

如果你是个初学者，或者需要另外得到学习BASIC程序的帮助，下列书可供参考：

Dwyer, Thomas A., and Margot Crltchfield. BASIC and the Personal Computer Rdading, Mass, Addison—Weslog Publishing C0.1978

Enders, Bernd, and Bob Petersem.BASIC Primer for the IBM PC Sa XT. New York, N.Y.: New Amew Americah Library, 1984

Hergert, Douglas. Microsoft Quick BASIC.2d ecl.Reclmocel Wash.: Microsoft Press.In Press

这本书的第一次版本讨论的是Quick BASIC2.0和Quick BASIC3.0的编程技术。

DOS

如果你需要了解更多的DOS信息，可参考下列书：

Duncan Ray. Advanced MS_DOS Redmond, Wash.:Microsoft Press, 1986

Wolverton Van.Supercharging Ms—DOS.Reclmond, Wash.: Microsoft Prdss 1986

各章的结构

每一章包括一个编程主题。在每一章里，给出了由浅到深的概念。例如，第5章"图形"。从讨论图素和屏幕坐标开始，然后是画简单的形状如线和圆，最后是动图。

如果你是一个有经验的BASIC程序员，可以直接跳到每章中所感兴趣的指定节去。但如果不熟悉所讨论的主题，从头到尾的学习每个章节将是有益的。

应用举例

每章结尾部分，在"应用举例"标题下列出了一个或几个完整的可执行程序。这些程序都存在快速BASIC磁盘上。阅读这些程序后，不必亲自输入就可看到它们是怎样工作的。文件名表示在每个程序的开头。

每章一些较长的举例程序分散成小程序也存在磁盘上。这些分散的程序文件名表示在导言说明中。

小字表

每一章结尾都列出了在本章中讨论的BASIC语句和函数小节表格，这些语言和函数都按编程的任务分组，例如有关获取数据文件信息或在屏幕输出中使用颜色。

语句和函数

当介绍一个特定的语句和函数的格式时，给出一个没有可选择项或细小差别的简单格式。然而有时需要描述位置填充符。完整的格式叙述，参阅《BASIC语言参考》一书。

印刷上的约定

这本手册使用下列印刷约定：

约定举例	说明
举例：INPUT	最左边的字体用来模拟显示在屏幕上的信息。 黑体字表示响应提示而输入的信息。
Apostrophe(')	撇号标志常用来解释一段程序。它是由产生右引号的键输入的。这个键上标有`或者'或者`符号。不要使用左引号键'或者`。注意它的用法。例如' Index variable
KEYWORDS	黑体大写字母表示BASIC的关键字，寄存器名，文件名和常数名。关键字是语句必要的一部分。除非它们用下面说明的双括号括起的字。在编写程序中，必须按要求正确地输入关键字，可用大写字母或小写字母。
Commancl	小写的黑体字表示命令名。
name	
Place holders	斜体字表示必须提供的信息例如文件名。 斜体字在文件中偶尔也用来突出重点。
[optional jtems]	双括号内是可选项。
{Choice1 | choice2}	大括号和竖线表示在2个或多个选择中必须选择一个。除非所有的项都包含在双括号中。
Rdpating	一个项后跟三个点表示后面
elemonts…	还有同样格式的项。

约定举例	说明
program ⋮ Fragment	一列三个点表示其间省略了一部分程序
KEY MAMES	大写字母用来表示键名和键的序列。例如ENTER和CTAC+R，注意"+"号表示一个组合键。例如CTAL+E表示在按下E键的同时按下CTAL键本书中所指的键名符合IBM个人计算机键盘顶部的键名。如果你使用不同的计算机，这些键功能略有不同。 光标移动键（有同又称箭头键）位于主键盘右边的数字键区，又称"方向"键。单个的方向键可以指箭头的方向（如上，下，左，右）或是标在键上的名子（PGUP，PGWM）
"Defined term"	引号通常用来提示文本中定义的术语

