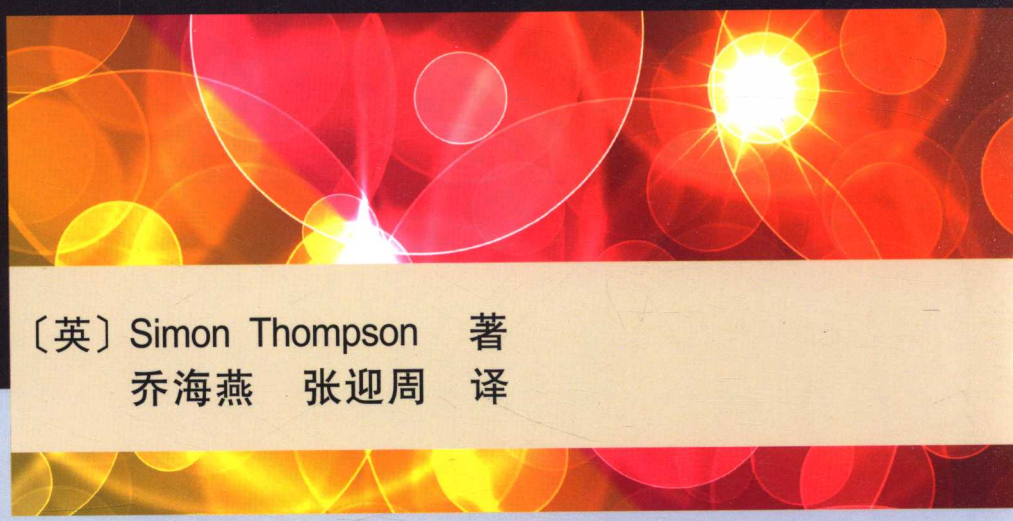
PEARSON

国外信息科学与技术优秀图书系列　计算机科学与技术

Haskell 函数式编程基础（原书第3版）

Haskell: The Craft of Functional Programming (Third Edition)



〔英〕Simon Thompson　著
乔海燕　张迎周　译

科学出版社

国外信息科学与技术优秀图书系列

Haskell 函数式编程基础

(原书第 3 版)

Haskell: The Craft of Functional Programming

(Third Edition)

〔英〕Simon Thompson 著

乔海燕 张迎周 译

科学出版社

北京

图字：01-2013-1143 号

内 容 简 介

本书是一本非常优秀的 Haskell 函数式程序设计的入门书，依次介绍函数式程序设计的基本概念、编译器和解释器、函数的各种定义方式、简单程序的构造、多态和高阶函数、数组和列表的结构化数据、列表上的原始递归和推理、输入输出 I/O 的控制处理、类型检测方法、代数数据类型、抽象数据类型、惰性计算等内容。书中包含大量的实例和习题，注重程序测试、程序证明和问题求解，易读易学。全书循序渐进，从基本的函数式程序设计直至高级专题，让读者对 Haskell 的学习不断深入。

本书可作为计算机科学和其他相关学科的高年级本科生、研究生的教材，也可供对函数式程序设计感兴趣的程序员、软件工程师等参考学习。

图书在版编目(CIP)数据

Haskell 函数式编程基础：原书第 3 版/(英)汤普森(Thompson, S.)著；乔海燕，张迎周译. —北京：科学出版社，2015.4
(国外信息科学与技术优秀图书系列)
书名原文：Haskell: The Craft of Functional Programming: Third Edition
ISBN 978-7-03-044168-3
Ⅰ.①H… Ⅱ.①汤… ②乔… ③张… Ⅲ.①函数-程序设计 Ⅳ.①TP311.1

中国版本图书馆 CIP 数据核字(2015) 第 083414 号

责任编辑：任 静 李 静／责任校对：郭瑞芝
责任印制：张 倩／封面设计：迷底书装

科学出版社 出版
北京东黄城根北街 16 号
邮政编码：100717
http://www.sciencep.com

北京市文林印务有限公司印刷
科学出版社发行 各地新华书店经销
*
2015 年 6 月第 一 版 开本：787×1092 1/16
2015 年 6 月第一次印刷 印张：34 1/2
字数：804 000
定价：150.00 元
(如有印装质量问题，我社负责调换)

