微型机与小型机

(BASIC语言)

程序调试技术

白英彩　刘寿和　编译

苏州电子计算机厂

一九八二年七月

前　言

BASIC系为Begineer's All-purpose Symbolic Instruction Code（初学者通用符号指令代码）的缩写。它是一种易学、易写而且具有“会话”性能的语言。也是国际上通用的比较简单的算法语言，适用于进行各种数值计算。对于具有中等文化水平的人员来说，只要经过短期学习和训练，就能掌握和使用这种语言来编写程序。因此，推广普及这种语言对于计算机的普遍应用是十分有意义的。目前，国内已拥有数千台计算机，其中包括有上千台小型计算机（如DJS—130，DJS—110等产品）和数量更多的微型计算机系统（如DJS—054等产品）。在这些小型和微型计算机中都已普遍配备了BASIC语言。

BASIC语言是一种解释性语言，源程序可用“会话”方式输入机器和执行，在打字机的键盘上允许修改程序和执行适当的语句。因此，用BASIC语言来编写程序，在检查和调试程序时是十分方便的。对于应用小型机和微型机的人员，特别是对于那些初次与计算机打交道的人员来说，应用BASIC语言来编写程序並进行软件调试是很适合的。

初次与某些计算机（如DJS—130，DJS—054等）打交道的人都深有体会：根据给定的任务，编制一个程序，可能花不了多少时间，但是欲把该程序在机器上调试好並加以执行，常常要花极大的力气，以致不知道从什么地方下手。我们正是根据这种体会，编译了这个材料，供广大同行作为参考。该书通过大量实际例子，引出问题，並一步一步地加以解决，引人入胜，使读者犹如身临其境。所提出的每一个问题，都经过几次迂回终得解决，使你不感到枯燥、乏味而是兴趣盎然。就是通过这些例题，对于各种程序调试技术（诸如流程图、注释语句、模仿计算机工作、打印语句、强制技术、分块调试技术、抽点打印调试技术、预埋调试法和插入码技术以及各种调试技术的综合运用，等等）进行了由浅入深的叙述。对于初学者容易忽略和出错的地方，均通过实例加以反复说明。掌握了这些调试程序的技巧，你将

再也不感到调试一个程序是件棘手的事情了。

应该说明的是，在本书中是采用BASIC来编写程序，且在SOL型微型计算机系统上进行调试示范的。但是，由于BASIC语言是一种通用的高级语言。所以，本书引申出来的各种调试程序的技巧，对于国内的各种微型机系统（如DJS—054等）和小型机（如DJS—130等）均有借鉴意义，有些调试技巧，可以直接加以引用。希望读者通过阅读本书有些得益，大力促进微型机和小型机的推广，书中难免有欠妥之处，也希望读者予以批评指正。

华南工学院计算机系付主任陈兴叶同志悉心审定过该书译稿，並提出些有益的意见。在此表示感谢。

本室下列资料扩大发行

DJS—100 系列机成套软件

包括《服务性程序》、《浮动反汇编使用说明、清单》《单用户 BASIC 解释程序框图、清单》、《检查程序》、《基本汇编程序》、《查错ⅠⅡⅢ使用说明、程序清单》、《算术库程序》、《RTOS 分析、使用说明、源程序》、《基本反汇编程序》、《浮点解释程序说明、清单、框图》、《JCY 语言》

定价：每套 27.80 元

《ECLIPSE 计算机手册》

本书据美国 DGC 公司原文译出。书中介绍了该机概况、内部结构、指令系统、输入/输出、操作面板等五部分内容，可供从事计算机设计、研制、生产和使用等工程技术人员参考。

定价：2.70 元

《电子计算机和数字设备抗干扰译文集》

本书选译了“高频通讯长线”、“测量抗扰度的组合仪表”、“高速数字设备产生假信号的某些原因”、“高抗扰度的逻辑电路设计”等九篇文章。 定价：1.40 元

部分磁盘资料

包括《ISOT 1370 磁盘驱动器说明书》、《磁盘驱动器逻辑图》、《活动头磁盘可靠性程序》、《活动头磁盘控制器诊断、说明》、《磁盘控制器、转接器说明》、《ISOT 1370 磁盘驱动器测试仪和可换、标准盘盒》、《磁盘操作系统(DOS)》 定价每套：15.20 元

