微型计算机的软件调试

Robert C. Bruce著

高文培 陈觉婷 马速成 范兴业 陈文杰 译

微型计算机的软件调试

Robert C. Bruce著

高文培 陈觉婷 马速成 范兴业 陈文杰 译

前　言

为了开发微型计算机在各行各业中的应用，使广大科技工作者，教师，管理人员和学生，尤其是计算机程序员尽快地掌握微型机的使用，我们翻译了美国一九八〇年出版的《Software Debugging for Microcomputers》一书，供广大微型机用户学习和掌握在微型机上调试程序的一套完整技术。程序的调试方法很多，但至今还没有一本专门系统地介绍程序调试方法的书，本书通过大量实例讲解如何在微型机上调试一个程序？如何发现程序中的错误？如何追踪产生错误的根源？以及如何消除程序中的错误？比较系统地介绍了最基本的程序调试技术。对广大计算机用户，尤其程序员是一本很好的参考书。本书以BASIC高级语言为基础介绍各种程序调试的技术，便于读者学习，但就调试技术来说，对用其它语言编程序也是实用的。我们基本上遵照原文翻译的，但根据实际问题，对书中个别例题做了很少的删改。

由于我们的水平所限，翻译和编辑过程中一定存在许多缺点，诚恳地希望广大读者批评指导。

微型机资料翻译组

一九八三年元月

绪　言

这是一本关于程序调试的书。其目的是在使用高级程序设计语言，例如BASIC语言（与原始的汇编语言不同），进行程序设计中，举例说明如何发现错误，追踪产生错误的根源以及如何彻底地消除错误。在计算机程序设计中“错误”（或称为故障）是为叙述不正确的运行程序所使用的一个术语。然而“调试”是纠正错误的过程。

调试任务不可避免地落在程序员的头上，这个程序员或许是或许不是该程序的原作者。虽然一些错误常常是程序投入运行之后由用户发现的，但是最好在程序的研制和试验阶段发现并修改所有的错误。在过去，尤其在微型机领域中，在小型机和大型机领域中也是一样，最后的用户和原程序员是一个人，或者至少两个都是程序员。然而今天的情况完全不同了，计算机非常普及，使用计算机的用户越来越多。微型机程序的用户多半不是程序员，很多人没有受到过计算机程序员所受过的专业训练和教育，但是他们买了微型机，而很快就厌烦了机器带来的游戏程序包，他们决心自己编程序（大多数常常用BASIC语言），因此，他们照样子学编写自己的应用程序。为了使一批用户变成程序员，调试程序是面临的最大困难，因此，如果他们想提高程序设计水平，设计先进的大型程序还须学习程序设计的技巧。

本书某些章节的内容的确作为一个整体来看，有时看来本书一方面是对程序设计师讲的，而另一方面又是对程序的

用户讲的。但无论是程序设计师还是用户，最终两者都是程序员，不同的是一个程序员在一个程序的生存周期中，开始是努力调试，在程序运行后还要负责维护。

调试技术，例如流程图技术，扦入注释语句技术，块调试技术以及在设计中扦入静态调试语句的技术，在程序的研制和试验阶段是很合适的，但是，它们对程序生存周期的实用阶段也是有效的。尤其，如果一个程序缺少流程图是很难调试的，没有内部文件资料更是如此。

反过来，模仿计算机，强迫技术，使用打印语句或在一组信息中抽点打印，以及如果需要的话扦入编码补丁是程序实用阶段所有技术的综合，在程序的试验阶段也是广泛使用的。

虽然本书假定读者已熟悉一定的BASIC语言。一般来说，在介绍和简短地讨论语言的概念和命令之后才把它们应用到特殊的例子中。然而书中的概念可以应用到用任何语言所写的调试程序中去。流程图，程序中的注释，强迫和模仿计算机等技术对于在任何计算机上运行的任何语言程序都是有效的和最基本的调试技术。实质上，所有剩下的调试技术都是在相同题目上变化打印语句。

调试一个程序时基本的要求（在一个运行的计算机上达到重复试验）是从计算机的二进制环境翻译成计算机程序员的字母数字环境。为了完成这种信息的转换，所有语言都提供了I/O(输入/输出)的功能，并且常常用PRINT或WRITE语句表示输出。因此，本书各章的内容可以应用到别的语言中去。