前　言

计算机技术的更新可谓日新月异，然而其基本理论却几乎保持不变。现代计算机的尺寸和性能与其诞生之初有着巨大的差别，但是它的基本结构自其诞生以来几乎没有变化。在程序设计领域，诸如面向对象的现代程序设计思想从研究领域走向商业主流经历了几十年。

从这个意义上讲，像 Haskell 这样的函数式程序设计语言是相对年轻的，但是随着多核芯片成为所有计算设备的标准配置，Haskell 的影响力将与日俱增。那么，学习 Haskell 这类函数式程序设计语言的意义是什么呢？

- 人们每天实际使用的许多安全无误系统是用函数语言构建的。例如，Linux 的窗口管理系统 Xmonad（xmonad.org）是完全用 Haskell 编写的；在 C 和 VHDL 中设计安全密码系统的 Cryptol 语言（www.cryptol.net）本身就是用 Haskell 实现的。
- 函数语言是一种通用程序设计语言，但是也为设计适用于特定领域的专用语言（DSL）提供了理想的工具，如硬件设计和财务建模等。
- 函数式程序设计语言为人们提供了一个*实验室*，现代程序设计的核心思想已经在其中体现得淋漓尽致。为此，函数式程序设计语言已经成为流行的计算机科学教学语言，并且影响着其他语言的设计。例如，Java 中泛化（generics）的设计便直接建立在 Haskell 多态模式上。
- 函数语言还可帮助读者从不同于 Java、C# 和 C++ 等语言的角度理解程序设计。即使读者今后不用 Haskell 编写任何大的程序，学习 Haskell 仍然可以使读者成为一个更好的 Java、C# 或 C++ 程序员，因为读者在编写代码时可以看到更大的代码可选择空间。
- 最后，读者可能发现自己已经在使用一种函数语言，即便这种语言不是 Haskell。微软推出的函数语言 F# 是 Visual Studio 的标准配置，并且在许多领域特别是在财经领域越来越受欢迎。Erlang 是一种商用并行函数语言，并且已经为许多基于网页的系统提供可扩展架构，包括 Facebook 聊天系统和 Amazon 及 Yahoo 的许多服务。

本书主要介绍 Haskell 函数式程序设计，首先简单解释函数式程序设计、Haskell 和 GHCi，然后介绍本书适合的读者。为了说明本书与其他相关书籍的区别，作者列出本书的特色以及各章的内容概述。然后解释本书与前面版本的不同，如何阅读本书，最后列出本书包含了哪些实例。

什么是函数式程序设计

函数式程序设计为用户提供了一个高层次的程序设计视野，由此带来的各种特点可以帮助用户建立优美、强大和通用的函数库。函数式程序设计的核心概念是函数。一个函数的功能是：对于任意输入，计算相应的输出结果。

体现函数式程序语言的表现力与通用性的一个例子是map函数，它通过某种特定的方式作用于列表的每个元素。例如，map可用于将一列数中的每个数加倍，或者将一个图形列表中的每个图形反色。

函数式程序设计的简洁性表现在函数定义的方式：一个等式可用于说明相应输入下的函数返回结果。例如，下面定义的函数 addDouble将两个整数相加，然后将其和加倍：

```
addDouble x y = 2*(x+y)
```

其中 x和 y为输入，2*(x+y) 为结果。

函数式程序设计的模型简单利落。例如，欲计算下列表达式的值

```
3 + addDouble 4 5
```

只需使用表达式所含的函数定义的等式，即

```
 3 + addDouble 4 5
 ⇝ 3 + 2*(4+5)
 ⇝ 3 + 2*9
 ⇝ 3 + 18
 ⇝ 21
```

这便是计算机如何求一个表达式值的过程，完全等同于使用纸和笔计算表达式值的过程，由此使得函数计算过程的实现具有透明性。

此外，我们还可以简单介绍程序的一般特性。对于上例addDouble，利用x+y 和y+x相等的性质，可以断言，对于所有的 x和 y，addDouble x y 和addDouble y x 相等。这样的证明思想在传统的命令式语言和面向对象（OO）语言中较难实现，但是这种证明在函数式程序语言中的实现要容易得多。

Haskell 与 GHCi

本书使用程序设计语言 Haskell。对于多数计算机系统，Haskell 提供了免费的编译器和解释器。本书使用 GHCi 解释器，它为初学者提供了理想的平台，并且具有编写周期快、界面简洁的特点。GHCi 提供 Windows、UNIX 和 Macintosh 免费版本。

