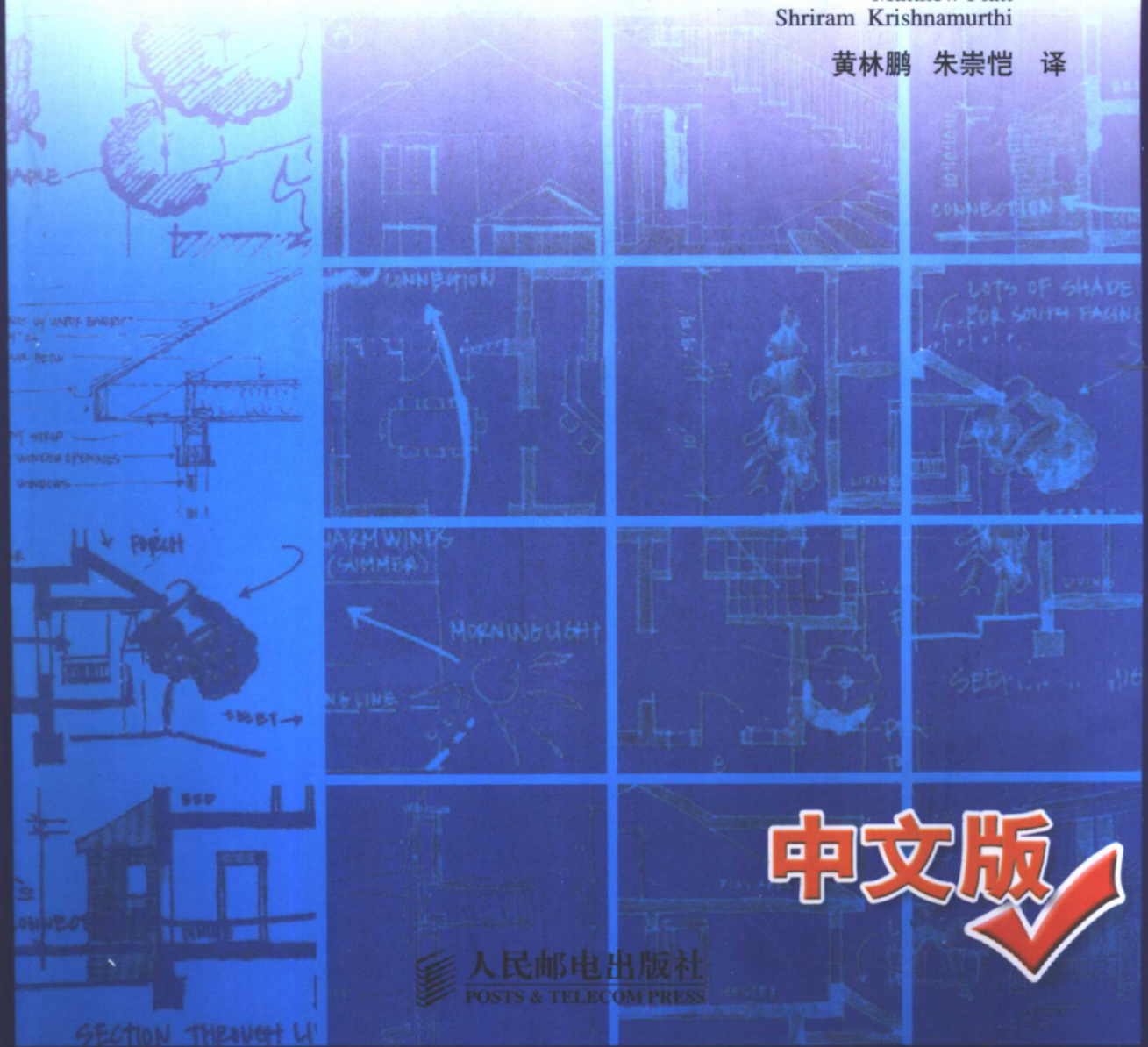
国外著名高等院校
信息科学与技术优秀教材

程序设计方法

How To Design Programs

〔美〕 Matthias Felleisen
Robert Bruce Findler
Matthew Flatt
Shriram Krishnamurthi
著

