BASIC语言的应用

——解题方法

庞绍义 编译
王以和 审校

西安交通大学出版社

内 容 提 要

本书大部分内容译自〔美国〕 Richard Dillman 所著的《PROBLEM SOLVING WITH BASIC》一书。主要讲授程序设计方法。全书共分十三章，每章基本上都分为三节。第一节为“逻辑”，重点讲算法设计，这是程序设计的核心，也是本书的核心。它包括如何分析问题，分析数据，找出处理规则，以及构造算法。第二节讲“语法”，主要介绍 BASIC 常用语句的基本规则。第三节是“解题示范”，讲授如何以语言为工具把算法翻译为程序，介绍解题的全过程。书中通过实例深入浅出地介绍了结构化程序设计，自顶向下程序设计，模块化程序设计，逐步求精法以及怎样加注释等技术。本书的宗旨是提高读者的程序设计能力，并使其规范化。

本书可作为大专院校学生的教材。也可供学习过 BASIC 语言后希望提高程序设计能力的各方面读者阅读。

BASIC 语言的应用——解题方法

编　　译：　庞绍义
审　　校：　王以和
责任编辑：　林　全

*

西安交通大学出版社出版
（西安市咸宁路 28 号）
西安交通大学出版社印刷厂印装
陕西省新华书店发行　各地新华书店经售

*

开本 787×1092 1/32　印张 10　字数 210 千字
1986年12月第一版　1987年6月第一次印刷
印数 1——5,000 册

书号：ISBN7-5605-0033-1/TP-6
15340·118　　定价：1.45 元

前　言

凡是学习过程序设计语言的读者都有一个共同的感受，学习一种计算机语言并不困难，但要是自己独立地编写一个程序(哪怕是一个并不复杂的程序)往往都会感到无从下手。究其原因，主要是因为程序设计需要两方面的知识——即算法设计和语法规则。程序设计语言主要讲授的是某种语言的语法规则，而不可能详细讲授程序设计的方法即算法设计。缺少了算法设计技能的结果，就是不知怎样编写一个程序，当然更谈不上更快地编写一个高质量的程序。为了帮助广大读者提高程序设计的技能，我们编译了本书。目的在于培养读者分析问题和解决问题的能力，掌握程序设计技术，推广和普及计算机应用，为四化建设服务。当然，程序设计不仅是一门艺术而且是一门科学，要掌握这门科学必须经过严格规范的训练，只有经过大量的实践才能把它真正学到手。读者自己编写的程序越多，程序设计方法掌握得也就越好。

本书各章基本上都分为三节，采用的是“逻辑——语法——解题示范”的顺序编写。“逻辑”一节研究解题方法，其中包括：问题分析，数据分析，寻找处理规则，算法设计等内容。算法设计是程序设计的核心，也是本书的核心，它是编写出好的程序的先决条件。“语法”一节讨论 BASIC 语言常用语句的基本语法规则。书中对 BASIC 语言只作复习性的讲授，这是考虑到学习本书的读者一般都已经学习过

BASIC 语言。“解题示范”一节把算法设计和语法规则通过一个实例结合起来，讲述如何把算法翻译为 BASIC 程序，从而介绍解题的全过程。本书实例以日常生活中的问题为主，很少涉及复杂的数学问题。

本书将算法设计和语法规则分开讲授，其一，是为了突出算法设计，在算法设计期间暂不考虑语法；其二，是为了说明算法的独立性。因为算法只与题目有关，它与使用哪一种程序设计语言无关，所以本书的“逻辑”部分对学习过其它程序设计语言的读者也很有启发和帮助。“语法”部分没有讲授任何一种具体计算机的 BASIC 语言。书中的BASIC语言内容大部分取材于“标准”BASIC文本，还有一些吸收自各计算机生产厂家的 BASIC 语言。这样可以摆脱具体机型的一些限制，以便把注意力集中在算法设计上面。

书中以实例深入浅出地讲授了七十年代以来最为流行的程序设计新技术，其中包括：结构化程序设计，自顶向下程序设计，模块化程序设计，逐步求精法，以及怎样加注释，怎样写程序说明书，怎样组织一个大题目的程序设计等等。