Haskell 的设计始于 20 世纪 80 年代后期，其初期目标是作为惰性函数式程序设计的标

准语言，并历经多次变化与修改。本书使用 Haskell 2010 编写，因为本书写作时 Haskell 2010 是 Haskell 的最新标准。Haskell 标准每年都会评估，但在未来新的版本标准中本书讨论的部分内容很可能会保持不变。

虽然本书涵盖了 Haskell 2010 的大部分特性，但也仅是一本程序设计教程，而不是语言参考手册。有关语言和函数库的细节以及 Haskell 的其他丰富资源参见 Haskell 主页`http://www.haskell.org`。

谁应该阅读本书

本书可作为计算机科学和其他相关学科的本科生学习函数式程序设计的基本教材。本书也可供计算机科学的初学者，或者具备一定计算机科学知识、初次学习函数式程序设计的学生使用，这两类学生均会发现本书内容既新颖又富有挑战性。

此外，本书也可供程序员、软件工程师和其他想学习函数式程序设计的人员自学。

本书自成体系，但是读者在使用 Haskell 的解释器时需要具备有关系统命令和文件操作等基本知识。本书也引入了一些逻辑记号，并且在引入时做了解释。另外，如果读者了解 $\log x$、n^2 和 2^n 的函数图像，这对于阅读第 20 章将会有帮助。

本书有什么特色？

介绍程序设计的教材总是有许多共同点，但是每种教材都各有特色。下面是本书的特色。

- Haskell 拥有一个很大的预定义函数库，尤其是列表上的一些函数，本书对此进行深入探讨，并且鼓励读者在看到函数的定义之前使用这些函数。这种方式可使读者学习进步更快，而且符合实际工作的习惯：大多数程序建立在预定义的大型函数库上，因此，从开始便采用这样的方式对于读者是非常有益的体验。
- 本书一开始便介绍了基于性质的测试工具QuickCheck。实践证明，QuickCheck 是一种轻量级又很有效地改善程序质量的工具，本书的例子也说明了这一点。
- 以列表函数推理为引子，本书将详细讨论函数程序的推理。推理的重点对象是自然数上的函数和列表函数，其原因如下：证明函数在列表上的性质似乎更实际；而且列表上的结构归纳原理更容易被读者接受。本书一开始便对基于性质的测试和推理做了对比。
- 图形一例在第 1 章引入，并贯穿全书。因此，读者可以看到同一问题的不同编程方式，并且有机会对不同的设计进行比较和思考。为了便于显示图形，本书也为该图形接口提供了基于浏览器的实现。

- 强调了 Haskell 是一种实用程序设计语言，介绍了模块的概念、I/O 的 do 表示法。其他单子（monadic）程序在后面的章节介绍。
- 类型在本书中具有重要的作用。在定义每个函数或者对象的同时，其类型也一并引入。类型的引入不仅便于检查一个定义具有作者期望的类型，而且我们把类型视为一个定义文档的重要组成部分，因为一个函数的类型准确地描述了如何使用这个函数。
- 分阶段介绍一些实例研究。例如，石头–剪子–布游戏、交互式计算器程序、正则表达式的处理、Huffman 编码及译码系统、一个小队列模拟程序包等。通过这些例子来介绍新的概念，并且说明现有的技术如何协同工作。第 viii 页列出每个实例的内容。
- 专设一章讨论使用 Haskell 嵌入领域专用语言，并给出一些基于单子（monad）和组合子（combinator）的领域专用语言。
- 最后一章包括 Haskell 的相关支撑和大量的网站链接。附录列出了其他的有用信息，如可用实现的细节、常见的 GHCi 错误、函数式程序设计、命令式和面向对象程序设计之间的比较。
- 其他的支持材料可以参阅网页`www.haskellcraft.com`和`www.pearsoned.co.uk/thompson`。

内容概要

第 1 章简要介绍函数式程序设计的基本概念：函数与类型、表达式与求值、定义、证明和基于性质测试的 QuickCheck。通过由字符构成的图形实例对高阶函数和多态等一些更深入的概念做了简单介绍。在该图形实例的第二种实现中讨论了领域专用语言，这也是第 19 章的主题。该图形实例是贯穿全书的众多实例之一，也是多次用于介绍新概念的例子。