黄林鹏 朱崇恺 译



中文版

人民邮电出版社
POSTS & TELECOM PRESS

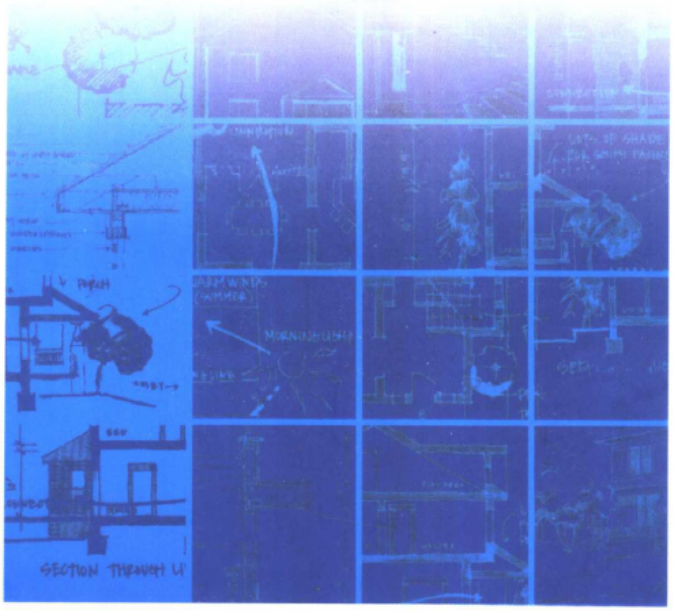
封面设计：胡平利

国外著名高等院校信息科学与技术优秀教材

程序设计方法

How To Design Programs

中文版



ISBN 7-115-11556-7

9 787115 115560 >

人民邮电出版社网址 www.ptpress.com.cn

ISBN7-115-11556-7/TP·3578

定价：49.00 元

国外著名高等院校信息科学与技术优秀教材

程序设计方法

Matthias Felleisen

[美] Robert Bruce Findler 著

Matthew Flatt

Shriram Krishnamurthi

黄林鹏 朱崇恺 译

人 民 邮 电 出 版 社

图书在版编目（CIP）数据

程序设计方法 /（美）傅雷森（Felleisen,M.）等著；黄林鹏等译.
—北京：人民邮电出版社，2003.12
ISBN 7-115-11556-7

Ⅰ.程… Ⅱ.①傅… ②黄… Ⅲ.程序设计—方法 Ⅳ.TP311.11

中国版本图书馆 CIP 数据核字（2003）第 074636 号

版 权 声 明

国外著名高等院校信息科学与技术优秀教材

程序设计方法

◆ 著 [美]Matthias Felleisen Robert Bruce Findler
Matthew Flatt Shriram Krishnamurthi
译 黄林鹏 朱崇恺
责任编辑 陈冀康

◆ 人民邮电出版社出版发行 北京市崇文区夕照寺街 14 号
邮编 100061 电子函件 315@ptpress.com.cn
网址 http://www.ptpress.com.cn
读者热线 010-67132705
北京汉魂图文设计有限公司制作
北京隆昌伟业印刷有限公司印刷
新华书店总店北京发行所经销

◆ 开本：787×1092 1/16
印张：30
字数：925 千字 2003 年 12 月第 1 版
印数：1-4 000 册 2003 年 12 月北京第 1 次印刷

著作权合同登记 图字：01－2002－1069 号

ISBN 7-115-11556-7/TP・3578

定价：49.00 元

本书如有印装质量问题，请与本社联系 电话：(010)67129223

内容提要

本书以 Scheme 语言为基础介绍计算和程序设计的一般理论和实践。

本书由 8 个部分和 7 个独立的章节（第 8、13、18、24、29、33、38 章）组成。8 个部分主要讨论程序设计，独立章节则介绍一些与程序设计和计算相关的话题。本书第 1 至第 3 部分介绍了基于数据驱动的程序设计基础。第 4 部分介绍了程序设计中的抽象问题。第 5 部分和第 6 部分是与递归及累积相关的内容。本书的最后两部分说明了设计程序的意义，阐述了如何应用前 6 个部分所描述的程序设计诀窍，以及使用赋值语句必须特别小心的一些问题。

本书可作为高等院校计算机科学与技术专业“程序设计导论”和“计算导论”的教材和教学参考书，也可作为函数式语言和 Scheme 语言的入门教材。

前 言

向儿童传授程序设计知识有悖于现代教育学。制定计划、学习教规、注重细节、严格自律有何乐趣?

——艾伦·佩利（1966年图灵奖获得者），《编程警句》

许多职业都需要进行某种形式的计算机编程。会计师使用电子表格和字处理软件编程，摄影师使用图片编辑器编程，音乐家使用音响合成器编程，职业程序员使用计算机编程。编程已成为一种人人都需要掌握的技能。

编写程序并不仅仅是一种职业技能。事实上，好的编程是件有趣的事，是一种创造性的情感发泄，也是一种用有形的方式表达抽象思维的方法。程序设计可以教会人们多种技能，如阅读判断、分析思考、综合创造以及关注细节，等等，这些技能对各种类型的职业来说都是重要的。

所以，在普通教育中，程序设计课程的地位应该和数学、语文一样重要。或者用更简洁的话来说，就是

每个人都应该学习如何设计程序。

一方面，程序设计跟数学一样，可以训练人的分析能力，不同的是，程序设计是一种积极的学习方法。在与软件的互动过程中，学生可以直接得到反馈，进行探索、实验和自我评价。与钻研数学习题相比，程序设计的成果，即计算机软件，更有趣，也更有用，它们能极大地增加学生的成就感。另一方面，程序设计跟语文一样，可以增强学生的阅读和写作能力。即使是最小的编程任务，也是以文字形式表达的，没有良好的判断和阅读技能不可能设计出符合规范的程序，反之，好的程序设计方法会迫使学生用适当的语言清晰地表达他的思考过程。