其它资料

《DJS-130 机逻辑图》（每本：1.50 元）、《DJS-130 机多路通讯技术说明书、逻辑图》（每套〔二本〕：1.50 元）、《COBOL 程序设计》（南京大学计算机科学系软件教研室编。每套〔三本〕：5.40 元）、《COBOL》（译文。每套〔二本〕：4.00 元）

《PDP—11 计算机评论》（译文）

本书叙述了 PDP—11 系列小型计算机的结构、性能、机器的组成等内容。着重评论了较为典型的 PDP—11/20 和 11/45 两档机，但对其它型号机的特点也一一作了指明，还研究了其外设、选件配套等问题，并探讨了以它为基础构成的双机和多机系统方案，可供从事计算机设计、研究、生产和使用人员参考。 定价：1.20 元

上述资料，倘选作学习班教材，一次需购 50 套以上者，可先联系，售价酌情予以优惠，售完为止。来函请寄：苏州电子计算机厂情报资料室收，电话：5686(转)

苏州电子计算机厂情报资料室

目　　录

第一章　流程图

编写计算机程序的第一步就是列出要求程序完成的所有任务。这样做有助于构思和明确目的，从而为程序员指明方向和目标。

调试工作並非总是在程序编完之后才进行。如果在程序研制工作的预备阶段，花一些时间找出潜在的错误，可以为以后实际的测试和调试阶段节省大量的时间。列出一张技术规范表就能发现那些一时实现不了的目标。实现不了的原因可能是由于设计要求太全面，以致缺乏时间和资金去完成它，或者是缺乏程序设计的经验。从这样的表格中还能识别出那些非常简单，只需将准备编写的程序的功能扩大一些就能达到的目标。

写出详细的技术规范有助于发现所谓的“概念性错误”：即由于对要求程序完成的工作不清楚或理解不完整而造成的错误。

例如考虑下述这样的技术规范：“编写一个能监督和控制住房需求的程序”。

幸好没有一个程序员会接到如此含糊的指示，並要求他写出满意的程序来。但是如果程序员自以为已经知道要求的是什么，並因此而认为没有必要将其写出来时，他们就往往会不知不觉地接受这类含糊的任务。

在着手完成程序设计任务之前，总是尽量先列出可供程序使用的输入，要求的输出内容以及程序主体要完成的工作。这样做至少能节省实际程序设计工作所需的时间。除此以外，由于理清了手头的作业所涉及的事物，因此常常可能避免产生诸如读入数据时搞错输入端口或计算时搞错转换系数之类较常见的错误。