第 2 章介绍 Haskell 的解释器 GHCi，它是格拉斯哥 Haskell 编译器（Glasgow Haskell Compliler）的交互版本。GHCi 是 Haskell 平台的组成部分，该平台也在这里介绍。本书接着介绍模块系统的基本知识、标准引导库和其他 Haskell 库，最后使用`Picture`类型的 SVG 实现进行练习。本章与第 1 章共同构成一个 Haskell 函数式程序设计课程的基础。

第 3 章阐述如何在整数、字符和布尔值上构造简单程序。自第 3 章开始，我们将通过习题复习相关内容。在此基础上，第 4 章进一步讨论定义函数的各种方式，如辅助函数、局部定义和递归等。第 4 章还以枚举类型的形式介绍最简单的数据类型。这些类型用于讨论石头–剪刀–布游戏，该例也在本书后面的章节讨论。

第 5 章介绍数组和列表形式的结构化数据。代数类型用于表示乘积类型、和类型，也是表示记录和可变长记录的另一种术语。在引进列表概念之后，介绍如何应用两种资源设计列表上的函数：一种是列表概括，它可以有效地展示函数`map`和`filter`的表达力；另一种是使

用一阶引导库函数和其他 Haskell 库函数。

引导库中几乎所有的列表函数都是多态函数，为此第 6 章将探讨多态的概念，介绍标准引导库的列表函数，并将其应用于各种扩展的例子中。第 7 章介绍列表上的原始递归，以及一个更复杂的文本处理的例子。

第 8 章介绍 Haskell 如何处理简单的控制台I/O：为此引入 Haskell 的扩展语法 do 记法，用于表达类型为（IO a）的程序，其有关的原理将在第 18 章介绍。作为实例，石头–剪刀–布游戏的交互版本将用于演示I/O程序设计。

第 9 章引入列表处理程序的推理，其中包括一系列相关逻辑知识的内容。该章介绍构造归纳证明的指导性原则，并且专门用一小节来介绍如何在失败的尝试中构造成功的证明。

第 10 章和第 11 章介绍高阶函数。第 10 章首先讨论函数参数，并且证明函数参数能够实现列表上的许多计算模式。第 11 章将介绍函数作为参数，这样的函数既可以定义为 λ 表达式，也可以定义为部分应用的函数。这些思想在第 12 章通过一些实例得到进一步的展示，包括贯穿本书的实例图形和石头–剪刀–布游戏，以及新的实例 —— 正则表达式的处理和索引表的建立。

类族（类型分类）允许函数重载，即函数在不同的类型上其实现方式也不同。第 13 章包括类族和 Haskell 的各种内置分类，并探讨 Haskell 的类型检测方法。一般地讲，类型检测的任务是解决函数的定义置于其类型上的各种约束。

诸如树这样的代数类型是第 14 章的主题，该章涵盖代数类型从设计和证明到它们与类族的交互作用等各个方面，并且介绍很多来自实践中的代数类型例子。这些例子在第 15 章中将得到进一步的讨论，包括使用 Huffman 编码对信息进行编码和译码的实例。实例研究首先介绍此方法的基础，然后讨论程序的实现。该系统被分解为容易处理的多个模块，为此介绍了 Haskell 模块系统的高级特性。

抽象数据类型（ADT）通过一组函数访问一个实现。第 16 章介绍 Haskell 的 ADT 机制，并且通过大量的例子说明如何使用 ADT，如队列、集合、关系等，此外给出了一个仿真例子的基本函数。

第 17 章介绍 Haskell 的惰性计算。惰性计算为程序员提供了一种处理回溯和无穷数据结构的独特方法。作为回溯的例子，我们研究语法分析，并且使用无穷列表编写进行式程序和一个随机数生成器。

Haskell 程序通过 IO 类型实现输入和输出，这在第 8 章已有介绍。第 18 章将继续介绍 IO 类型，并展示一些大型例子，包括一个交互式前端计算器。do 表示法的基础是单子。单子还可用于不同类型的基于动作的程序设计中，该章的后半部分还将对此作进一步介绍。