本书是基本的程序设计教科书，讨论如何从问题描述产生组织严谨的程序。本书把注意力集中于程序的设计过程，不强调算法和语言细节，不注重于某个特定的应用领域。这门介绍性的程序设计课程有两个根本性的创新。创新之一是给出一系列明确的程序设计指导。现有的程序设计课程往往趋向于给出含糊的、不明确的建议，如“自上而下设计”或者“结构化程序设计”等。与此不同，本书给出了一系列程序设计指导，由此引导学生一步一步地从问题的描述出发，通过明确定义的中间过程，得出程序。在这个过程中，学生将学会阅读、分析、组织、实验和系统思维能力。创新之二是使用了一个全新的程序设计环境。过去的编程教材往往简单地假设学生有能力使用某种专业程序开发环境，而忽略程序设计环境对学生学习的影响。本书为初学者提供的程序设计环境会随着你所掌握的知识的多少而改进，该环境最终可

以支持完整的 Scheme 语言，使用该语言既可以编写大型程序又能编写脚本程序，可以完成所有领域的编程任务。

本书讨论的编程指导以程序设计诀窍（programming design recipe）阐述[1]。设计诀窍指导程序设计初学者逐步掌握问题求解的过程。有了设计诀窍，程序设计的初学者就不用再盯着空白的纸张或计算机屏幕发呆了，他们可以自我检查并核对设计诀窍，使用“问答”方式进行程序设计并取得进步。

本书通过识别问题的范畴来建立设计诀窍，而问题范畴的识别基于表示相关信息的数据类型。从该数据类型所描述的结构出发，你可以用一个清单推导出程序。图 1 给出的设计诀窍包含了程序设计的 6 个基本步骤，每个步骤都将产生定义明确的中间结果：

1. 问题数据类型描述；
2. 程序行为的非正式描述；
3. 说明程序行为的例子；
4. 开发程序的模板或视图；
5. 把模板转换成完整的定义；
6. 通过测试发现错误。

程序设计决窍的基本部分
1. 问题分析和数据定义
2. 合约，用途说明与结果的描述，函数头部
3. 例子
4. 函数模板
5. 函数定义
6. 测试

图 1 程序设计决窍的基本步骤

主要差异在于第 1 步和第 4 步之间的关系。

使用设计决窍不仅对初学者有所帮助，对教师也有益。教师可以使用清单检查初学者解决问题的能力，诊断错误所在，并提出具体的纠正措施。毕竟，设计决窍的每一阶段都会产生一个定义明确、可检查的结果。如果一个初学者遇到了困难，教师可以借助清单检查他的中间结果，并判断问题之所在。教师还可以针对程序设计决窍中某一特定的过程给学生提供指导，提出合适的问题，并推荐额外的练习题。

为什么每个人都应该学习编写程序

想象会把不知名的事物用一种形式呈现出来，诗人的笔会
使它们具有如实的形象，空虚的事物也会有了居处和名字。

—— 莎士比亚，《仲夏夜之梦 V(i)》

目前越来越少的人在编写程序代码，主张每个人都应该学习编程似乎有些奇怪。事实上大多数人是在使用应用程序包开发软件，即使是程序员也使用“程序生成器”由规则（如商业规律）创建程序，看起来他们似乎不需要编写代码。那么，为什么还说每个人都应该学习编程呢？

问题的答案可以从两个方面阐述。第一，传统形式的编程确实仅仅对少数人来说是有用的。但我们这里所讨论的编程模式对每个人，不管是使用电子表格的行政办公室秘书还是高科技公司的程序员，都是有用的。换句话说，这里所讨论的编程概念远比传统的编程观念广泛。第二，本书以最小影响原则来

[1] 那些熟悉 C/C++、Basic 和 Pascal 等程序设计语言的读者可以将前言中提到的程序（program）理解为过程（procedure）或方法（method）。

讲授编程思想，着重于分析问题和解决问题的技能，而不是强迫大家掌握传统的编程语言和编程工具。

要想更好地理解现代编程思想，请仔细观察一下目前流行的应用程序包，如电子表格。如果用户先把描述一个单元 A 和另一个单元 B 依赖关系的公式输入电子表格，接着，输入单元 B 的值，电子表格就会自动计算单元 A 的值。对于复杂的电子表格，一个单元的值可能依赖多个其他单元，而不仅仅是一个。

其他应用程序包也需要类似的计算。考虑文字处理和样式表软件。样式表说明了如何由待定的词或句建立一个（或一部分）文档。当提供了特定的词句之后，文字处理软件就会把样式表中的名字替换为特定的词句，从而建立文档。类似地，某个进行网页检索的人可能会指定若干个关键字、给定关键字之间的顺序以及哪个关键字不必在网页中出现。在这种情况下，搜索结果将取决于搜索引擎的高速缓存和用户所输入的检索表达式。

最后，使用程序生成器的技巧其实就是使用应用程序包的技巧。程序生成器由高层功能描述生成传统程序设计语言代码，例如由商业规律或科学定律产生 C++或 Java 程序。规律把数量、销量以及库存记录联系起来并说明计算过程。而程序的其余部分，特别是如何与用户交互以及如何将数据存储于计算机磁盘等等，则几乎或完全不需要人的干预。