为原始的程序绘制流程图

图1—1是一个制作小甜饼过程的流程图。虽然这个流程图有若干步骤，但是，由于整个过程是从一个起点沿着一条直线执行到一个终点，所以其结构很简单。（图1—1 见下页）

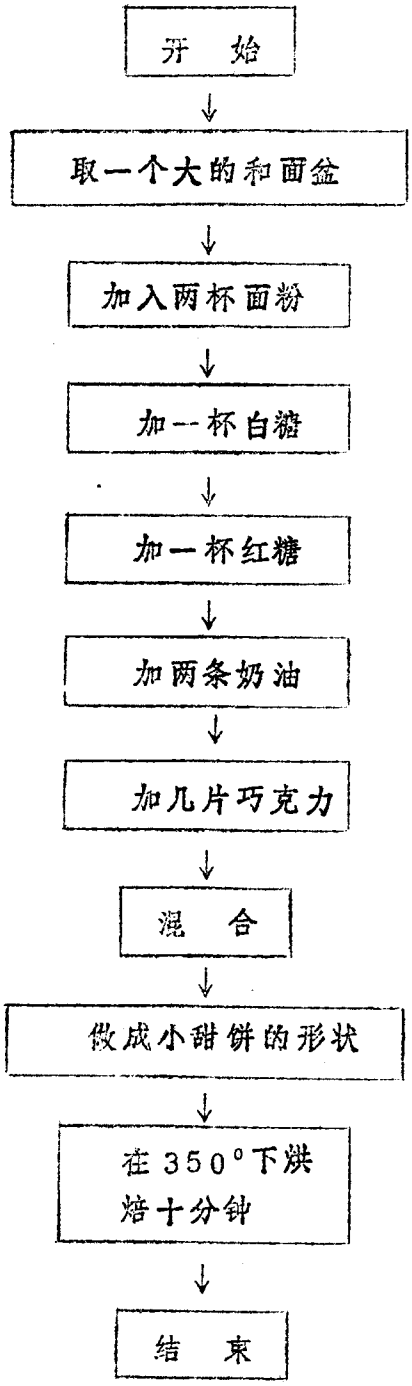


图 1—1　制作小甜饼过程的流程图

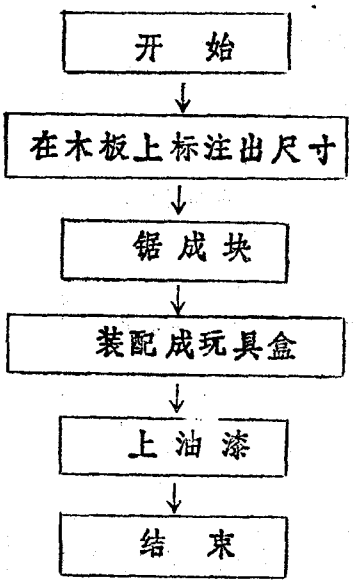
流程图是介于任务的一般说明与完成该任务所需的一系列详细而具体的指令之间的中间步骤。

图1—2以一个制作儿童玩具盒的过程为例说明了程序编制的三个步骤。该任务要求制作一个大小与旅行皮箱相仿，並可供幼儿使用足够坚固的红色木制玩具盒。可以注意到，这三个步骤的复杂程度是不同的。

技术说明：用木板制作一个玩具盒並涂上红漆。

盒子的尺寸应与旅行皮箱相仿，並足够坚固以供儿童使用。

流程图



程序：

步骤	作业
1。	做一个木制玩具盒
2。	决定侧面尺寸为18"×36"
3。	决定盖子和底尺寸为24"×36"
4。	决定端面尺寸为24"×18"
5。	取一张$\frac{1}{2}$"的胶合板

6. 对每个面进行下述步骤
7. 用一把尺
8. 量尺寸
9. 用铅笔在木板上做标记
10. 下一个面
⋮ ⋮

图 1—2 制作儿童用玩具盒的技术说明、流程图和“程序”。注意：这个“程序”不是计算机程序，而是一系列详细的制造步步骤。

技术规范描述了整个工作，並列出对最后结果提出的一些质量要求。流程图把整个工作分成若干独立的任务，並按适当的顺序将它们连在一起。“程序”步骤是最具体的步骤。例如，在图 1—2 中，程序步骤是关于各个面的尺寸，应该使用的木板厚度等等的详细指标。

由于流程图是一种图解方法，所以在程序编写之前和编写之后广泛地被用作辅助调试手段。编程时首先画出流程图，然后以流程图作为指南编写程序，这样就能避免许多潜在的错误。

例如，若将制作玩具盒的任务画成如图 1—3 所示的流程图，那会出现什么情况呢？

这样编出来的玩具盒制作程序有错误。由于遵循该程序仍可以做成一个玩具盒，所以这个错误不是“致命的”错误。但是这样做出来的玩具盒是不完善的。如果按这个流程图去做的话，那么，由于制造的第一步是在木板上涂漆，然后才在木板上划线，並用锯子锯成块，所以后面的步骤很可能毁坏木板上的漆层。