本书的内容分两大部分和一个总结。第一部分包括前五

章，论述基本的调试技术——调试工具。第二部分是把前面讲的各种调试方法组合起来应用到调试程序中去。最后一章是前面各章材料的总结，也是调试技术的综合发展，在最后这一章我们看到所有最好的调试工具。

由于人们编了大量的程序，他们也调试了不少的程序，从而人们得到了很多的程序设计技巧。聪明来自于经验是对的，但是很难得到完善的知识。已经搞程序设计多年的老手，从第一个UNIVAC开始几乎天天在他们的程序中追踪搜索错误。

由于我们已经很熟悉某个特殊计算机的语言，并且我们已编出很多值得信赖的程序，错误率明显地下降。然而，正是由于长期使用某种语言，尤其在有限的时间内完成复杂的程序时，常常导致一种精神上的特殊盲目性。当这种不幸的事出现时，许多人会发现没有一本即完善又非常好的书。对于这种情况以及在本书中还没有叙述的一些调试技术，一方面是因为所有其它技术都已包含在里面，另一方面也是因为对许多读者来说这些技术很简单。

对程序的编制者来说，说明一个程序是很简单的事，但往往被忽略。尤其对计算机程序懂得很少或没有受过计算机程序设计训练的人一定要详细地说明程序。给一个程序员的同伴看一个程序或给一个与程序无关的人说明一个程序常常能捉捕一些语法和与语言有关的错误。对于其它错误——由逻辑或程序结构而引起的错误，本书将能帮助和指导读者。

目　　录

常常可以使得错误率减到最小。

调试一个程序时，程序员的一个基本目的是把隐藏在计算机内部的信息取出来，以便供人们检查和分析，看其是否反映客观的实际问题，打印语句很好地满足这个要求。将打印语句加到可疑的运行程序中可以检查出程序中的错误。本章研究打印语句的使用，指出如何正确地使用打印语句来调试程序。

调试程序的目标之一是尽可能做简单的工作。尤其当程序涉及大量数学计算或条件分支时，人工进行强迫机器做某些事情是化简测试和检查过程的有效方法。化简程序调试的一种方法是尽可能用 0 或 1 去检验程序，以减少许多复杂的计算，这样可使调试的程序容易得到预期的正确结果。当那些实际结果与所期望的结果不一致时，错误几乎无疑是存在的。并且我们故意让程序执行所有分支的流程（即强迫机器执行所有可能的路线），这样通常可以查到错误，那怕是小的错误。本章讨论强迫技术和应用。

从执行单一路径的小程序到执行多路径、多分支、多个大的子程序，程序越来越复杂。本章的主题——块调试是适合在大的程序中寻找错误和修改错误的方法。块调试技术是检查除主程序外的每个独立的子程序，并使用虚拟的主程序和为某种特殊目的而编写的管理程序来调用这个子程序。这种调试技术尤其对通常根据结构程序设计原理所写的程序进行调试时更为方便，并给程序员以有效的方法来控制所要检

查的程序。

第七章　抽点打印……（202）

复杂的程序未必包含复杂的错误，但是对于程序中所包含的错误能提供比较好的隐藏错误的地点是很重要的。隐藏在程序中的错误，造成程序的故障难以查找常常是因为程序中没有详细的线索而引起的。插入打印语句的技术是程序员观察程序内部运行情况的重要手段，但是如果程序中使用了大量的变量或多路分支的执行路径，复盖着潜在的故障点，恰恰由于打印语句会导致程序长度的加倍，盲目地去打印所有变量会增加程序的运行时间，致使难以找到错误地点。扦入抽点打印是本章的课题，抽点打印技术把大量的诊断信息收集在一起，打印格式要具有易读性，然后作为与整体有关的信息来显示它，也就是有目的的显示所需要的信息。抽点打印技术给出动态地观察程序的运行情况，而不像打印语句是在程序运行结束后静态地观察，然而仍然允许程序员花费必要的时间去分析每个显示结果。用直接打印语句通常是无能为力的。

第八章　在设计中安排调试语句……（246）

在某种环境下或一组特殊情况下调试的程序，当改变环境后这个程序可能不正常运行。强迫技术企图通过对所有可能的执行路径的测试来解决这个问题，但是当路径变化很多时，强迫技术变得无能为力，因为它仅仅适用所包含的结果很简单的情况。块调试的目的是企图通过对每个子程序分别测试来解决这个问题。如果各部分都正常运行，则认为整体没问题。但有时这是靠不住的，因为不能说每个子程序调试完了就说整体没有错误存在，很可能有的子程序之间的信息

