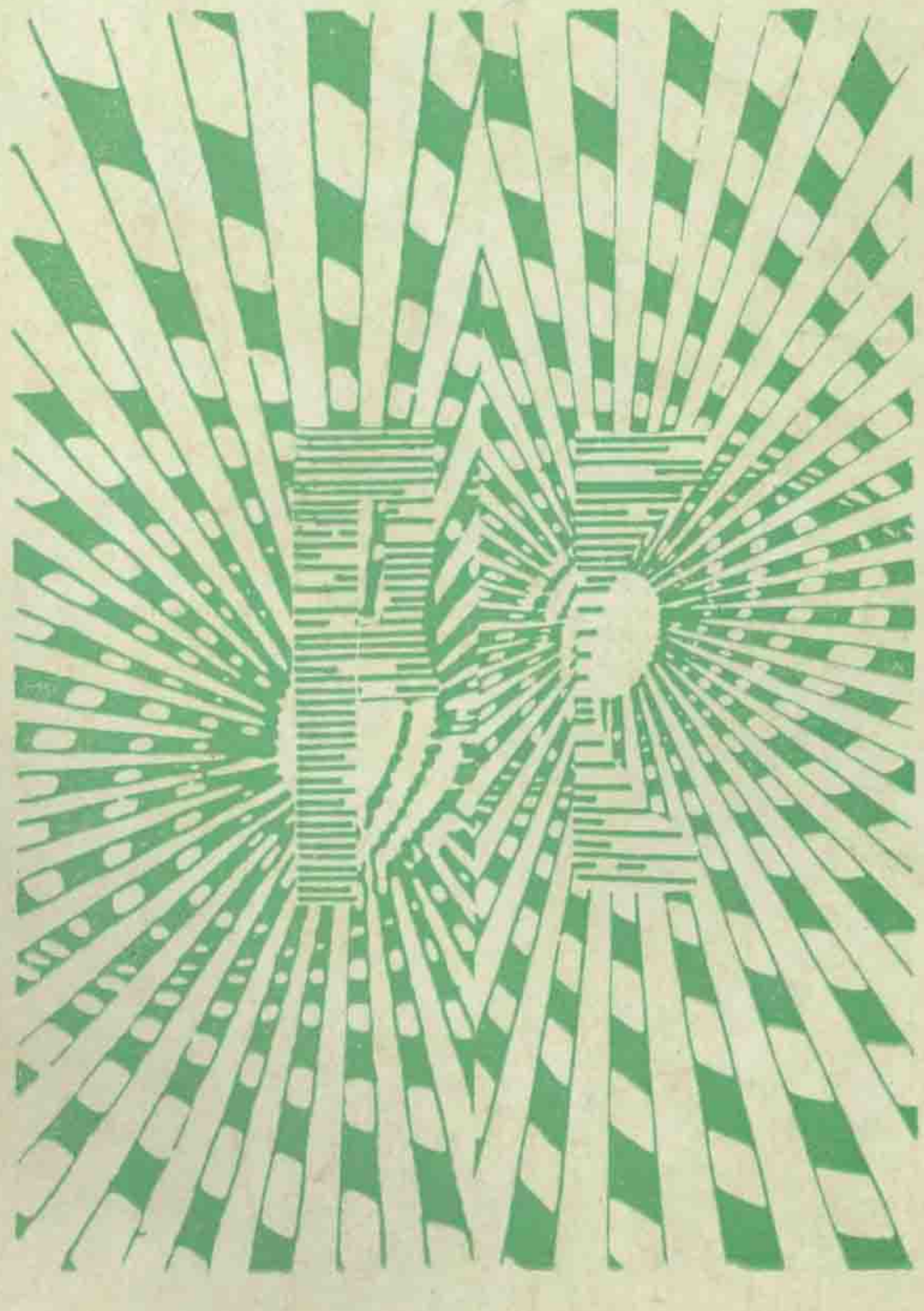
李斌　张成明　吴宗粹　编译

微机BASIC

程序设计实用教程



陕西电子编辑部

微机BASIC程序设计实用教程

李斌　张成明　吴宗粹

编译

陕西电子编辑部

内 容 提 要

本书主要根据美国 J. P. Lamoitier 的 “BASIC EXERCISES FOR THE IBM PERSONAL COMPUTER” 一书编译而成的。书中通过对大量实用例题的陈述、分析、框图设计、程序编制及运行结果的示范，系统讲述了 BASIC 程序设计的方法和技巧。所举实例及练习从简到繁，由浅入深，循序渐进，使读者易理解、好掌握，在学习过程中逐步积累知识和经验，锻炼和提高解决实际问题的能力。

本书的实例中包括了几何学，整数及数据处理、代数方程和微积分的求解、财务及税收、运筹学、统计及智力游戏等各方面的应用，具有较大的实用价值。书中的程序大都是通用的标准程序，不仅可直接在长城0520机、IBM PC、IBM PC/XT及其各种兼容机上运行，也可在Apple机、PET/CBM机、TRS-80机，以及各种小型机上运行。