如果这真的是一个计算机程序，那么按图 1—3 所示的流程图做出来的玩具盒将被称为“废品”。

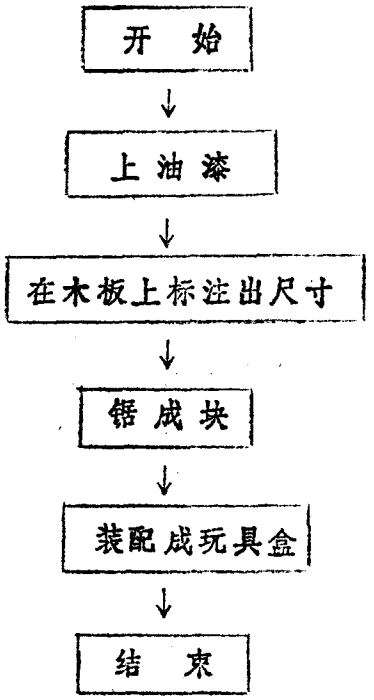


图 1—3　步骤颠倒的玩具盒制作流程图。这个次序不对的流程图会产生一个错误。

1. 流程图的符号

到目前为止，举例用的流程图都画成一系列方框，且这些方框都排成一直线。每个方框中都写明了该步骤应完成的工作；每个方框都用表示程序执行顺序的箭头与其上，下两个方框连接起来。

但是，大多数程序都不是沿着一条直线途径从头执行到尾的。程序常常会通过分支转入多个执行途径；利用循环多次执行同一段程序；完全脱离程序的主干部分去执行一个完整的子程序，然后再返回主程序。

如果希望在流程图中清楚地表示这些执行流向，就需要多种通用符号。矩形符号无法表达所有的问题，只用于表示命令语句，例如：

锯成块

以及

上油漆

流程图也能画出条件测试语句的结果。如图1—4所示，IF语句用菱形来表示。

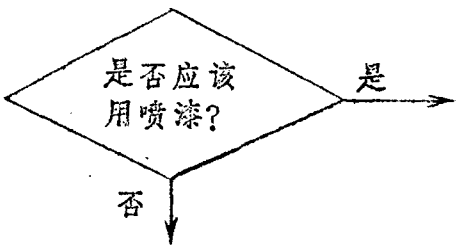


图1—4　条件语句在流程图中表示成带两个可能的走向（YES或NO）的菱形

“循环”是可供程序员使用的最灵活的概念之一。在一个循环内，一定数量的语句作为一个程序块加以执行；执行的遍数由一个循环控制变量的值决定。这个程序块每执行一遍，控制变量就增加规定的数量。当控制变量的值达到一个预定的截止值时就跳出这段程序，然后从紧跟在这段程序之后的那条语句继续执行下去。

流程图中表示循环的一般形式如图1—5（A）所示。典型的循环控制符号如图1—5（B）所示。最后还有一种通用的流程图符号：圆圈〔见图1—5（C）〕。图1—5（C）中的第三部分（标有“至A”的圆圈）是表示流程图转页或转子程序时使用的符号。