提示，

下列格式（以"LOCK…UNLOCK"语句为例）示出了本书中的许多印刷约定。

```
LOCK{#}filenumher{,{record | {stavt}Toend}}
  ⋮
UNLOCK{#}filenumler{,{reeord | {start}To end}}
```

本书中使用的编排格式

下面是在这本书中和分配在磁盘上编写程序的指导行。推荐使使用使用这些指导行以使程序易于阅读。当然，在写程序时，不是必须使用用它们。

·关键字和符号常数用大写字母

```
'PRINT, DO, LOOP…

  ' PRINT, DO, LOOP, UNTIL are keywords:
  PRINT "Title Page"
  DO LOOP UNTIL Response$ = "N"

  ' FALSE and TRUE are symbolic constants
  ' equal to 0 and -1, respectively:
  CONST FALSE = 0, TRUE = NOT FALSE

'CONST FALSE=0 …
```

·变量名是带有一个大写字母开头的小写体，在多个部分组成的变量名中，允许中间用用大写字母表示一部分的开始

```
Num Record%=45
Date Of Brith$ = "11/24/54"
```

·用行标号来代替行号。但行标号的使用在事件陷落和错误陷落，以及带有RFSTORE的DATA语句中受到限制。

```
'Timer Handller cmd Screen two Pata are Line lakels.
 ON TIMER GOSUB Timerhandlor
```

RESTORE Screen two Data

·用单个撇号引出一个注释，

```
'This is a a comment; these two limes are ignored
'by the program when ttis running
```

·在过程和子程序中的控制流程块和语句从循环中缩进一些，

```
SUB GetInput STATIC
   FOR I% = 1 TO 10
      INPUT X
      IF X > 0 THEN
         .
         .
         .
      ELSE
         .
         .
         .
      END IF
   NEXT I%
END SUB
```

第一章　控制流程结构

这一章介绍怎样使用控制流程结构——特别是循环和判断语句——来控制程序执行的流程。循环结构按照用户指定的次数重复执行一段顺序语句。判断语句使得程序决定可选择的几个路径之一。

学完这章后，你将会掌握在编写程序时，如何用相关的循环判断语句完成下列任务：

- 使用关系运算符比例表达式
- 用逻辑运算符来联接串或数字表方式，并决定结果表达式是真或假。
- 用IF THEM ELSE和SELECT CASE语句产生程次分支。
- 编写指定次数的循环来重复执行语句。
- 编写重复执行的循环，直到指定的条件为直为止。

1.1　改变语句的顺序

一个语句块是由一行或多行上的一个或多个相关语句组成的。流程控制语句不检查表达式的左边。程序的逻辑流程从左到右，从上到下执行语句块。当然有些简单的程序，也可用单方向流程编写。但一种编程语言的大部分功能和应用来源于用判断和循环结构来改变语句执行顺序的能力。

借助判断结构，程序可求出表达式的结果，然后依据这个计算结果产生不同的语句块分支。借助循环结构，程序可重复地执行语句块。

如果用BASICA编程之后，又转入快速BASIC，你会为增加的下列各种灵活的控制流结构而满意：

- IF…THEH…ELSE语句
- SELECT CASE语句
- DO…LOOP和EXIT DO语句
- EXIT FOR语句，它是提供退出FOR NEXT循环的选择方法。

1.2　逻辑表达式

逻辑表达式只出现在判断结构和循环结构中，简单地说，逻辑表达式就是任何可以返回“真”或“假”结果的表达式。如1.3节和1.4节所述。判断和循环结构中，BASIC使用逻辑表达式。在下列IF…THEH…ELSE中包含一个逻辑表达式：

```
X<Y:
IF X<Y THEN CALL Procedure1 ELSE CALL Procedure2
```

在上个例子中，如果逻辑表达式是真（即变量X实际上大于变量Y），则执行Procedure1，否则（X小于或等于Y）则执行Procedure2

在上面的例子里也示范了逻辑表达式的一个通常用法：比较两个表达式（在本例中就是X和Y）来确定它们的关系，这些比较是用表1.1的关系运算符来完成的。

表1.1 BASIC中的关系运算符

运算符	含义
=	等于
< >	不等于
<	小于
<=	小于且等于
>	大于
>=	大于且等于

可以用关系运算符来比较字符串表达式。在这种情况下，“大于”，“小于”等等是按照字母的顺序。例如下面表达式是真，因为“deduce”＜“deduct”