所有这些活动都是让计算机软件为我们做某些事。其中一些活动使用科学符号，一些使用固定格式的自然语言，另一些则使用具体的编程符号。实际上这些活动都是某种形式的编程，其本质可归结如下：

1. 把某个量与另一个量相关联；
2. 用值代换名进行关系计算。

事实上，上述两个概念刻划了使用最低级的程序设计语言，如机器语言，和使用最流行的程序设计语言，如 Java，进行编程的本质。程序将输出和输入相联系，将程序应用于特定的输入，就是在计算中用具体的值代替相关的名字。

没有人可以预知今后 5 年或是 10 年内会出现哪种类型的应用程序包。但是，使用应用程序包仍然需要某种形式的编程。要使学生们掌握编程，学校要么强迫他们学习代数，它是编程的数学基础，要么让他们进行某种形式的程序设计活动。有了现代化的程序设计语言和程序设计环境，选择后者可以更有效地完成任务，还可以使代数学习的过程变得更加有趣。

设计诀窍

烹饪既是孩童的游戏也是成人的乐事，细心烹饪是爱的举措。
——克雷格·卡莱波恩（1920—2000），《纽约时报》饮食版编辑

学习设计程序就像学习踢球一样，必须练习断球、运球、传球和射门。一旦掌握了这些基本技术，下一个目标就是学习担任某个角色、选择并实施合适的战略，如果没有现成的，需要创造一种。

程序员和建筑师、作曲家以及作家一样，是富有创造性的人。他们的念头从白纸开始，先构思概括，再把它写到纸上，直到写出的东西能充分反映他们的思想为止。他们使用图形、文字或其他方法来表达建筑物风格、描述人的特征或是谱写音乐旋律。他们能胜任自己的职业，是因为经过长时间的练习，他们能本能地使用这些技能。

程序设计者也是先形成程序框架，然后翻译为最初的程序版本，再反复修改，直到与最初的想法相符。事实上，好的程序员会多次编辑和修改自己的程序，最终达到某种形式的标准。这和足球运动员、建筑师、作曲家以及作家一样，他们必须长期练习行业必需的基本技能。

设计诀窍类似于控球技巧、写作技巧、乐曲编排技巧和绘图技巧。通过学习和研究，在程序设计领域，计算机科学家已经积累了许多重要的方法和技巧，本书挑选了其中最重要和最实用的一些，由浅入深，逐一讲解[1]。

---

[1] 我们的设计诀窍参考了 Daniel P. Friedman 关于结构递归的工作、Robert Harper 关于类型理论的工作以及 Michael A. Jackson 关于设计工作的方法。

本书大约有一半的设计诀窍涉及输入数据和程序之间的关系。更准确地说，它们描述了如何从输入数据的描述得出整个程序的模板，这种基于数据驱动的程序设计方式最常见，易于创建、理解、扩展和修改。其他设计诀窍有生成递归（generative recursion）、累积(accumulation)和历史敏感性(history sensitivity)。其中，递归型程序可以被重复调用以处理新的问题；带累积器的程序在处理输入的过程中收集数据；历史敏感性程序可以记住程序被多次调用的信息。最后，但不是最不重要的，是抽象程序的设计诀窍。抽象是把两个（或更多）相似的设计概括为一个并由它衍生最初示例。

在许多场合下，往往会由问题联想到设计诀窍。在另外一些场合下，则必须在几种可能性中作出选择，不同的设计决窍可能会导致不同的程序结构，它们之间的差别可能很大。对于一个具有创造性的程序员来说，做出选择是很自然的事情。除非程序员十分熟悉所有可选的设计诀窍，完全理解选择某个诀窍而不是另一个诀窍的后果，否则程序设计过程不可避免按事论事，甚至会导致离奇古怪的结果。我们希望通过制订一系列设计诀窍来帮助程序员理解选择什么以及如何进行选择。

上面解释了“编程”和“程序设计”的含义，读者应该理解到本书所讲授的思想方法和技能对多种职业来说都相当重要。要正确地设计程序，你必须：

1. 分析通常使用文字表述的问题；
2. 在抽象表达问题实质的同时使用例子进行说明；
3. 用精确的语言阐明所表述的语句和注释；
4. 通过检查、测试对上述活动进行评价和修改；
5. 关注细节。

所有这些行为对商人、律师、记者、科学家、工程师以及其他人来说都是有用的。

尽管传统意义上的编程也需要这些技巧，但初学者往往不理解它们之间的关系。问题是，传统的程序设计语言以及传统形式的编程需要学生完成大量的登记工作并记住许多与特定语言相关的细节。简而言之，琐碎杂事淹没了技术本质。要避免这个问题，教师必须使用一种适合初学者的程序设计环境，它尽可能不增加学生额外的负担。在开始编写本书的时候，这样的工具并不存在，因此我们就自行开发了。