本书取材精练、内容丰富、结构严谨、语言通俗，而篇幅又不大，适合于具有初中以上文化程度的广大读者自学；也适合于各级各类学校（包括中、小学），各种学习班、培训班作为教材或教学参考书。

前　　言

BASIC语言是各种微型、小型电子计算机最基本、使用最广泛的一种 程序设 计语言，是广大用户必须掌握的十分重要的工具，也是绝大多数初学者最易学习和掌握的一种交互式会话语言、学习计算机应用的入门语言。

如何来学习计算机程序设计呢？现在出版的各种教科书、参考书大都是从基本语句入手来一步步的讲述和举例。能否采用更有效的方法，即主要通过一系列结合实际应用的设计训练来学习，使能达到事半功倍的效果。本书即为此而编写，它主要根据 美国J.P.Lamoitier的“BASIC EXERCISES FOR THE IBM PERSONAL COMPUTER”一书，再结合我国的实际进行增补编译而成的。本书通过对一套在各学科领域具有实用价值的典型的问题的陈述、分析、框图设计、程序编制及运行结果的示范，系统讲述了 BASIC 程序设计的方法和技巧。书中的实例及练习均是从简单到复杂，由浅入深、循序渐进，特别适合于具有很少的科技基础及计算机知识的读者进行自学或短期培训用。读者在学习过程中将可一步步的积累知识和实际经验，锻炼和提高解决问题的能力，学会用 BASIC 程序去解决自己工作和生活中的实际问题。

本书中的全部程序均采用在国内外应用最广泛的IBM PC机BASIC语言来编写的，这些实用程序不仅可用于IBM PC及其各种兼容机上，同时只需略加修改即可在AppleⅡ、TRS-80以及其它各种有 BASIC 语言的微、小型机上运行。至于本书中所讲述的程序设计方法和技巧，则对进行其它各种语言程序设计也是非常有用的。

本书的每个例题均采用下述科学的方法来讲解：首先是关于问题的提出，问题的分析；其次是程序框图的绘制和注释；第三，编写相应的程序并给出运行情况的样本；最后在各章的末尾给出了一些有实用意义的练习题给读者去作；在全书的最后还给出了所有练习题的参考答案，供读者检验自己学习理解的情况和设计程序的能力。

本书的各章主要内容简介如下：

第一章——课程介绍：主要通过一个计算所得税收入的简单实例来阐明怎样设计BASIC程序。

第二章——程序框图：在开始编写BASIC程序之前，怎样设计组成一个好的程序框图。在本书的其余各章中进一步表明，设计一个良好的框图对程序设计的极大重要性。

第三章——整数：整数在广大的领域中——从古典数学（埃及分数）到现代计算科学（整数制的转换）——的应用都是十分重要的。本章讲述了整数应用的多种程序的设计。

第四章——几何学：讲述了对某些相当复杂的解析几何的公式如何设计 简单 的BASIC语言程序来求解，以及将这些程序计算应用于非常具体的问题（如象围栏的修造等）。同时还讲述了怎样把这些简单而有用的程序结合在一起，以便能利用终端绘制曲线图。

第五章——数据处理：有关于分类、合并文件和较复杂的按指定年代报告营业时间的调

整等例题，它也包括了告知任一日期是星期几之类的日常事务处理的简单程序。

第六章——数值计算：关于代数学和微积分的一般公式的使用。有求多项式、积分值及解方程的几个例题。同时讲述了微、小型计算机程序设计中的重要问题——数值结果和精确程度——的探讨。

第七章——财务：包含销售和增长率预测，贷款偿还方案的比较等有趣而实用的计算。还有关于更先进的所得税收入的应用等。

第八章——游戏：在前面各章知识的基础上，搞一点智能程序的设计。用计算机玩游戏可训练使用者的键盘操 作水平和 反应能力。关于BASIC随机函数的使用也结 合在投骰子的程序中来说明。

第九章——运筹学：着 重强 调BASIC中数组和下标变量的使用和实 际应用，如任务安排、工程管理(PERT)以及最理想的旅程计划等。

第十章——统计学：包括统计学中常用的均值、方差和标准差的计算，以及更有用的结果：不对称和峰态。还有线性回归和测定BASIC的随机数产生函数RND的分布情况的 程序实例。

第十一章——杂例：通过十分有趣的“黄道带的宫座”和国际象棋中“八个皇后问题”来讲述信息处理的程序设计及技巧，并说明了分类处理对BASIC程序编制的限制。

本书取材精练、内容丰富、结构严谨、语言通俗流畅，而全书篇幅又不大，特别适合有初中以上文化的广大读者自学；也适宜于各级学校（包括中、小学）以及各种学习班、培训班作为辅助教材或教学参考书。