逻辑表达式也经常使用“逻辑动算符”，AND，OR，NOT，IMP和EQU，这些运算符允许组成从一个或多个逻辑表达式中组成复合的测试。

expression1 AND expression2

该表达式为真只有在expression1和expressron2都正确时才成立。因此在下面的例子中，只有逻辑表达式X<=1和Y<=Z都为真情况下，才打印出“AII stored”信息。

IF (X<=Y) AND (Y<=Z) THEN PRINT “AII stored”

在上例中，逻辑表达式的括号并不是必需的。因为关系运算符<=在逻辑运算符AND之前求出。然而括号能能帮助组成一个易读的复合逻辑表达式，并保证其各个部分按安排的顺序求值。

BASIC用数值-1和0表示真与假。相反地，你可以从BASIC打印出的真表达式和假表达式中而看到这一点。

```
X = 5
Y = 10
PRINT X   Y    ' Evaluate and print a true Boolean expression.
PRINT X > Y    ' Evaluate and print a false Boolean expression.
```

□Output

```
-1
 0
```

当考虑BASIC'S NOT的NOT运算符是怎样工作的，-1使表示真就更有意义了。NOT转换操作数的二进制表达式的每一位，将1变成0，0变成1。因此，整形数0（假）按一系列的16个0位来存贮，NOT 0（真）则按16位1存贮，见下述：

FALSE=0000000000000000

TRUE=NOT FALSE=1111111111111111

按BASIC中存贮整形数所用的补码方法，16位1表示数值-1。

重要提示：

当BASIC求出一个逻辑表达式为真时，则显示-1，然而，BASIC把任何一个非零值都认为是值，如下列所示

INPUT “Enter a value: ”, X

```
IE XTHEH PRINT X "IS true"
```

□输入举例及其输出结果：

```
ENtter avalue: 2
2 is true
```

BASIC中的NUT语句是逐位运算符，而有些语言如C和Pascal中，NOT语句有两种运算，一种是逐位运算，另一种是“逻辑”NOT运算，区别如下：

- 逐位运算NOT，仅对-1值返回假（0）
- 逻辑运算NOT，对任何真（非0）值都返回假（0）

在BASIC中，任何一个不等于-1的真表达式，NOT运算返回其它一个真值，如下所示

表达式值	NOT表达式值
1	-2
2	-3
-2	1
-1	0

注意：反当表达式的求值为-1时，NOT表达式才为假。在程序中，如果要定义一个逻辑常数或逻辑变量，应使用-1表示真。

你可以用数值0和-1来定义用在循环或判断结构中的助记逻辑常数。这种方法用在本书的许多例子里。下面是一个程序的摘录，把名为Amount数组中的元素按大小进行排队：

```
' Define symbolic constants to use in program:
CONST FALSE = 0, TRUE = NOT FALSE
.
.
.
DO
   Swaps = FALSE
   FOR I = 1 TO TransacNum-1
      IF Amount(I) < Amount(I+1) THEN
         SWAP Amount(I), Amount(I+1)
         Swaps = TRUE
      END IF
   NEXT I
LOOP WHILE Swaps          ' Keep looping while Swaps is TRUE.
.
.
.
```

1.3 判断结构

依据表达式的值，判断结构使程序完成下列两个动作之一：

1. 执行判断结构体内的数条可选择的语句之一。
2. 分枝到判断结构体外程序的其它部分

在BASICA中，判断标志通过单行的IF__THEN__EISE语句处理。在它最简单的格式IF__THEN__EISE中，表达式对IF关键字后求值，如果表达式是真（非0），程序执行THEH关键字后的语句，如果表达式是假（0），程序继续执行IF__FHEN行的下一条语句。下列BASICA程序段的第50-70行就是IF__THEH的例子

```
30  INPUT A
40  ' If A is greater than 100, print a message and branch
45  ' back to line 30; otherwise, go on to line 60:
50  IF A > 100 THEN PRINT "Too big" : GOTO 30
60  ' If A is equal to 100, branch to line 300; otherwise, go
65  ' on to line 80:
70  IF A = 100 THEN GOTO 300
80  PRINT A/100 : GOTO 30
.
.
.
```