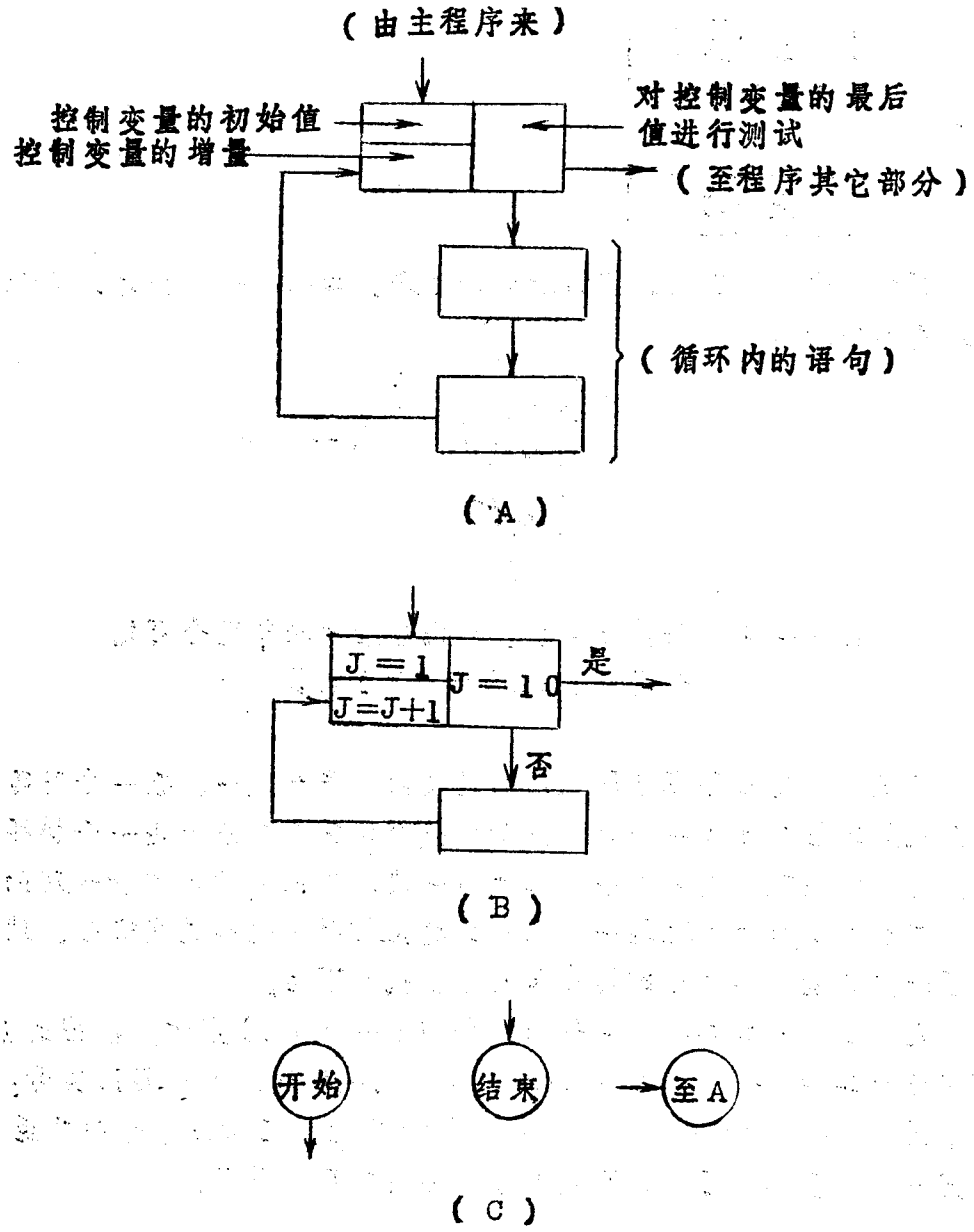


图 1—5　流程图的符号。（A）流程图中的循环。至少要有一组条件，通过对这组条件的测试使程序能返回已经执行过的点去。（B）循环控制符号的循环过程；（C）园圈在流程图中表示“端点”。

现在就可以利用这三种符号（矩形、菱形和圆圈）来绘制流程图了。在绘制流程图的过程中可以看出流程图在调试工作中所起的作用。

2．流程图符号的用法

现在先为一个非常简单的任务绘制流程图。这个任务是从终端设备上获得一组（十个）数，计算每个数的平方、平方根、立方及立方根（见图1—6）。

这个程序采用循环结构。这个循环将执行十遍，每遍读入一个数，並求出该数的平方、平方根、立方及立方根，然后转回循环的顶部使循环计数器加1；若循环变量的值小于截止值，则再次执行循环中的操作。

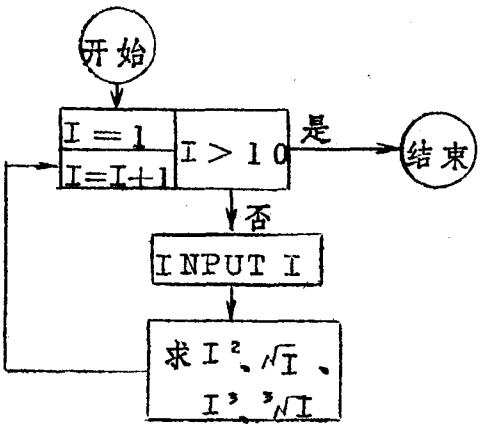


图1—6　获得十个数並计算其平方、平方根、立方及立方根的流程图

这个程序如何工作呢？

变量I被用作为循环的控制变量。变量I的初始值为1，且每通过一遍循环就加1。循环变量的值一旦变得大于10（即当I＝11时），就脱离循环。所以，循环可象予期的那样执行十遍。