本书是由李斌工程师（编译初稿，整理大部分习题及参考答案，并进行了全书插图贴字和誊写工作）；张成明高级工程师（审校初稿，并提供了部分习题）；吴宗粹副教授（撰写了提要和前言，还负责了全书的汇编审校）合作编译而成。有关的老师对初稿提出了宝贵的意见和建议，我们对他们表示衷心的感谢。

本书 编译 者与J.P.Lamoitier先生一样，希望本书将促进所有的读者能通过实际使用的训练而 学会BASIC语 言。并热诚的欢迎建设性的批评和建议，我们将致以深切的谢意。

编 译 者

1987年6月初稿

1987年12月完稿

于西安

目　　录

第一章　课程介绍

引　言

每个人都能够通过一些实际训练而学会一种计算机BASIC程序设计的方法。本章将说明程序设计不只是专业人员的事情，而是每一个有志者均可掌握和应用的。首先通过一个简单的实例，你将会了解BASIC语言的基本结构和规则，并阐明改进程序的方法。本章还指出了要理解这些细则并非需要较全面的BASIC知识，而只需具有最初步的计算机知识和英语基础即可以。

虽然能够通过阅读教科书来增进你的BASIC运用能力，但通过编制实际程序来学习BASIC则更为有趣和快速，这种方法可提供无法估价的程序设计的经验。如果你从头到尾学习和做完本书各章的例题和习题，将会使你获得正确可靠的BASIC语言的知识和设计程序的能力。

1.1　计算所得税收入

作为我们的第一个例题，是用下面的公式来计算所得税收入。通常是用数字表示所得税收入：

所得税收入 = 总收入 - N*1000

其中，N表示雇员的人数。

这个实例可用下述几行BASIC语句来描述：

```
20  INPUT G,N      (输入总收入和N)
30  T=G-N*1000     (计划总收入-N*1000)
40  PRINT T        (打印结果)
50  END
```

这个程序虽然简单，但却表示了关于BASIC程序的基本结构：

- 每一行具有一个行号。计算机运行时按行号数从小到大的顺序执行。
- 每一行包含一个指令，使计算机完成一定的运行动作。
- INPUT指令是一个从键盘读入数据或字符的指令，是给计算机送入信息的。
- 30行这个指令中的乘法运算在程序中用星号(*)表示。
- 程序以END命令结束，这是可选择的。

如果上述程序在一台计算机上运行，程序和用户之间将产生下面的对话：

```
? 2160, 5          (用户键入)
16160              (计算机打出)
```

当计算机执行到一个INPUT指令时，计算机就会打印出一个问号，提示用户它正在等

待输入数据。

在前面的对话中，用户键入21160和5，在这个程序中，由于紧跟在INPUT后的变量名是G和N，键入的第一个值21160，被计算机接受并赋值给G，第二个值5，被赋给N。使用这些值，计算机则完成程序30行的计算，并打印出结果16160。

这个结果是正确无疑的，但这种对话方式是模糊不清的。让我们对此程序做些改进，显示一种更好的图示，继续前进，打印出一些注释性的文字。打印的这些文字，放置在一个PRINT指令之后的双引号之内，改进的程序如下：

```
10 PRINT "Gross income, Number of dependents";
20 INPUT G,N
30 T=G-N*1000
40 PRINT "The taxable income is";T
50 END
```

这些分号将制止打字机字头自动换行

现在，用户和计算机之间的对话就更容易理解了：

```
Gross in come, Number of dependents? 21160, 5
The taxable income is 16160
```

（当程序等待数据时，通常显示出“？”）

在许多BASIC版本里，包括本书，前两个命令10和20，能够写成一个命令：

```
20 INPUT "Gross incme, Number of dependents"; G, N
```

当IBM个人计算机BASIC执行INPUT命令时，如果是一个逗号，通常问号将不会被打印出；若换成一个分号，问号则紧跟在这一行。这就提供了控制问号的安插（或其它敏捷的符号），并且能导致更感兴趣的对话。

1.2 计算所得税收的另一种方法

根据1981年一个真实的美国国内税收服务(IRS)表格1040，我们能够碰到一个所得税收入T的更为详细的计算：

$$T = G - D - N \times 1000$$

其中：G 是调整后的总收入；

D 是总扣除；

N 是雇员人数（以前的）。

我们合并这些新信息，重新定义程序如右：

```
10 INPUT "Total income"; I
20 INPUT "Total adjustments"; A
30 G=I-A
40 INPUT "Total deductions"; D
50 INPUT "Number of dependents"; N
60 T=G-D-N*1000
70 PRINT "The taxable income is";T
80 END
```

计算机和用户之间出现了下面的对话：

```
Total income? 27624
Total adjustments? 1737
Total deductions? 4727
Number of dependents? 5
The taxable income is 16160
```