在本书编译过程中，承蒙王以和教授多次指导，并在百忙中挤出时间仔细审校。王运路同志通阅了全书初、终稿，提出了许多宝贵意见。同时得到了校内许多师生的关心和帮助，出版社的大力支持，在此谨向他们表示衷心地感谢。由于时间仓促，本人水平所限，错误在所难免，敬请读者批评指正。

庞绍义

1985年 6 月于西安交大

目　录

第一章 序 论

§1. 引 言

仅读本书还不能使你成为一个专业的程序设计者，只有经过大量的程序设计实践才能使你达到这一目标。本书的目的是教你怎样分析问题，如何应用程序设计技术，这些技术是专业程序设计者公认的、编写“优秀”程序的最有效的方法。

§1.1 程序设计的学习方法

程序设计的方法不是唯一的。虽然近十多年来对程序设计的方法进行标准化已经取得了不少成绩，但是程序设计不仅是一门艺术而且是一门科学。要掌握这门科学，运用它去解决实际问题，就必须掌握分析问题和编写程序这两方面的知识。只靠读一本书或听听别人的谈论，就想学会程序设计是不现实的，只有经过大量的实践，最好是在计算机上亲自动手来学习和掌握。

初学者就更应该花费大量的时间去使用计算机。真正想把程序设计技术学到手，就应该自己编写一些小程序并上机调试，以此来检验和巩固各部分所讲授的新内容。这是学好程序设计的唯一方法。读者自己所编写的程序越多，程序设计技术就会掌握得越好。

§1.2 逻辑与语法

本书各章差不多都采用了“逻辑——语法——解题示范”的顺序编写。“逻辑”部分研究解题方法，数据分析，逻辑结构及其表示方法。这是设计算法所必需的。“语法”部分讨论 BASIC 语言的语法规则。“解题示范”部分给出完整的程序，它举例说明前面讨论的内容。

算法是程序设计的核心，而算法的设计也是较难的内容。程序设计语言的语法规则是不变的，必须严格遵循，相对来讲语法是比较容易掌握的。逻辑设计确是需要认真对待的，因为即使是一个很简单的问题，算法也有多种，而且每一种算法都能产生可接受的结果。初学者往往体会不到这一点，他们注意的往往只是“交出程序”来。

编写完程序后再回顾一下程序的结构与风格是大有好处的，这样可以看出自己的程序设计技术与“优秀”的程序设计技术相比差距何在。程序设计者还应该在上机之前纠正程序中的错误。对个人来讲，自己查出的错误比全靠计算机查出错误收获要更大。

本书是将算法设计和 BASIC 程序编写分开来写的。“逻辑”部分讲算法设计技术而不涉及 BASIC 语言。这样安排是为了强调算法设计与程序编写之间的差别，同时又要求读者在算法设计时考虑伪代码中的问题而不是 BASIC 语言中的问题。“逻辑”一节讨论算法设计，紧接着的下一节就是“语法”，在这一节里讨论实现前一节的算法设计所需要的 BASIC 语句。然后再在“解题示范”一节用 BASIC 语言实现算法，从而写出一个完整的能运行的 BASIC程序。这种编写方法能使读者逐步看清，设计一个好的算法是编写

一个好的程序的先决条件。

§1.3 本书所用的BASIC

本书的例题没有使用任何一台具体计算机的 BASIC 语言。书中所用的 BASIC 大部分选自“标准” BASIC，还有一些吸收自各计算机生产厂家的 BASIC。这样可以不受任何一种具体机型 BASIC 语言的束缚（因此可使文本简单，易于理解和仿效），同时还可以使读者注意到，要想实际进行程序设计，就得阅读自己所用计算机的 BASIC 语言参考手册。

§1.4 程序结构与风格

当代程序设计的主要注意力集中在结构与风格互补的思想上。“结构”指的是解决问题的途径，算法设计和程序编写，所有的程序都是由称之为“结构”的若干基础逻辑模块构成的。结构主要有三种，即线性序列结构，重复结构和判断结构。虽然使用结构的方法对好的程序设计不一定非要不可，但它确实使程序或多或少地符合“规范”的形式。由以后的讨论可见，程序符合“规范”的形式是程序设计行业的一个主要目标。程序设计风格涉及到所编程序可见的外貌情况。从根本上讲，一个风格良好的程序应该逻辑简单，有详细的说明，并且容易阅读。