但是，从键盘上读入的值被读到了变量I中去。由于变量I已被用作循环变量了，所以这是一个严重的错误。必须记住：循环中的任何操作都不得改变循环变量的值。显而易见，如果读入的第一个数大于10，那末程序在尚未读入到规定个数的数值之前就会脱离循环。

改正这个错误的办法之一是将流程图改成如图1—7所示的那样。现在这个程序至少能正确地工作。但是其中仍然有错误：由于程序没有输出，所以无法得知各次计算的结果。为了改正这个错误，可如图1—8所示在YES分支与“结束”之间插入一条命令。

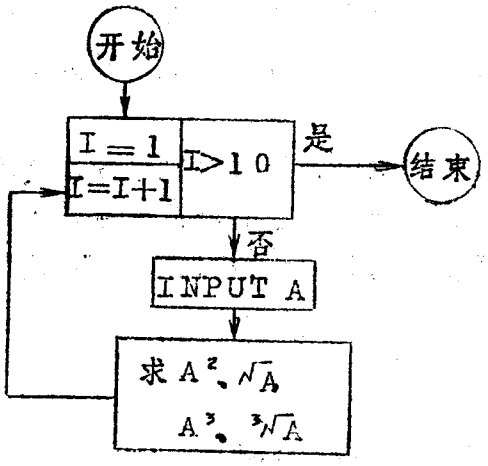


图1—7　修改后的流程图

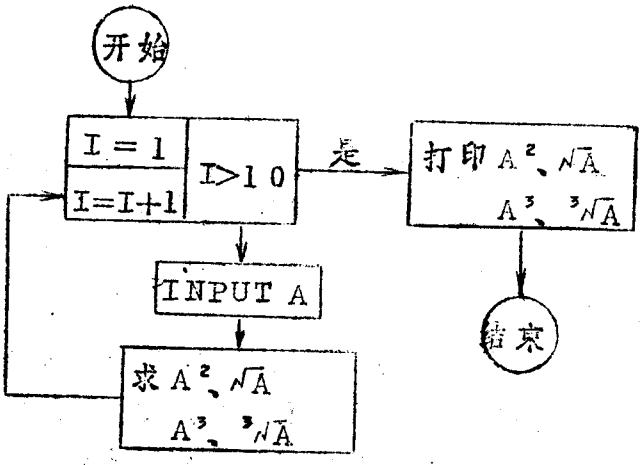


图1—8　在YES分支与“结束”之间插入纠正错误所需的打印语句

现在这个程序能执行十遍循环，每遍读入一个数值而不影响循环变量的内容。当执行完十遍以后，程序就打印出结果来。这个程序是否完善了？

遗憾的是，由于打印语句安排在循环之外，所以，只有与最后读入的值相对应的那个结果被显示出来。将打印语句移入循环中，就能纠正这个最后的错误（见图1—9）。

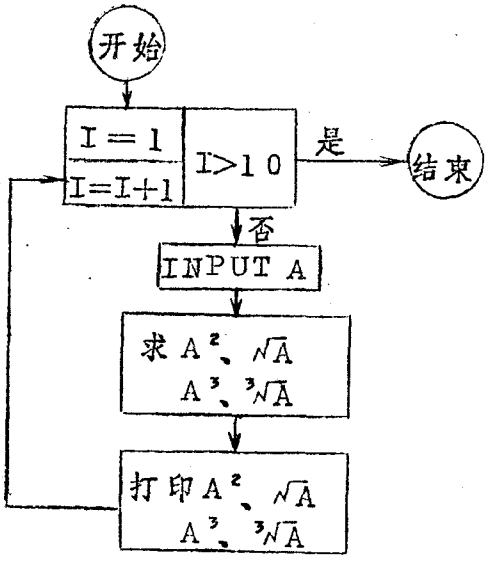


图1—9　调试的一个步骤：将打印语句移入循环中

现在这个程序能正确地工作了，但在程序中仍然有错误。然而，不是“致命”的错误，只是“外观”方面的错误。若按照该流程图的结构编写程序，则其输出就会是：

```
PLEASE INPUT A NUMBER （请输入一个数）
? 4
SQUARE IS 16.000（平方是16.000）
SQUARE ROOT IS 2.000（平方根是2.000）
```

```
CUBE IS 64.000（立方是64.000）
CUBE ROOT IS 1.587（立方根是1.587）
PLEASE INPUT A NUMBER（请输入一个数）
? 2.48
SQUARE IS 6.150（平方是6.150）
SQUARE ROOT IS 1.575（平方根是1.575）
CUBE IS 15.253（立方是15.253）
CUBE ROOT IS 1.354（立方根是1.354）
PLEASE INPUT A NUMBER（请输入一个数）
      （如此等等）
```