在这个例子中，程序里每一个变量都赋予一个名字。在计算机科学术语中，这个名字被称为“标识符”。让我们回过头来，列出这个程序中所有的标识符：

I	总收入	G	调整后总收入
A	总调整	N	雇员人数
D	总扣除	T	所得税收入

使用单一的字母命名标识符是标准BASIC的限制(通用于"家用计算机"的BASIC)。这些标识符可以仅是一个字母或者是一个字母和一个数字。然而，IBM个人计算机BASIC标识符可以扩展到任意长度，但只有前40个符号是有效的。用这种BASIC，可以通过赋给更形象化的名字来改进程序的易读性：

I→ TOTINCOM	G→ GROSSING
A→ TOTADJUS	N→ NOFDEPEN
D→ TOTDEDUC	T→ TAXINCOM

1.3 结语

这个基本的例子表明了如何用BASIC设计一个简单的程序。要着手编写更有特色的程序，必须首先学会分析一个程序和设计一个"流程图"的技术。这两种技巧将会在以后各章得到发展。

本章提出的计算所得税收入的例子，在第七章计算实际的所得税收益时将继续扩展。

练习题

1.龙形储钱盒内贮存有钱币，其中N1个1元的，N2个5角的，N3个5分的，N4个2分的，N5个1分的。以元为单位统计该盒内共有多少钱?

2.某人今年存入银行5000元人民币，利率是9‰，问八年后本利共是多少?

3.以年利率i投资P元钱，把利息再投资，N年后总金额为：

$$A = P(1 + i)^N,$$

若以季投资，则N年后总金额为

$$A = P(1 + i/4)^{4N}$$

现某人按6‰投资5000元人民币，分别按年和季计算复利，问10年后总金额各为多少?

第二章　程序框图

引　言

本书第一章中，我们已经学习了BASIC语言的基本原理，了解了如何编写一个简单的程序。在后几章里，例题将变得较为复杂，我们学习编写程序的方法（即直接地写出程序）将不再是可行的。因为提出的问题更复杂，必须首先分析问题，然后在程序列表编码之前画一“流程图”。的确，经验表明，绘制流程图对程序设计是一个无法估价的辅助物，特别是对初学者尤其如此。

本章的目的是说明构思一个流程图的专门技术。后几章将提供许多应用这些资料的机会，实践绘制流程图的技术。

以后，有了经验，有可能减少花费在设计程序框图的时间，但对初学者来说，这种实践是必不可少的。

2.1　框图的意义

框图是解决一个问题的过程的图解表示。在目前的技术状态下，框图对程序设计者来说是很有用的，但对计算机来说，它是不能理解的。然而，框图提供了一种在分析问题时，一些关键性的部分不会被忽略的查证方法。框图也可以促进完成同一程序设计项目的各种人之间的相互交流。总的说来，对一个初学者，一个详细的框图的制定，是获得完善的程序设计的第一步。

2.1.1　框图的不同类型

在实际使用中，框图有三种类型：

(1)系统框图：主要运用在数据处理应用方面。这种框图表明文件和程序之间的联系。

(2)概念框图：常用于包含一些相互影响的复合算法的大型程序的宏观描述。

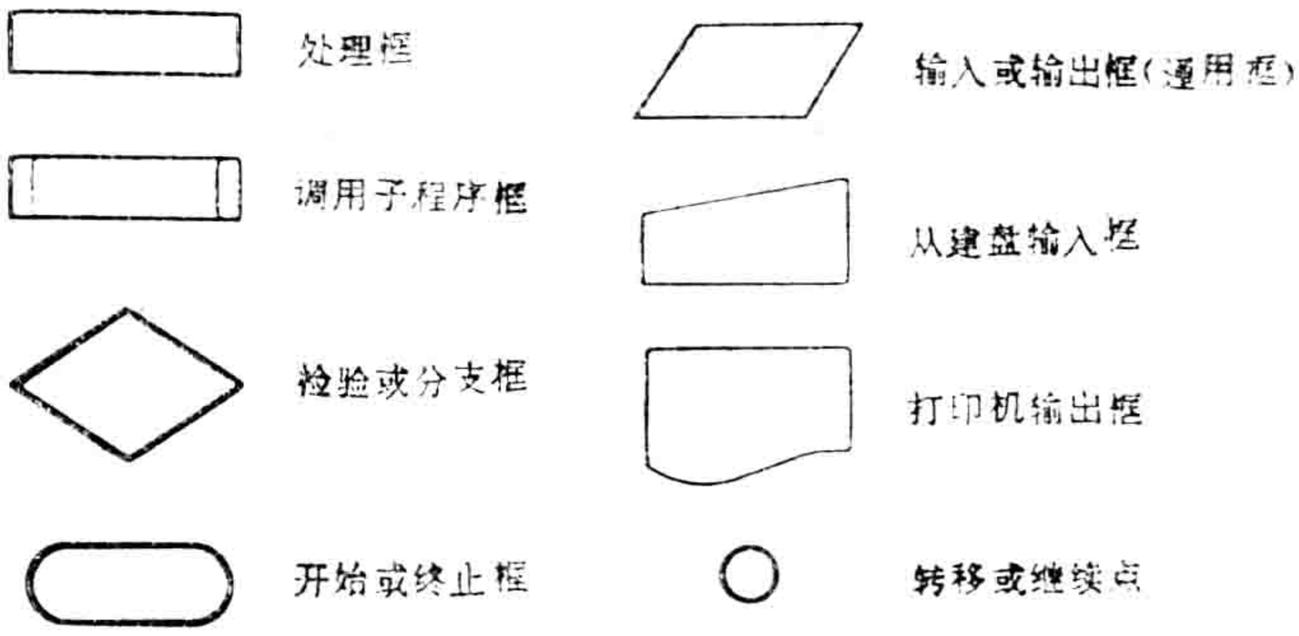
(3)详细框图：制定一个完全准确的计划过程的表示。这类框图去掉了所有潜在的意义不明确的内容，使得程序设计更为容易。