传输问题还不知道，因此块调试有时很难寻找地址。本章讨论可能的补救办法：在程序设计过程中安扦一些静态调试语句，以便在调试程序时容易查找错误的地址，即保证在故障的第一个标志点处找到错误。

有时尽管调试者的调试艺术很高，但错误仍然可能被漏掉。通常在模拟程序或博弈程序中这是常见的事，因为在这种程序中其结构复杂性更像鱼网而不像树。在这样的程序中每一次计算都可能影响程序的执行流程。使用系统的调试方法得到完全成功是很难的，例如强迫技术和块调试技术。本章讨论最后的补救办法：在程序中打补丁。一定弄懂特定的程序在做什么以及补丁达到什么目的和作用才能安全地使用这种技术。

很少仅仅使用一种技术就能成功地调试一个程序。很明显，一个程序的各个子部分用不同的方法调试处理所得到的成功的程度也不同。本章使读者通过典型的调试实例，指出使用各种调试方法常常碰到的各种错误的类型，以及例举一个程序员应如何使用所有可能的工具，各种方法的综合，在尽可能短的时间内找到准确和有力的对策，成功地调试出一个程序。

第一章 流程图

在编写一个计算机程序时，首要的一步是在纸上列出程序预期完成的所有任务。这样帮助程序员构思和弄清目的，即给程序员指出方向和明确目标。

调试程序并不全是指程序编好之后在机器上所进行的试验工作，在程序研制的最初阶段用在捕捉潜在错误所花费的时间是完全必要的，这意味着会节省许多实际测试阶段的时间，使得调试工作能够顺利进行。在调试之前写一个详细的程序规格明表是很重要的，它可以指出那些不切实际的目标，所谓不切实际的目标是指对所要处理的任务由于缺乏时间，资金或者缺少熟练的程序员等等，所制定的不现实的目标。规格说明表就像程序的任务书一样，任务要明确，说明要清楚，规格说明表与编码所达到的目标是一致的。

写一个规格说明表帮助捕捉我们称之为概念性的错误：这些错误是由不清楚或不完全理解程序做什么所引起的。

例如，考虑下面的规格说明："写一个程序监视和控制一个房屋的各种需要。"

遗憾的是没有一个程序员能处理这样含糊不清的说明和给出满意的编码。但是许多人不自觉地交出他们自己刚才还模糊不清的问题的解答，好像他们知道人家希望要做什么似的。这样做是不行的，一个程序员在接受一个任务之前必须弄清程序的规格说明。

看手一个程序设计任务之前，总是尽量列出哪些是输入变量，哪些是输出变量，以及程序实体要完成一些什么任务。如果这些都没有，拿来任务就写程序编码，看来好像节省了实际程序设计开始之前的时间，但是，这将导致在程序编制过程中浪费更多的时间，并且往往会使更多的基本错误漏掉，甚至需推倒重来，实际上延长了调试程序的周期。例如，从错误的输入点读数据或使用错误的转换因子计算一个数。

画原始程序的流程图

在编写程序之前画程序的流程图是十分重要的，这样可以避免犯结构上的错误。简单的流程图是沿着直线从单一的路径开始到单一的路径结束。但往往是具有多分支的流程图。程序的流程图又称为框图，是介于通常的任务描述和完成这个任务所需要的一步一步的指令之间的步骤。请注意规格说明，流程图和程序这三级之间的区别。规格说明描述了任务是什么并列出完成任务的要求和最后结果。流程图是将主要任务分类，即分成若干个独立的部分并按适当的顺序把它们连接起来。程序步是最具体的细节。

因为框图是一种用图形直观表示任务的方法，所以无论是在编写程序之前还是在其后都是被广泛地用来捕捉程序中隐藏错误的有效方法。所以流程图是调试程序的重要手段。

流程图的组合块

到目前为止，我们所看到的程序框图的例子都是由一系

列直线段和方框组成的。每个方框描述了要执行的步骤，用带箭头的线段连接每个方框，以表明程序所执行的次序。然而，大多数的程序从开始到结束的执行路线都不是单一的直线进程。往往程序是多路分支的执行路线，同一段编码循环执行若干次，或离开主程序的控制去执行一个完整的子程序后再返回。