强调程序结构与风格是因为商业（作为商品出售的）程序不仅要实现用户提出的要求，而且必须容易修改。重要的是读者要知道这类程序一般都很大，具有长久地使用寿命，并且要经常修改。（教室里的程序常常不具有这些特点）因为软件的成本主要是人工，并且因为现有软件的维护是一直需要的，所以大家都承认，结构化的、风格良好的程序设计

是以低成本生产出高效软件的最可能的途径。有志于从事程序设计行业的读者应该知道，“使程序能工作”只是评估程序设计的许多因素中的一个因素而已。

§1.5 自顶向下设计

如上所述，初学者应该花大量的时间在终端上使用计算机。要通过使用来熟悉操作系统，编辑程序，BASIC 语法（特别是它的“错误信息”），但是也必须强调，优秀的程序设计者并不是在终端上写程序的。初学者常常弄错“程序设计”的全部含意，疏忽了算法与程序是两回事。大多数的课程（本书也不例外）都是从初看起来很难的简单问题入手，最终很容易地予以解决了。因此初学者往往以为所有的程序都是这样编写的，即打入计算机一组 BASIC 语句，然后进行调试，直到得到正确的结果。

当然，这是因为早期的问题太简单而产生的一种幻想。二十行以下的程序可以这样进行（即使这样简单，优秀的程序设计者也不主张这样干），但是程序行数哪怕是很少的增加都会明显地增加程序的复杂性。不认识这一点的人一定会发现，他在一个程序上尽管日复一日的调试，但并无多大进展。本书把算法设计的内容与语法的内容分开，是为了强调需要把题目分析和算法设计作为编写程序的先决条件。而且在算法之前首先进行问题分析，这样就可以自顶向下地推导出解来。

“自顶向下设计”、“模块化设计”和“逐步求精”已经成为一种流行的同义词了，但它们常常被错用或错误理解。本质上讲它们包含的概念很简单，即首先以最大可能的概括说法来描述一假定性的解。这样做便于抓住问题的总体

情况。这种“顶级”的描述一旦建立以后，就要对它重复地进行考察，以便用更详细的指令来逐步取代概括性的描述。请看下面这样概括性的句子：

我要去商店买一盒糖果。

考察这一描述，可以看出它完全包括了打算做的事。这是第一级的描述。为了得到第二级，我们发现第一级中的某些描述还比较含糊。确切地说，“去商店”和“买一盒糖果”是怎么回事呢？让我们来研究“去商店”。

第一级　去商店

第二级　步行去公共汽车站

乘7路公共汽车

在大差市站下车

现在我们把细节加进了上面的描述。虽然有些仍不十分清楚（例如“乘7路公共汽车”是怎么回事），但其方向要比第一级清楚得多，而且这里所加的细节都归属于原来总体概括性的句子。自顶向下设计的实质就是允许问题求解者作出详细的解法而又同时保持整个问题清晰明确。我们还可以再细分下去。

第二级　乘7路公共汽车

第三级　在车站等候车辆开来

如果来的车辆是公共汽车，看是几路车

如果是7路，等车进站

车门打开后，上车

买车票

如果有座位就坐下，否则就站着

现在你可以看到，例如“买车票”还需要进一步定义，于是

这个过程还可以再继续进行下去。

当然上面这个例子并不是一个典型的程序设计问题，但思路是相同的。学会这样思考问题的人就很有希望成为一个有能力的程序设计者。而不按这种方法思考问题的人则很难写出工作良好而又易于修改的程序来。

§2. 程　　序

§2.1 编写程序

计算机可用来打印信息，求出一些数据之和，把一组单词按字母顺序排列等等。但是，计算机自己一件事也不会做。只有我们给它提供了详细的指令，准确地告诉它如何执行每一个任务之后，才能指望它给我们做某件事情。这些指令的集合就叫程序。程序是用计算机能“懂得”的某种语言编写的。