通过附加在IF__THEH后的ELSE语句，可以使程序执行一组动作。如果表达式为真，程序完成关键字THEH后的一组动作，如果表达式为假，程序完成在ELSE关键字后的另一组动作。下面的一段程序示出了在IF__THEN__ELSE语句中ELSE的作用。

```
10  INPUT "what is Tour password" ; Pass$
15 `if user enters "sworclfiish" , branch to Line 10
20 `othevwise, print a message and branch backtot Line 10:
30  IF Pass; "swordfish" THEN 50 EISE PRINT "TYY again" : GOTO10
  :
```

同时，BASICA的单行IF__llteH__EISE适合于简单的判断结构，如果想写复杂的判断语句，它会导致实际上很难读的程序代码。特别是当写一个全部选择动作都在IF__THEN__ELSE语句之内或多重的IF__THEN__ELSE时，这个问题更为突出。在下面一个BASICA程序例子中，可以看出，跟踪一个简单的测试也是相当困难的。

```
10 ' The following nested IF...THEN...ELSE statements print
15 ' different output for each of the following four cases:
20 '     1) A <= 50, B <= 50      3)  A > 50, B <= 50
25 '     2) A <= 50, B > 50       4)  A > 50, B > 50
30 '
35 INPUT A, B
40 '
45 ' Note: even though line 70 extends over several physical
50 ' lines on the screen, it is just one "logical line"
55 ' (everything typed before the <ENTER> key was pressed).
60 ' BASICA wraps long lines on the screen.
65 '
70  IF A <= 50 THEN IF B <= 50 THEN PRINT "A <= 50, B <= 50"
ELSE PRINT "A <= 50, B > 50" ELSE IF B <= 50 THEN PRINT "A >
50, B <= 50" ELSE PRINT "A > 50, B > 50"
```

为了避免上述复杂的语句，在QuiekBASIC中包括了一个IF__THEN__ELSE语句块，使得一个判断不再局限在一个逻辑块。下面的例子是用IF__THEN__ELSE语句块重写上述的BASICA程序

```
INPUT A, B
IF A <= 50 THEN
   IF B <= 50 THEN
      PRINT "A <= 50, B <= 50"
   ELSE
      PRINT "A <= 50, B > 50"
   END IF
ELSE
   IF B <= 50 THEN
      PRINT "A > 50, B <= 50"
   ELSE
      PRINT "A > 50, B > 50"
   END IF
END IF
```

关于IF…THEN…ELSE语句块详见1.3.1节。

QuickBASIC还有SELECT CASE…END SELECT语句作为判断结构（以后应用SELECT CASE来表示）。无论是IF…THEN…ELSE语句块还是SELECT CASE语句，都允许程序按逻辑判断代码，而不需把多个语句行压缩到一行。这样，不仅增加了编程时的灵活性，同时，也使程序易读易修改。

1.3.1 IF…THEN…ELSE语句块

表1.2示出了IF…THEN…ELSE语句块的格式及其应用举例。

表1.2 IF…THEN…ELSE语句块和应用举例

Table 1.2
Block IF...THEN...ELSE Syntax and Example

Syntax	Example

IF *condition1* **THEN**
  [*statemeniblock-1*]
[**ELSEIF** *condition2* **THEN**
  [*statementblock-2*]]
.
.
.
[**ELSE**
  [*statementblock-n*]]
END IF
	IF X > 0 THEN
	  PRINT "X is positive"
	  PosNum = PosNum + 1
	ELSEIF X < 0 THEN
	  PRINT "X is negative"
	  NegNum = NegNum + 1
	ELSE
	  PRINT "X is zero"
	END IF