但须注意，框图应尽可能地不受程序设计语言的制约。

2.1.2　标　准

绘制框图的标准和符号已由美国国家标准局(ANSI)所颁发。绘制所有标准框图的符号图样已由IBM和其它几家公司提出，并广泛地得到应用。使用在框图设计中主要的符号如下所示。

框图的组成部分：



我们应注意，这儿有许多方法能够用来描述算法、程序和系统。列举几个如下：变形语言，伪码，结构图，数据流程图，Warnier流程图，“输入—处理—输出”(IPO)，层次IPO(HIPO)等等。这些方法中许多具有很大的优点，更深入的学习，将会理解到它们是一个复杂的过程，必须借助于程序设计才能对其有较为深刻的了解。对初学程序设计者来讲，绘制框图的方法具有易于接受和广为理解的优点。

我们从一个简单的“小框图”来开始绘制框图的讲解。在程序设计中，我们允许答案不是唯一的。我们将继续研究更为复杂的情况。

2.2 求两个数A和B的最大值

我们假定X是A和B两数中较大的值。要获得这一答案，如何使得必写的指令数减低到最小程度？

第一种解法：我们比较A和B，如果A≥B，我们贮存A值到X中，否则把B存于X中。这种方法能够用一个框图形式来表示（见图2.1），在这个框图中，设置了一个比较“A≥B”的菱形框，两个矩形框相当于分配（说明）。相当的BASIC指令序列见图2.2

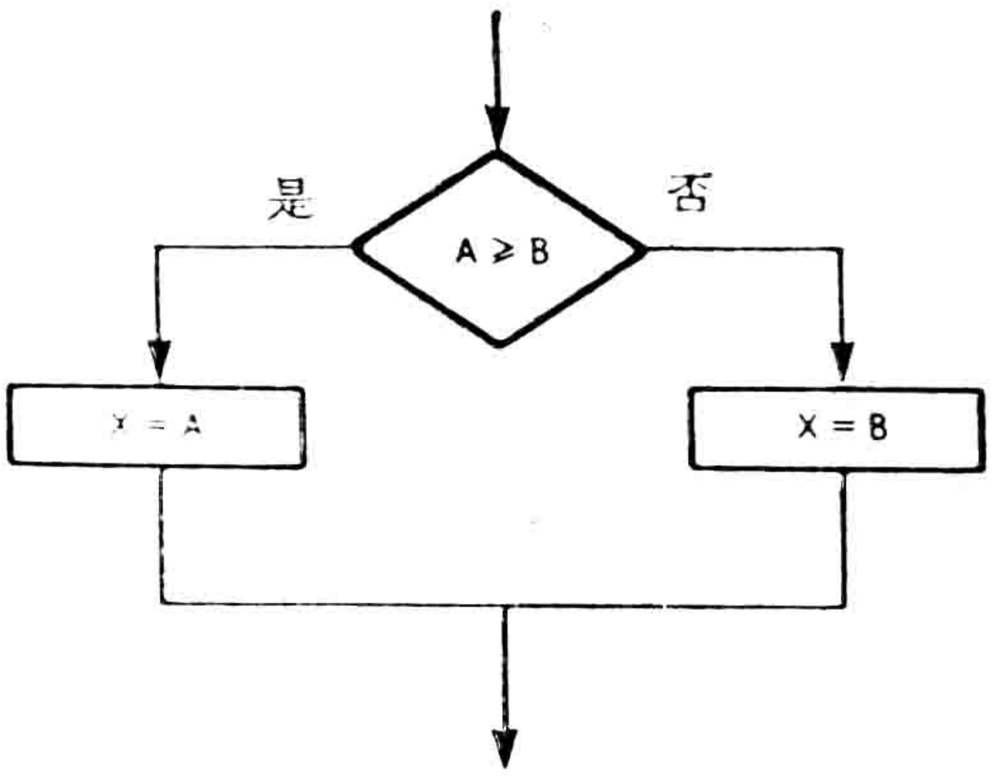


图2.1 求两数中较大者的框图

```
100 IF A>=B THEN 130
110 X=B
120 GOTO 140
130 X=A
140 '(continuation)(后继语句)
```