选择求解一个特定题目的一组指令并非总是易事。题目往往是用自然语言或文字来叙述的，在一个题目能被编写程序并由计算机执行之前，必须把它从自然语言翻译成为计算机能“懂得”的语言。这一工作是由程序设计者完成的。

然而，程序设计者并不只是当翻译，他们还是问题的求解者。因为计算机不会思考，所以它不能解题，而只能执行指令。只有在程序设计者有了解决该问题的方法，并且把这种方法教给计算机之后，计算机才会解这个题。

所以，读者可以发现，任何程序的编写都包括了两部分：

1. 解题。程序设计者必须提出计算机能执行的一系列解题步骤，才可能得到该问题的解。这一过程叫作研究程序

逻辑或叫设计该程序。

2. 程序设计者把这一系列解题步骤翻译成一种计算机语言的程序。在翻译过程中，必须遵循该语言的语法规则。这一过程叫作编写和调试程序。

语法规则就是计算机对要执行的指令进行语法检查时所遵循的规则。不同的计算机语言其规则也不一样，本书讨论的是 BASIC 语言，所以我们只讨论 BASIC 语言的语法。

§2.2 执行程序

计算机的型号很多，不同型号的计算机操作步骤也不一样。当你把一个程序写好以后，应该查阅一下你所使用的计算机的用户参考手册，看看怎样输入程序，如何通知计算机开始执行这个程序等。计算机用户手册中包括了该计算机的全部操作命令。如果计算机打印出了某些你不懂得的信息，或者你要作某种操作但不知怎样作的时候，通常都可以在计算机用户手册中找到答案。

§2.3 BASIC 语法错误

当你通知计算机执行你的程序时，它首先要对程序进行语法检查，看看程序是否完全遵循了该计算机的 BASIC 语法规则。如果查出了程序中有违反语法规则的语句，计算机就打印出语法错误信息，告诉程序有错。

计算机执行你的第一个程序时，可能会打印出很多的错误信息，你决不能因此而灰心。计算机只接受完全正确的语句。即使是一个有经验的程序设计者，也免不了出错，也得花很多时间去改正错误，这就是调试程序。

你会发现大部分语法错误都是由于按键操作错误所引起的。例如，你本来要打入 READ 但打成了 REMD，计算机

就作为一个错误识别出来。此时你可以很简单地重新打入这个单词即可改正这个错误。如果计算机指出在某一行有一个错误，但你看这一行却没错，此时你就应该查阅 BASIC 语言参考手册，搞清错误的原因。如果查阅参考手册后还搞不清楚，那就应该请教对你有帮助的人。往往别人可以很快发现自己看不出来的错误。

直到所有的语法错误都改正以后，计算机才能正确地执行你的程序。但是，一个程序可以运行了并不等于它就是完全正确的。你还得检查输出，看看它是不是你所要求的结果。

要是程序的输出不对，还得再回过头去检查程序逻辑。这种检查的最好方法是取一枝铅笔和一张纸，逐行检查程序，跟踪每一行所发生的情况，就好象你是一台计算机在执行这个程序一样。这是一项缓慢而冗长的工作，但如果做好了，往往总是可以查出程序中的错误的。

§2.4 程序实例

在下一章中，我们将讨论解题方法，你就会看到程序设计者应该如何研究程序逻辑。这里是一个完整的 BASIC 程序例子，它给你提供了一个完成了的程序是什么样的感性认识。

```
10  REM THIS PROGRAM COMPUTES
    THE SUM OF
15  REM THE FIRST 10 NUMBERS
20  REM (本程序计算前 10 个数之和)
30  N = 1
40  S = 0
```

```
 50  IF N>10 THEN 90
 60  S=S+N
 70  N=N+1
 80  GOTO 50
 90  PRINT"THE SUM IS",S
100  END
```

程序 1.1

这个程序求 1+2+3+…+9+10 之和。如果把它输入计算机并执行它，将在屏幕上显示出这样的结果：

THE SUM IS 55

作为练习请把此程序输入你的计算机并执行之。