选用 Scheme 和 DrScheme

我们把美归于简单，
不含多余部分，
边界清晰，
与一切相关联，
是中庸之道。
——拉尔夫·沃尔多·爱默生，《人生苦旅》

本书选择 Scheme 作为编程语言，辅助程序设计环境为 DrScheme，软件可以免费从本书的正式网站下载[1]。

尽管如此，本书并不是一本介绍 Scheme 程序设计语言的书籍，它仅涉及部分 Scheme 结构。具体来说，本书仅使用 6 种 Scheme 结构（它们是函数定义和调用、条件表达式、结构体定义，局部定义以及赋值等）以及大约 12 个基本函数，它们就是讲授计算和编程原则所需要的全部东西。希望把 Scheme 当作一种工具来使用的人则需要阅读其他的材料。

对初学者来说，选用 Scheme 是很自然的。首先，程序员可以把注意力集中于两个要素，即前面所指出的基本编程原则：程序就是数量之间的关系，对于特定的输入求取结果。使用 Scheme 语言核心，在教师的指导下，学生在第一堂课就可以开发出完整的程序。

[1] Scheme 有一个正式的定义，即由 Richard Kelsey、Willism Clinger、Jonathan Rees 和许多 Scheme 实现者编辑的“Scheme 修改报告”。要了解该报告以及 Schme 的不同实现，请访问 www.schemers.org。请注意，本书对该报告进行了扩充并针对初学者进行了剪裁。

其次，可以方便地将 Scheme 组织成从简单到复杂的一系列不同级别的语言。这个性质对初学者来说是至关重要的。当初学者犯了简单的符号错误时，一般程序设计语言会给出含糊的、与语言高级特征相关的错误消息。初学者往往浪费很多时间来查找错误所在，由此产生学习上的挫折感。为了避免这些问题，通过持续对赖斯大学计算机实验室的程序设计初学者的观察，经过谨慎选择，DrScheme 实现了若干不同层次的 Scheme 程序设计环境。按照安排，不同的环境会给出与学生当前知识水平相适应的错误消息。更好的是，分层会避免许多基本错误。当学生学习了足够的编程和语言知识后，教师可以建议他们接触更丰富的语言层次，由此编写更有趣、更简练的程序。

另外，DrScheme 提供了一个真正的交互式环境。环境由两个窗口组成：一个是 Definitions 窗口，在其中可以定义程序，另一个是交互窗口，其行为就像是一个袖珍计算器，你可以在其中输入表达式，由 DrScheme 求出它们的值。换句话说，计算由袖珍计算器完成，而这是学生们都相当熟悉的。很快，计算形式就从袖珍计算器上的算术运算向前推进，变成对结构体、表和树的计算。使用交互式的计算方式甚至可以鼓励学生们用各种方法进行程序实验，从而激发他们的好奇心。

最后，使用包含丰富数据结构的交互式程序设计环境可以让学生把注意力集中于问题的解决和程序设计活动之上。关键的改进是，交互式求值环境避免了（几乎是）多余的关于输入和输出的讨论。这一点改进带来了几种结果。第一，掌握输入和输出函数需要记忆，学习这些东西单调乏味、令人厌烦，相反，如果使用固定方式的输入和输出，我们就能将精力集中于问题求解技术的学习；第二，良好的面向文字的输入需要深奥的编程技能，最好从问题求解的课程中学习（学生应该从更高级的课程中学习）。教那些糟糕的面向文字的输入，是对老师和学生时间的浪费；第三，现代软件一般采用图形用户界面（GUI），GUI 是程序员使用编辑器和“向导”设计的，不是手工完成的。学生最好学习使用与标尺、按钮、文本框等相关的函数，而不是背诵那些与流行的 GUI 库相关的特定协议。简而言之，在初次介绍编程时就讨论输入和输出是对宝贵的学习时间的浪费。如果要进一步学习，掌握必需的 Scheme 输入输出知识也比较方便。

总而言之，只要少量几节课，学生们就可以学会 Scheme 语言的核心，这种语言和传统的程序设计语言一样强大。这样，学生立即就可以把注意力集中于编程本质，这将极大增强他们解决一般问题的能力。

本书正文部分

本书由 8 个部分和 7 个独立的章节（书中第 8、13、18、24、29、33、38 章）组成。8 个部分主要讨论程序设计，独立章节则介绍一些与程序设计和计算相关的话题。图 2 给出了本书各部分之间的依赖关系，可以看出，你可以按不同的顺序来阅读本书，只阅读部分内容也是可以的。

本书第一至第三部分包括了基于数据驱动的程序设计基础。第四部分介绍了程序设计中的抽象问题。第五部分和第六部分与递归及累积相关。本书前 6 部分使用了纯函数式（或称代数式）的程序设计风格，即无论计算多少遍，同一个表达式每次计算的结果总是相同。这种特性使程序易于设计，程序性质易于推导。不过，为了处理程序之间的接口和解决其他领域的问题，我们放弃了部分代数性质，引入了赋值语句。本书的最后两部分说明了设计程序的意义，更精确地说，它们阐述了如何应用前 6 个部分所描述的程序设计诀窍，以及使用赋值语句必须特别小心的一些问题。