领域专用语言（DSL）是 Haskell 应用非常有效的领域，第 19 章讨论什么是 DSL，如何将 DSL 嵌入到像 Haskell 这样的语言中，并介绍浅嵌入和深嵌入的区别。构建在先前实例上的领域专用语言，以及新的实例，如 QuickCheck的生成语言，这些例子可以很好地诠释单子

模式的领域专用语言。

第 20 章将讨论程序的性能，即程序计算其结果所需的时间和空间。该章也介绍度量 Haskell 程序运行性能的基本知识。本书的最后一章即第 21 章，包括各种相关应用的概述、Hasekll 的扩展和进一步的研究方向，其中还提供了许多的网页地址和参考资料。

附录涵盖各种背景资料。附录 A 提供了有关函数式程序设计和 OO 程序设计的链接；附录 B 是函数式程序设计常用词汇表；其他附录包括 Haskell 运算符和 GHCi 错误汇集，以及有助于理解 Haskell 程序和 Haskell 的各种实现的信息。最后一个附录是大规模 Haskell 项目的建议。

本书所有例子的 Haskell 代码以及其他背景资料均可以从`www.haskellcraft.com`下载。

第 2 版到第 3 版的变化

第 3 版的变化覆盖范围广。每一章都增加了新的内容，特别是新的例子，在读者反馈的基础上，内容编排顺序也有变化。以下是具体的变化内容：

- QuickCheck贯穿全书并用于测试 Haskell 程序。本书开篇便介绍了性质的定义，并用 QuickCheck测试这些性质，而且说明了性质本身也可能定义错误。本书也介绍了 HUnit，但是应用不多。
- 证明贯穿全书，QuickCheck作为证明的补充：QuickCheck给一个性质提供证据，证明性质的正确性。本书代码库提供了用户定义的生成器，读者在第 19 章学习如何在定义生成器之前测试自己的代码。
- 新增一些例子，有的例子出现一次，有的例子在多章节中出现。这些例子包括石头–剪刀–布游戏、扑克游戏、图形的 SVG 显示和正则表达式。在石头–剪刀–布游戏和正则表达式例子中特别强调了**函数作为数据**。
- Haskell 应用很成功的一个领域是它给**领域专用语言**（DSL）提供基础，这是第 19 章讨论的内容。该章首先介绍开发 DSL 的原因，然后通过图形和正则表达式例子讨论浅嵌入和深嵌入的区别。最后通过图形专用语言的命名和 QuickCheck的生成器例子介绍了单子式的 DSL。
- 内容顺序有变化，有些内容提前了[①]。数据类型的介绍提前了许多，其中枚举类型放在 4.5 节，非递归类型放在 5.3 节。IO 交互程序在第 8 章介绍以支持石头–剪刀–布游戏程序：这种处理只介绍编写 IO 程序的 do 语法，其深层的机制留在第 18 章介绍。最后，局部定义首先出现在 4.2 节。

①有趣的是要求将内容提前的人数是要求将内容推后人数的 10 倍多。或许理想的书是在第 1 章引入所有概念和内容，然后在余下的 20 章讨论这些内容或者展开内容。

- 第 2 版主要使用 Hugs 解释器，本版使用 GHCi，这个交互解释器是 Haskell 平台的组成部分。在介绍 GHCi 的同时，详细介绍了如何通过使用 Hackage、Cabal、Hoogle 和 Hayoo 最大限度使用 Haskell 库和包。
- 在第 2 版图形的“ASCII 艺术”实现上增加了基于现代浏览器的 SVG/HTML5 功能的实现。
- 第 20 章关于性能的讨论增加了如何在 GHC 中度量实际 Haskell 程序性能的内容。
- 作为一个附录，添加了一组项目建议。更多的支持材料可以从`www.haskellcraft.com`得到。

第 1 版到第 2 版的变化

这些变化在第 2 版的序言中有详细说明，这里概述如下。

- 在列表上定义函数的方法。为了避免学生在定义每个列表函数时使用递归，首先介绍了列表概括和列表库函数，其后再介绍递归。这种变化是否有效仍无定论。
- 图形一例是贯穿全书的实例，这个例子给出程序视觉上的表示，同时解释了一般 Haskell 列表程序中高阶函数和多态函数的作用。
- 本版与 Haskell98 一致，特别是使用了标准函数的标准命名。本版也介绍了 Hugs 解释器，它是本书选择的一个解释器实现。
- 对于 I/O 程序和单子程序使用了 do 语法，而不是函数记法>>=和 return。
- 更系统地介绍 Haskell 类型和类型检测机制。
- 使用程序设计解决问题的方法，内容大致基于波利亚（Polyá）的工作。