输出看上去很乱。但是，由于这个程序按照适当的顺序完成了予定定的任务，所以从技术上说至少可以认为该程序在流程图阶段已调试好了。但是如果程序是完全按照流程图中的格局编写的，那末就输出格式而言，严谨的程序员仍然会认为其中有错误。

采用图1—10所示的流程图就能纠正最后这一个“结构性”错误。如果严格地按照修改后的流程图编写程序，且没有犯句法或打印错误，那末程序就能完美地进行工作，其输出就很整齐：

数	平 方	平方根	立方	立方根
4	16.000	2.000	64.000	1.587
2.48	6.150	1.575	15.253	1.354

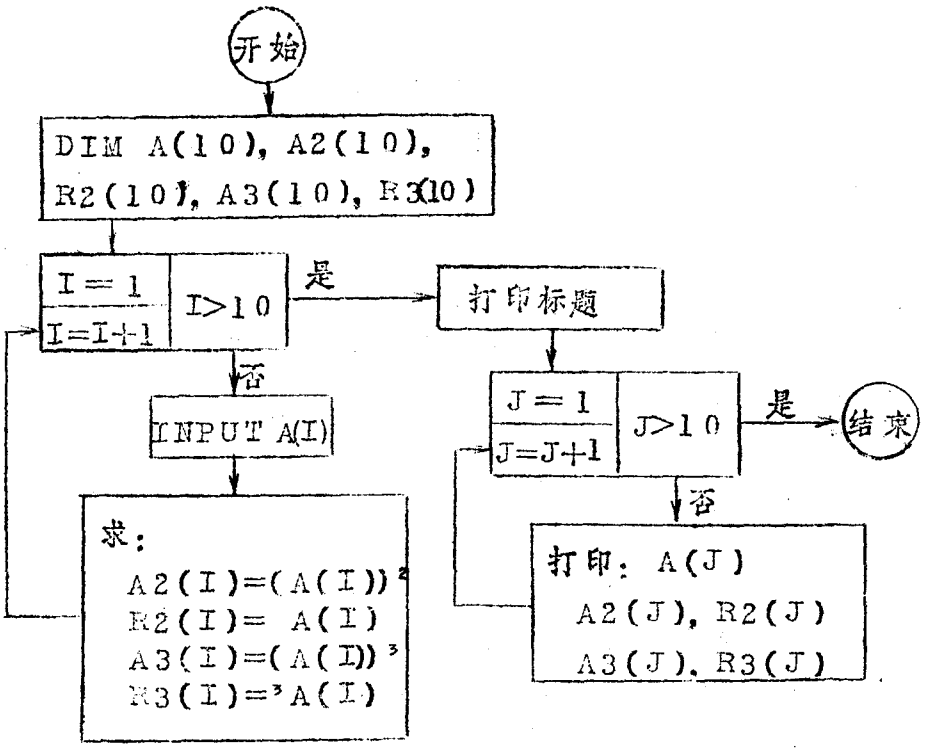


图 1—10 纠正结构性错误

为已有的程序绘制流程图

在开始着手逐行编写程序之前先画出流程图，是查找程序流程中潜在错误的一种有效办法。通常这类错误是最难查的，所以能查出来是件大好事。

同样，为已编写好的程序绘制流程图，对在程序中寻找难以捉摸的错误来说是卓有成效的。其中的关键是，流程图是以图形的形式表达程序流程的，那些根据混乱的语句编号和条件分支难以查出的错误一旦被画成图形形式，就很显眼。

现以下述程序为例进行说明。虽然对这个程序作了很好的文件编制，但是由于该程序很长，又足够复杂，所以除非画出其流程图，否