独立章节则介绍一般性的、非本质的，对计算和程序本身来说是重要的话题，但并不涉及程序设计有的是在严格的基础上介绍本书所选定的 Scheme 子集的语法和语义，有的是介绍了另外的程序设计结构。独立章节 5（第 29 章）讨论了抽象的计算开销（包括时间和空间开销），并介绍了向量的概念。独立章节 6（第 33 章）则比较了两种数值表示技术以及处理它们的方法。

只有某些独立章节的内容的学习可以推后，直到需要时再学习。对于学习与 Scheme 语法和语义相关的独立章节尤其应该注意。但是，考虑到图 2 中第 18 章的重要地位，我们应该及时学习。

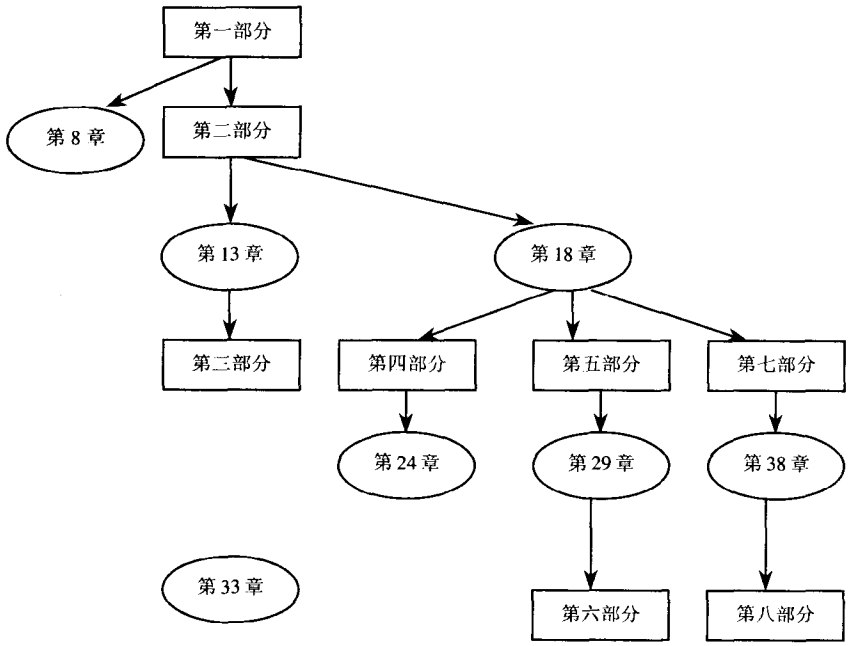


图 2 本书各部分和独立章节之间的依赖关系

程序的逐步求精：系统化程序设计方法对于开发大型项目特别有意义，也特别重要。而开发单一函数到小规模的包括多个函数的项目则需要另外一种设计思想：逐步求精，即先设计程序核心，再增加功能，直至满足整个需求目标为止。

学习完第一节课后同学就应该有了程序逐步求精的初步印象。为了使学生熟悉这种技术，书中给出了许多补充练习，这些使用简短概述引出的练习可以指导学生进行程序逐步求精的训练。第 16 章将明确阐述这种思想。

另外，本书会重复使用某些练习和例子。例如，第 6.6 节、第 7.4 节、第 10.3 节、第 21.4 节、第 41.4 节以及最后 2 节中的一些练习题都涉及了如何在画布上移动图片的问题。这样，学生就会多次看到同样的问题，而每一次讨论都会增加他们对程序组织的了解。

本书通过逐步把功能加到一个程序体中的方法来示范为什么程序员必须遵循设计诀窍，借助问题的解决方式向学生展示如何在可用的设计诀窍中进行选择。有时候新知识的作用只是帮助学生改进程序的组织结构，换句话说，要让学生了解到在他们初步的工作完成之后，编程过程并没有结束，如撰写论文和书籍一样，程序也需要进一步的编辑和修改。

教学软件包（TeachPack）：完成工程项目的一个要求是程序员必须进行团队合作。在程序设计教学环境下，这意味着一个学生写的程序必须与另一个学生编写的程序相匹配。为了模仿“与另一个程序相配”这一概念，本书提供了 DrScheme 教学软件包。粗略地说，教学软件包模仿了一个合作者，由此可以避免由于合作者程序存在错误而带来的不便。技术性的说法是，工程总是由视图和程序组件模型两个部分组成（在模型—视图软件体系结构意义上），在典型的环境下，学生设计模型，教学包则提供视图。通常，教学包以（图形）用户界面的形式提供视图，而不是单调乏味、毫无意义的代码。事实上，这种分离模仿的是真实世界中的工程分工。

为了使模型与视图相符，学生必须注意函数的规范，必须遵循设计诀窍。对程序员来说，使模型与视图相符是一种非常重要的技能，但程序设计的初级课程常常未能给予足够的重视。在第四部分，为了说明创建 GUI 的过程并不神秘，我们将说明如何建立一些简单的 GUI，以及 GUI 事件是如何触发函数调用的，但我们不会把很多时间花费在一个需要死记硬背，只需很少思考的主题上。

进度：根据需要，每个学校都可以有自己的教学进度表。在赖斯大学，讲授整本教材以及其他一些附加材料通常需要一个学期的时间。一个研究性大学的教师可以使用类似的进度。而高中教师就必须放