自变量 Condition1，Condition2 等等都是表达式，它们可以是任何一个数字表达式——在这种情况下，真表任一个非0值，假是0。或者它们是一个逻辑表达式——在这种情况下，真是-1，假是0。如1.2节所述，逻辑表达式多用运算符如<或=>来比较两个数字表达式或字符串表达式。

IF，ELSE IF，和 ELSE 从句跟在语句块之后，在块内的每条语句都不能和 IF，ELSE IF从句在同一行上，否则，BASIC认为它是一个单行IF…THEH语句。

BASIC对IF…ELSE IF 从句从上到下进行求值，跳过语句块，直到找到一个真表达

式时，就执行与表达式相符合的语句，然后分枝到块外去执行ELSEIF后面的语句。

如果IF或者ELSEIF从句后没有一个表达式为真，BASIC就跳到ELSE 从句去，如果它存在，就执行它的语句。相反，如果没有ELSE语句，程序就继续执行END IF之后的语句。

ELSE和ELSE IF从句都是可选项，如下例所示：

```
' If the value in X is less than 100, do the two
' statements before END IF; otherwise, go to the INPUT
' statement following END IF:

IF X < 100 THEN
   PRINT X
   Number = Number + 1
END IF
INPUT "New value"; Response$
.
.
.
```

一个IF_THEN_ ELSE块可以包含多个ELSE IF语句，见下例：

```
IF C$ >= "A" AND C$ <= "Z" THEN
   PRINT "Capital letter."
ELSEIF C$ >= "a" AND C$ <= "z" THEN
   PRINT "Lowercase letter."
ELSEIF C$ >= "0" AND C$ <= "9" THEN
   PRINT "Number."
ELSE
   PRINT "Not alphanumeric."
END IF
```

大多数情况下，尽管有多个条件是真，但往往仅执行一个语句块，例如，假设你对下例输入了ACE，它将打印INPUT too short信息，而不会打印Can't shart with an a

```
INPUT Check$
IF LEN(Check$) > 6 THEN
   PRINT "Input too long"
ELSEIF LEN(Check$) < 6 THEN
   PRINT "Input too short"
ELSEIF LEFT$(Check$, 1) = "a" THEN
   PRINT "Can't start with an a"
END IF
```

IF_THEN_ELSE语句可以被嵌套，换句话说，你可以把一个IF_THEN_FLSE语句放在另一个IF _THEN_ELSE之内，如下例所示：

```
IF X >   THEN
   IF Y > 0 THEN
      IF Z > 0 THEN
         PRINT "All greater than zero."
      ELSE
         PRINT "Only X and Y greater than zero."
      END IF
   END IF
ELSEIF X = 0 THEN
   IF Y = 0 THEN
      IF Z = 0 THEN
         PRINT "All equal zero."
      ELSE
         PRINT "Only X and Y equal zero."
      END IF
   END IF
ELSE
   PRINT "X is less than zero."
END IF
```

1.3.2 SELECT CASE语句

SELECT CASE语句是一个多重选择的判断结构，类似于 IF__THEN__ELSE语 句块，SELECT CASE能运用的地方，语句块IF__THEN__ELSE也能运用。

两种语句的主要区别在于SELECT CASE是对单个表达式求值，并根据其结果执行不同的语句或分枝到程序的不同部分。相反，IF_ THEN__ELSE 语句块可以计算非常复杂的表达式。

举例

下面举例说明SELECT CASE和IF__THEN__ELSE语句的相似和不同之外。

下例说明了用IF__THEN__ELSE块来实现多重选择。

```
INPUT X
IF X = 1 THEN
   PRINT "one"
ELSEIF X = 2 THEN
   PRINT "two"
ELSEIF X = 3 THEN
   PRINT "three"
ELSE
   PRINT "must be integer from 1-3"
END IF
```

上述例子用SELECT CASE重写如下：

```
INPUT X
SELECT CASE x
   CASE 1
      PRINT "one"
   CASE 2
      PRINT "two"
   CASE 3
      PRINT "three"
   CASE ELSE
      PRINT "must be integer from 1-3"
END SELECT
```