如何阅读本书

本书是一个有机整体。新内容逐渐引入，并用新的例子和贯穿全书的例子加以说明。有些内容脱离主要内容，读者可以跳过这些内容以构建一个较短的课程。

程序证明　本书强调程序证明，始于第 9 章，后续内容包含在 11.6 节、14.7 节、16.10 节和 17.9 节中。

程序性能　第 20 章是程序时间和空间性能自成一体的内容。实例的研究用于给出大程序而不是单个函数的例子，也用于展示特定的思想和结构。

图形　用于说明列表在建模中的作用，以及高阶函数和多态函数的价值，如`zipWith`、`map`和(++)。图形在第 19 章也用于演示领域专用语言的浅嵌入和深嵌入，以及命名的图形说明单子式 DSL 的例子。

石头–剪刀–布　在这个例子中，读者在 4.3 节看到 Move 作为枚举的第一个例子，在 8.1 节看到函数作为数据的策略。该例也在第 8 章提供了 I/O 交互的例子。

计算器　计算器的例子始于 14.2 节引入的描述数值表达式的代数数据类型 Expr。第 16 章引入描述变量值的抽象数据类型 Store。第 17.5 节说明如何将文本经语法分析得到数值表达式，最后在 18.3 节给出交互式循环读入–计算–输出的计算器。

图书馆数据库　5.7 节引入图书馆数据库时介绍了如何使用列表概括构建列表上的程序，而不显式地使用递归。

超市账单　该例出现在 6.7 节，又一次显示了列表库函数的作用，这里也没有使用递归。

文本处理　7.6 节的文本处理一例与以上两例不同，这里必须显式地使用列表上的递归。

正则表达式　正则表达式是作为函数也是数据的例子引入的，然后在第 19 章介绍深嵌入时再次出现，以区别于过去的浅嵌入。

哈夫曼编码　第 15 章专门讨论多模块应用。本章的主要目的是展示一个实际的大型程序例子。

其他例子　其他的例子在第 17 章讲解，包括仿真、关系和图，这些例子用于展示惰性计算 (Lazy evaluation)。

致谢

一本书只能通过读者的反馈来改进，因此我非常感激每一位给予我帮助和评论的读者。在此，我特别要感谢 Thomas Schiling。他提出使用 cabal 和 hackage 以及更宽广的 Haskell 平台的建议非常珍贵。Eerke Boiten 校对了关于扑克游戏的材料。Olaf Chitil 根据他近期的教学经验提出了很多实践的建议。这些人们以及其他在肯特（Kent）和美国的同事讨论了关于在 Haskell 中构建 DSL 的意义以及单子在 DSL 中扮演的角色。

在伦敦 Erlang Solutions 的同事为 “Haskell 夜校” 带来了许多听众，这门课上使用了这本教材的部分内容。同学们的问题和讨论对于改进本书许多地方的表达提供了很大的帮助。

在 Addison-Wesley 工作的 Rufus Curnow 从本书第 3 版的编写之初到出版提供了许多支持，非常感谢他的贡献和耐心。非常感谢 Simon Lake 对于结尾工作的帮助，还要感谢审稿人给出了专业性和建设性的建议，这些建议既包括针对我建议的修改，也包括他们自己的建议。

如果没有 Haskell 社区的共同努力，这本书就不会顺利完成。你们不仅提供了一流的系统，如 GHC，而且创造了许多开源的套装软件库和应用程序，使得 Haskell 用起来更令人愉悦。非常感谢!

我非常感谢肯特大学给我休假的机会，让我完成这本书的新版本。如果没有这次假期，这本书还需要很多年才会问世。

最后，感谢我的家人在我编写这本书时给予我的支持、理解和鼓励，我将这本书献给

他们。

Simon Thompson

Canterbury, 2010 年 12 月

出版社鸣谢

感谢以下网站慷慨地允许使用他们的图片：

Fotolia.com: Seam Gladwell/Fotolia 79, 178

我们已经尽快追踪版权所有者，并对于无意的疏忽预先致歉。我们会在本书的后续版本中适当的地方致谢。

目　录