慢进度。许多尝试使用本教材的高中教师在一个学期内完成了前三个部分的教学；而少数高中只使用本书第一部分，从计算的角度讲授代数问题的求解；另一些高中则用一年的时间教完整本书。要得到有关教学进度表的更多信息，请访问本书的网站。

本书站点：本书有两种版本，除了纸介版本，在网站

http://www.htdp.org/

可以免费获得电子版本。

网站提供了一些附加材料，包括前面提到过的各种类型的补充练习。目前网站提供有可视化的小球游戏模拟，更多的练习将在不久的将来加入网站。

致谢

我们特别感谢四个人：罗伯特·科克·卡特赖特，他与本书的第一个作者合作开发了赖斯大学此入门性课程的前身；丹尼尔·P·弗里德曼，他在 1984 年要求本书的第一个作者重写了 *The Little LISPer*（由麻省理工学院出版社出版），而这也是本书写作计划的开始；约翰·克莱门特，他负责设计、实现和维护 DrScheme 软件；还有保罗·斯特克勒，他忠实地支持我们，帮助我们开发所需的程序设计工具组件。

有许多友人和同事帮助我们开发这本教材，他们在自己的课堂上使用本教材，并且对本书的初稿给出具体的评论。我们对他们的帮助和耐心表示感谢，这些人包括伊恩·巴兰德，约翰·克莱门特，布鲁斯·迪拜，迈克·厄恩斯特，凯瑟·菲尔勒，丹尼尔·P·弗里德曼，约翰·格林纳，约翰·斯通，杰拉尔丁·莫林和瓦尔迪马·忒门。

在赖斯大学，本书草稿在课程 Comp 210 上使用了 12 次，学生们提出了许多宝贵意见。众多 TeachScheme！研讨会的参加者在他们的课堂上使用了本书的最初草稿，他们中的许多人提出了评论和建议。作为其中代表，这里列出一些作出过积极贡献的人，他们是：巴巴拉·阿德勒女士，斯蒂芬·布洛赫博士，杰克·克莱先生，理查德·克莱门斯博士，凯尔·吉列先生，克伦·布拉斯女士，马文·赫男得先生，迈克尔·亨特先生，克伦·诺斯女士，贾明·雷蒙德先生以及罗伯特·里德。克里斯托弗·费雷荪和他的父亲耐心地参与了本书前几个部分的工作，让我们直接了解到了年轻学生的观点。感谢他们中的每一位。

最后，Matthias 在这里表达他对海尔格的感激，感谢她多年以来的耐心，感谢她为一个心不在焉的丈夫和父亲建立了一个家庭。Robby 要感谢黄馨慧的支持和鼓励，没有她，他不可能完成任何事。Matthew 感谢顾文沅给予的忠诚的支持和不停的争论。Shriram 感激凯瑟·菲尔勒的支持、忍耐和俏皮语，同时对她参与本书部分工作表示感谢。

目 录

第一部分 简单数据的处理

第 1 章 学生、教师和计算机 …… 3

第 2 章 数、表达式和简单程序 …… 5

2.1 数和算术运算 …… 5
2.2 变量和程序 …… 6
2.3 字处理问题 …… 9
2.4 错误 …… 10
2.5 设计程序 …… 12

第 3 章 程序就是函数加上变量定义 …… 15

3.1 函数复合 …… 15
3.2 变量定义 …… 17
3.3 函数复合练习 …… 18

第 4 章 条件表达式和函数 …… 20

4.1 布尔类型和关系 …… 20
4.2 函数和条件测试 …… 22
4.3 条件和条件函数 …… 25
4.4 条件函数的设计 …… 27

第 5 章 符号信息 …… 31

第 6 章 复合数据之一：结构体 …… 34

6.1 结构体 …… 34
6.2 补充练习：绘制简单图形 …… 36
6.3 结构体定义 …… 38
6.4 数据定义 …… 41
6.5 设计处理复合数据的函数 …… 43
6.6 补充练习：圆和长方形的移动 …… 46
6.7 补充练习：刽子手游戏 …… 49

第 7 章 数据的多样性 …… 52

7.1 数据混合与区分 ……52
7.2 设计处理混合数据的函数 ……55
7.3 再论函数复合 ……58
7.4 补充练习：图形的移动 ……60
7.5 输入错误 ……61

第 8 章 语法和语义 ……63

8.1 Scheme 的词汇 ……63
8.2 Scheme 的文法 ……64
8.3 Scheme 的含义 ……65
8.4 错误 ……68
8.5 布尔值表达式 ……70
8.6 变量定义 ……71
8.7 结构体的定义 ……72

第二部分 任意数目数据的处理

第 9 章 复合数据类型之二：表 ……77

9.1 表 ……77
9.2 任意长的表的数据定义 ……80
9.3 处理任意长的表 ……82
9.4 设计自引用数据定义的函数 ……84
9.5 更多关于简单表的例子 ……86