如果要清楚地描述程序的流程，单用矩形框是不够的，还需要选择一些通用的符号。我们通常用矩形框表示命令语句，用菱形框表示判断语句。如图1—1表示一个条件语句。

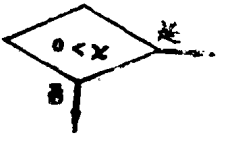

图1—1 条件语句，表示两种可能的选择——是或否

循环是一个程序员使用最多也是最重要的概念。在一个循环体中，包括许多执行语句作为一块，用循环控制变量的值控制编码块执行若干次，这个循环控制变量每执行一次增加一个规定的量，当循环控制变量达到预先规定的终止值时，则循环体的编码块停止执行，程序继续执行循环体下面的语句。图1—2（A）给出循环程序框图的一般形式。图1—2（B）给出典型的循环控制变量的符号表示方法。图1—2（C）给出一些流程图的符号。其中图1—2（C）的第三个符号表示转到A去执行，通常用它表示从一张框图的某个地方转到另一张框图的某个地方或跳到某个子程序去执行。

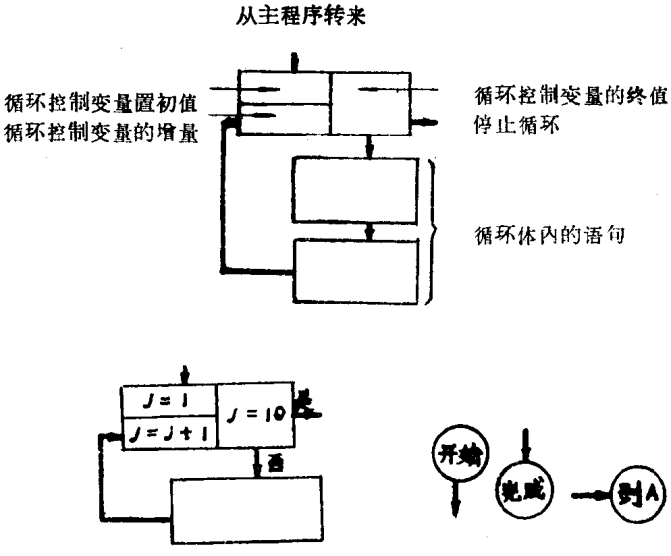

图1－2 （A）（B）循环程序的框图。（C）通用符号

流程图块的使用

我们从一个非常简单的任务开始：从终端上输入10个数，对于每个数，计算它的平方、平方根、立方和立方根（图1—3）。

这个程序循环执行10次，每次通过循环体读一个数，并求这个数的平方、平方根、立方和立方根，然后再回到循环头，循环计数器增加1；如果

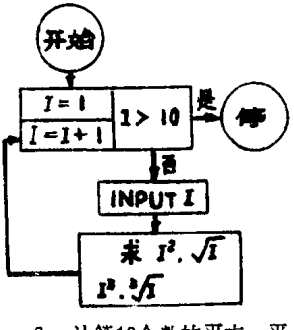

图1－3 计算10个数的平方，平方根，立方，立方根框图

循环控制变量的值小于等于终值，则循环再执行一次，否则停止执行。即图1—3中的循环控制变量I的初值为1，并且每循环一次增加1，当循环控制变量I>10，换句话说I=11时，转出循环，因此循环体被执行10次正好停止。

但是，从键盘输入给变量I的值已事先做了图1—3中计算10个数的平方，平方根，立方和立方根的循环控制变量，因此，这是一个严重的错误：即循环控制变量的值绝不能在循环体里面跟着变化。很明显，如果读入的第一个数大于10，程序将转出循环，而不能正确运行。

对上面发现的错误我们修改成如图1—4所表示的框图。现在至少这个程序可以正确地运行，但是仍然有错误：因为没有输出，所以没有办法知道计算的结果。于是我们在“是”这个分支与“停”之间插入打印语句，如图1—5。