图2.2根 据2.1写出的程序

如果我们利用更先进的BASIC，可以写为：

```
100  IF A>=B THEN X=A ELSE X=B
```

第二种解法：避免120的分支，我们改变图2.1，移去一个说明指令。给出框图见2.3，相应的BASIC程序见图2.4。

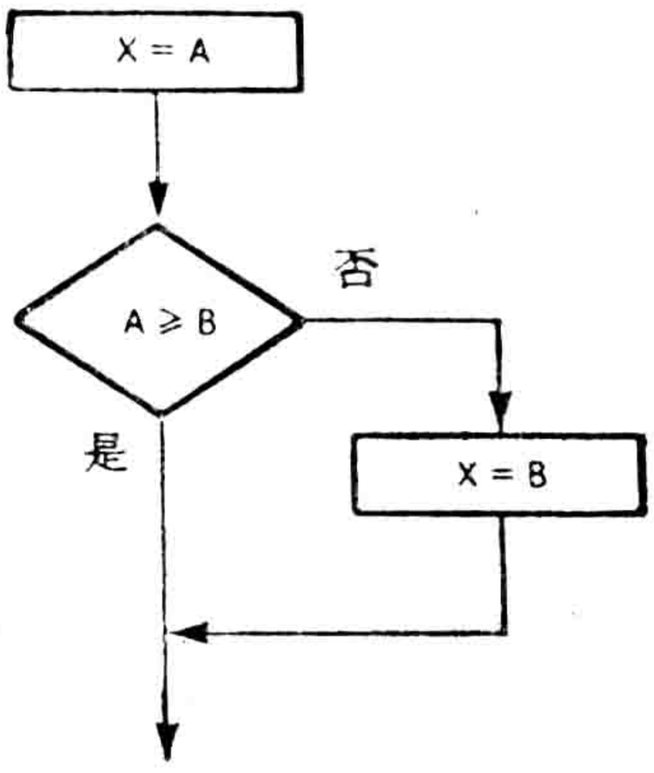


图2.3　更有效的框图

```
100 X=A
110 IF A>=B THEN 130
120 X=B
130 '(continuation)
```

图2.4　无GOTO的程序

在这个解法中，我们没有使用GOTO指令，因而使得程序得以精简。用更好的BASIC来写，我们有：

```
100  X=A
101  IF B>A THEN X=B
```

第三种解法：对于包含有最大值(MAX)和最小值(MIN)函数的BASIC解释程序来说，我们只需这样写：

```
100  X=MAX(A,B)
```

上面介绍的最大值和最小值函数，仅对极少数家用计算机是通用的。

注意：当这两种函数有效时，它们常常接受一个任意参数。例如，我们可以写：

```
Y=MAX(X,3,Z,C)
```

甚至

```
Y=MIN(X+Z,V*W,K*SIN(A))
```

2.3　一个完整的框图例子：求一个数组的最大元素

假定，我们想要找出一个具有100个元素的数组A中最大的数。

对这种方法我们提出下面的分析：

- 置X=A(1)
- 给I取值2,3,4,……,100,
- 比较X与A(I)
- 如果X<A(I)，传送A(I)的值到X，否则继续将X与下一个A(I)值进行比较。