第 10 章 表的进一步处理 ……90

10.1 返回表的函数 ……90
10.2 包含结构体的表 ……93
10.3 补充练习：移动图片 ……98

第 11 章 自然数 …… 100

11.1 定义自然数 …… 100
11.2 处理任意大的自然数 …… 101
11.3 补充练习：创建表，测试函数 …… 103
11.4 自然数的另一种数据定义 …… 104
11.5 更多与自然数有关的性质 …… 108

第 12 章 三论函数复合 …… 110

12.1 设计复杂的程序 …… 110
12.2 递归的辅助函数 …… 111
12.3 问题泛化与函数泛化 …… 114

12.4　补充练习：字母的重新排列 …… 117

第 13 章　用 list 构造表 …… 119

第三部分　再论任意大数据的处理

第 14 章　再论自引用数据定义 …… 125

14.1　结构体中的结构体 …… 125
14.2　补充练习：二叉搜索树 …… 131
14.3　表中的表 …… 135
14.4　补充练习：Scheme 求值 …… 137

第 15 章　相互引用的数据定义 …… 139

15.1　由结构体组成的表与结构体中的表 …… 139
15.2　为相互引用的定义设计函数 …… 144
15.3　补充练习：网页再谈 …… 145

第 16 章　反复精化设计 …… 147

16.1　数据分析 …… 147
16.2　定义数据类型，再改进它们 …… 148
16.3　改进函数和程序 …… 150

第 17 章　处理两种复杂数据片段 …… 152

17.1　同时处理两个表：第一种情况 …… 152
17.2　同时处理两个表：第二种情况 …… 154
17.3　同时处理两个表：第三种情况 …… 156
17.4　函数的简化 …… 159
17.5　设计读入两个复杂输入的函数 …… 160
17.6　处理两个复杂输入的练习 …… 161
17.7　补充练习：Scheme 求值之二 …… 164
17.8　相等与测试 …… 165

第 18 章　局部定义和辖域 …… 172

18.1　用 local 组织程序 …… 172
18.2　辖域和块结构 …… 183

第四部分　抽 象 设 计

第 19 章　定义的相似性 …… 189

19.1　函数的类似之处 …… 189
19.2　数据定义的类似之处 …… 195

第 20 章 函数也是值 …… 199

20.1 语法和语义 …… 199
20.2 抽象函数和多态函数的合约 …… 200

第 21 章 抽象设计的例子 …… 204

21.1 从实例中抽象 …… 204
21.2 抽象表处理函数的练习 …… 208
21.3 抽象与惟一控制点 …… 209
21.4 补充练习：再论图片移动 …… 210
21.5 注意：由模板设计抽象 …… 211

第 22 章 使用函数进行抽象设计 …… 213

22.1 返回函数的函数 …… 213
22.2 把函数当成值来进行抽象设计 …… 214
22.3 图形用户界面初探 …… 216

第 23 章 数学方面的例子 …… 223

23.1 数列和级数 …… 223
23.2 等差数列和等差级数 …… 225
23.3 等比数列和等比级数 …… 225
23.4 函数曲线下方的面积 …… 228
23.5 函数的斜率 …… 229

第 24 章 定义匿名函数 …… 234

24.1 lambda 表达式的语法 …… 234
24.2 lambda 表达式的辖域和语义 …… 235
24.3 lambda 表达式的语用 …… 237

第五部分 生 成 递 归

第 25 章 一种新的递归形式 …… 241

25.1 为桌上的一个球建立模型 …… 242
25.2 快速排序 …… 244

第 26 章 设计算法 …… 248

26.1 终止 …… 249
26.2 结构递归与生成递归的比较 …… 251
26.3 做出选择 …… 252

第 27 章 主题的变更 …… 256

27.1 分形 …… 256
27.2 从文件到行，从表到表的表 …… 260
27.3 二分查找 …… 263
27.4 牛顿法 …… 267
27.5 补充练习：高斯消去法 …… 269

第 28 章 回溯算法 …… 273

28.1 图的遍历 …… 273
28.2 补充练习：皇后之间的相互攻击 …… 277

第 29 章 计算的代价和向量 …… 280

29.1 具体的时间和抽象的时间 …… 280
29.2 “阶”的定义 …… 284
29.3 向量初探 …… 286

第六部分 知 识 累 积

第 30 章 知识的丢失 …… 297

30.1 一个与结构处理相关的问题 …… 297
30.2 一个关于生成递归的问题 …… 300

第 31 章 设计带累积器的函数 …… 304

31.1 认识累积器的必要性 …… 304
31.2 带累积器的函数 …… 305
31.3 把函数转换成带累积器的变体 …… 306

第 32 章 使用累积器的更多例子 …… 315

32.1 补充练习：有关树的累积器 …… 315
32.2 补充练习：传教士和食人者问题 …… 319
32.3 补充练习：单人跳棋 …… 321

第 33 章 非精确数的本质 …… 323

33.1 固定长度的数的算术运算 …… 323
33.2 上溢出 …… 327
33.3 下溢出 …… 328
33.4 DrScheme 数 …… 328

第七部分 改变变量的状态

第 34 章 函数的记忆 …… 333