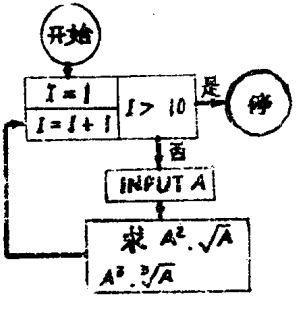

图1—4 为消除错误修改后的框图。

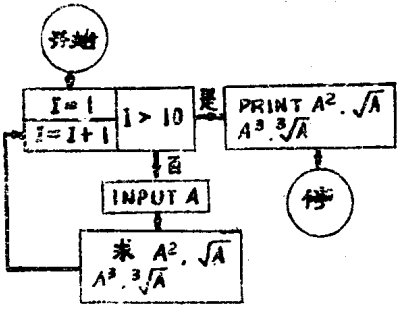

图1—5 插入打印语句后的框图。

现在程序能执行10次，每次读入一个值，这个值已不影响循环控制变量的内容，并且当程序全部执行完时，打印出

结果。现在程序完全正确吗？

很遗憾，只打印出最后读入的数的运算结果，因为打印语句被放在循环体的外面，所以只打印一次。因此，必须把打印语句移到循环体中，如图1—6所示。

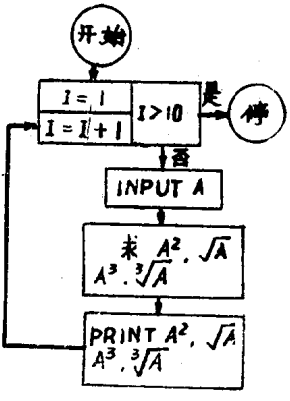

图1－6　打印语句移到循环中。

最后，这个程序将能正确地运行。还有一个缺欠，但这是对程序“美化”的问题，而不是致命性的错误。按照上面框图所编的程序输出情况如下：

```
PLEASE INPUT A NUMBER
? 4
SQUARE IS 16.000
SQUARE ROOT IS 2.000
CUBE IS 64.000
CUBE ROOT IS 1.587
PLEASE INPUT A NUMBER
? 2.48
SQUARE IS 6.150
SQUARE ROOT IS 1.575
CUBE IS 15.253
CUBE ROOT IS 1.354
PLEASE INPUT A NUMBER
（等〃）
```

这种输出格式很不好看。从技术上讲，调试一个程序

（至少在框图阶段）做到上面那样也可以了。但是，从美观来讲，尽管程序是正确的，一个好的程序员仍然认为这个程序有缺陷。最后，这种结构和格式上的缺陷经过修改如图1—7所示。如果程序编码与修改后的框图一致，以及如果没有语法和印刷错误，这个程序的功能更完美，输出格式看起来更漂亮。输出格式如下：

NUMBER	SQUARE	SQ.ROOT	CUBE	ROOT
4	16.000	2.000	64.000	1.587
2.48	6.150	1.575	15.253	1.354

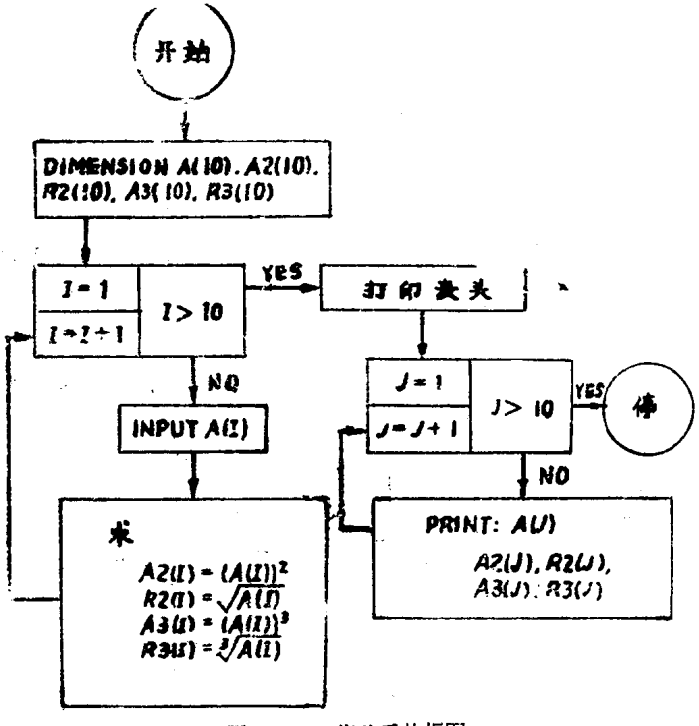

图1－7　修改后的框图。