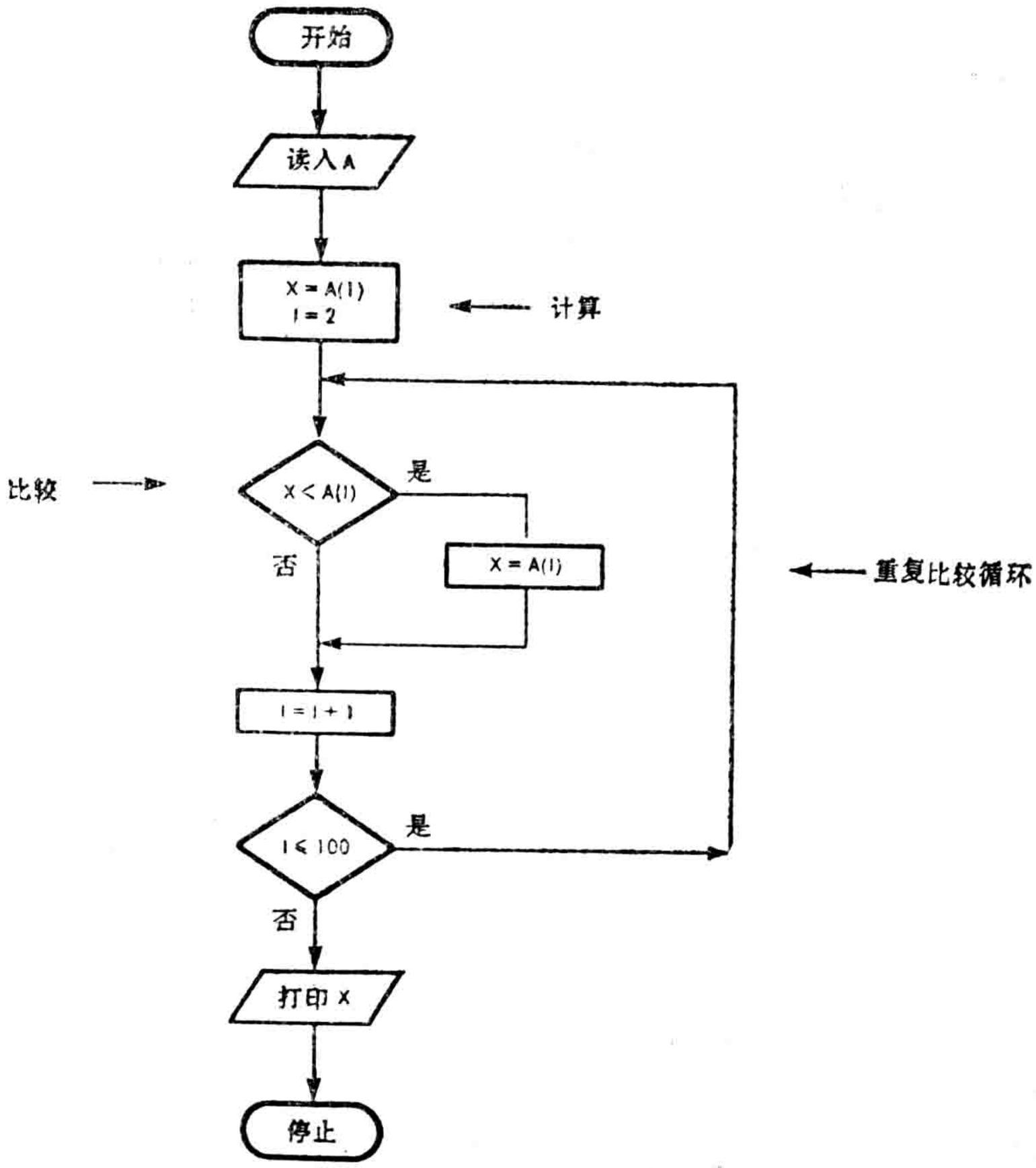


图2.5　找数组中最大元素的框图

当完成后，X中将获得最大值。这种方法用框图表示，见图2.5。

图2.5中的图解法阐明了如下常规：

* 输入或输出指令被一个平行四边形框围住；
* 计算指令由一个矩形框围住；
* 比较指令用一个菱形框围住。

我们也须注意一种表达方式

I = I + 1

表达成最一般的形式，一个计算指令可以写成

变量 = 〈表达式〉

这个指令意味着表达式的数值要被计算并分配贮存在等号左端的变量中。基于这个原因，这种形式的指令被称为“指令陈述”或“赋值”。字符“=”在这儿作为赋值符。但是，在一个菱形框内，指令

I = 100

意味着“比较I到100并看它们是否具有相等的值”。这个指令决不意味着100这个值被贮存

于I中。换言之，一个菱形框中符号“=”的作用就象其它比较符（如<，>等）一样。

2.4 如何检验框图

如果一个程序是由不正确的框图得到的，它肯定不会产生正确的结果。我们在进入程序设计阶段之前应尽可能使框图正确无误。

为达此目的，我们可以对这个框图进行“人工检查”，即模拟在计算机上的操作来一步一步地追踪框图路径，以保证这些命令是正确的，并检查核实（用手工）这些复杂的计算。

让我们回到前面图2.5所示的框图，并设想一个较小的五个数值的数组。

在开始时，我们置X = A(1)，这样X就取得值3（见图2.6）。

I	1	2	3	4	5
A(I)	3	2	4	-1	6

图2.6 五个元素的数组

现在我们沿这个框图走一次。图2.7示出了X随I而改变的内容。

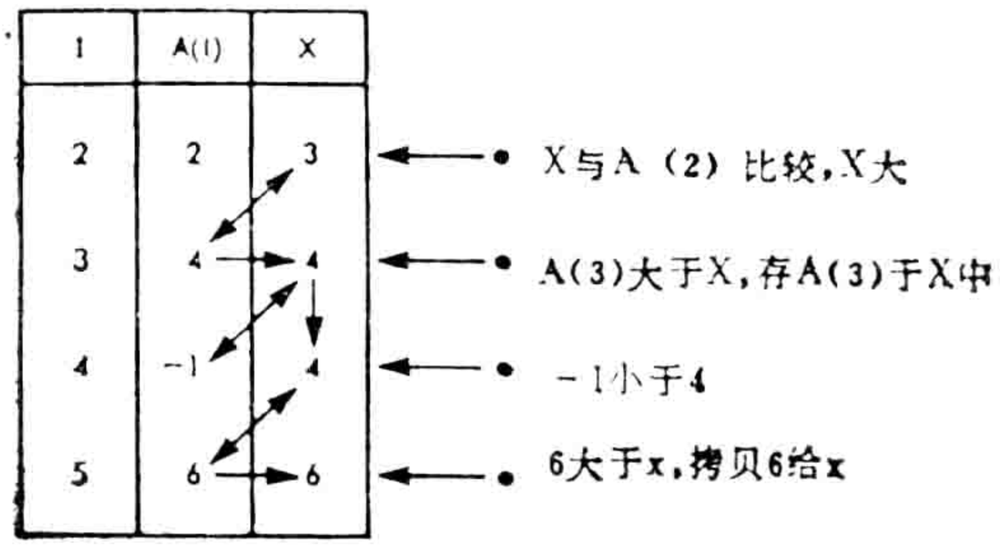


图2.7 元素的比较

我们注意到，使用这种方法，X中的确被变换进去了数组的最大元素。因此，我们可以进行下去，并可以按这个框图编写程序。

注意，这种方法仅能用于相对简单的框图。

图2.5所示的框图可以用多种方法转换成BASIC程序。图2.8所示的就是一例。

这个程序并非是最完善的描述，但它易于理解：

- 110行到130行读入整个数组。
- 140行到180行寻找数组的最大元素。
- 200行，210行等保存读进数组的100个元素的实际值。

```
100 DIM A(100)
110 FOR I=1 TO 100
120   READ A(I)
130   NEXT I
140 X=A(1)
150 FOR I=2 TO 100
160   IF X>=A(I) THEN 180
170   X=A(I)
180   NEXT I
190 PRINT "The largest element in tne array =";X
200 DATA ...
210 DATA ...
410 END
```

图2.8 最大元素程序

评论这个程序：该数组包含恰好100个元素这个程序才能运行。最初读一自然数N，它是这个数组中元素的实际数才是合意的。我们能够提供一个程序使它能适应并处理任一体积N直到100的数组。图2.9给出了基于这一观点的较好的程序。

```
100 DIM A(100)
105 READ N
110 FOR I=1 TO N
120   READ A(I)
130   NEXT I
140 X=A(1)
150 FOR I=2 TO N
160   IF X>=A(I) THEN 180
170   X=A(I)
180   NEXT I
190 PRINT "The largest element in the array =";X
200 DATA 5
210 DATA 3,-2,34,5,0
410 END

The largest element in the array = 34
```

图2.9　修饰最大元素的程序

注：仔细查看这个程序，我们可以看到：

* 105指令读数组元素数N。
* 200行置数5相当于这儿是五个元素。
* 210行置5个元素的值。
* 这个程序由指令DIM A(100)限制于100个元素。修改这条指令，该程序能够具有处理较大或较小量的能力。
* 这种结构形式使得程序修改方便，易于阅读。

注意，利用FOR循环是一个通用的规则，终值与其是固定不变的，还不如是可变的为更好。

2.5　判断点

在一个框图中，一个判断点具有一个入口和两个或三个出口。图2.10举例说明了这一点。符号“？”用作一个比较符。

有一种特殊情况，一个框图中某个判断点可能多于三个出口，由于框图必须表示一般的算法，因此可能出现这种情况。标准框图设计过程并不用多于三个出口来详细表示一个判断点，但图2.11说明多出口也能表示。

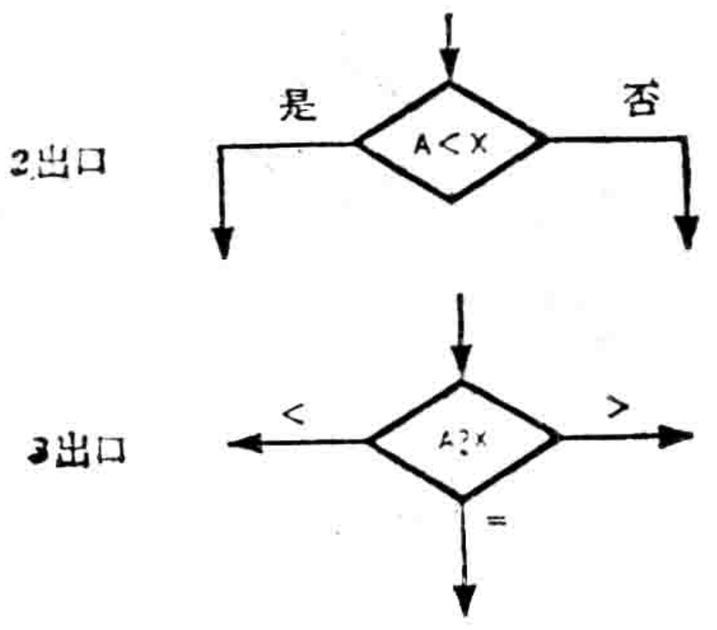


图2.10　两个和三个出口的判断点

2.6　分支的跳跃转移技术

我们如何画出一个框图，使它左边的框执行奇数序列，而右边的框执行偶数序列？这种