

代码之髓：编程语言核心概念 (图灵程序设计丛书)

作者：[日]西尾泰和

版权信息

书名：代码之髓：编程语言核心概念

作者：[日] 西尾泰和

译者：曾一鸣

ISBN：978-7-115-36153-0

本书由北京图灵文化发展有限公司发行数字版。版权所有，侵权必究。

您购买的图灵电子书仅供您个人使用，未经授权，不得以任何方式复制和传播本书内容。

我们愿意相信读者具有这样的良知和觉悟，与我们共同保护知识产权。

如果购买者有侵权行为，我们可能对该用户实施包括但不限于关闭该帐号等维权措施，并可能追究法律责任。

图灵社区会员 ptpress (libowen@ptpress.com.cn) 专享 尊重版权

[版权声明](#)

[前言](#)

[致谢](#)

[本书构成](#)

[示例代码下载](#)

[第 1 章 如何深入高效地学习语言](#)

[1.1 在比较中学习](#)

[语言不同，规则不同](#)

[C 语言和 Ruby 语言中的真假值](#)

[Java 语言中的真假值](#)

[1.2 在历史中学习](#)

[理解语言设计者的意图](#)

[应该学哪种语言，我们无从所知](#)

[学习适用于各种语言的知识](#)

[1.3 小结](#)

[第 2 章 程序设计语言诞生史](#)

[2.1 程序设计语言诞生的历史](#)

[连接电缆](#)

[程序内置](#)

[FORTRAN 语言问世](#)

[2.2 程序设计语言产生的原因](#)

[懒惰：程序员的三大美德之一](#)

[语言们各有各的便捷](#)

[2.3 小结](#)

[第 3 章 语法的诞生](#)

[3.1 什么是语法](#)

[运算符的优先顺序](#)

[语法是语言设计者制定的规则](#)

[3.2 栈机器和 FORTH 语言](#)

[计算的流程](#)

[如何表达计算顺序](#)

[现在仍然使用的栈机器](#)

[3.3 语法树和 LISP 语言](#)

[计算流](#)

[如何表达计算顺序](#)

[现在仍然使用的语法树](#)

[3.4 中缀表达式](#)

[语法分析器](#)

[规则的竞争](#)

[3.5 小结](#)

[第4章 程序的流程控制](#)

[4.1 结构化程序设计的诞生](#)

[4.2 if语句诞生以前](#)

[为什么会有if语句](#)

[为什么会有if..else语句](#)

[4.3 while语句——让反复执行的if语句更简洁](#)

[使用while语句的表达方式](#)

[不使用while语句的表达方式](#)

[4.4 for语句——让数值渐增的while语句更简洁](#)

[使用for语句的表达方式](#)

[不使用for语句的表达方式](#)

[foreach——根据处理的对象来控制循环操作](#)

[4.5 小结](#)

[第5章 函数](#)

[5.1 函数的作用](#)

[便于理解——如同一个组织](#)

[便于再利用——如同零部件](#)

[程序中再利用的特征](#)

[5.2 返回命令](#)

[函数的诞生](#)

[记录跳转目的地的专用内存](#)

[栈](#)

[5.3 递归调用](#)

[嵌套结构体的高效处理](#)

[嵌套结构体的处理方法](#)

[5.4 小结](#)

[第6章 错误处理](#)

[6.1 程序也会出错](#)

[6.2 如何传达错误](#)

[通过返回值传达出错信息](#)

[出错则跳转](#)

[6.3 将可能出错的代码括起来的语句结构](#)

[John Goodenough 的观点](#)

[引入 CLU 语言](#)

[引入 C++ 语言](#)

[引入 Windows NT 3.1](#)

[6.4 出口只要一个](#)

[为什么引入 finally](#)

[成对操作的无遗漏执行](#)

[6.5 何时抛出异常](#)

[函数调用时参数不足的情况](#)

[数组越界的情况](#)

[出错后就要立刻抛出异常](#)

[6.6 异常传递](#)

[异常传递的问题](#)

[Java 语言的检查型异常](#)

[检查型异常没有得到普及的原因](#)

[6.7 小结](#)

[第 7 章 名字和作用域](#)

[7.1 为什么要取名](#)

[怎样取名](#)

[名字冲突](#)

[如何避免冲突](#)

[7.2 作用域的演变](#)

[动态作用域](#)

[静态作用域](#)

[7.3 静态作用域是完美的吗](#)

[嵌套函数的问题](#)

[外部作用域的再绑定问题](#)

[7.4 小结](#)

[第 8 章 类型](#)

[8.1 什么是类型](#)

[8.2 数值的 on 和 off 的表达方式](#)

[数位的发明](#)

[七段数码管显示器](#)

[算盘](#)

[8.3 一个数位上需要几盏灯泡](#)

[从十进制到二进制](#)

[八进制与十六进制](#)

[8.4 如何表达实数](#)

[定点数——小数点位置确定](#)

[浮点数——数值本身包含小数部分何处开始的信息](#)

[8.5 为什么会出现类型](#)

[没有类型带来的麻烦](#)

[早期的 FORTRAN 语言中的类型](#)

[告诉处理器变量的类型](#)

[隐性类型转换](#)

[8.6 类型的各种展开](#)

[用户定义型和面向对象](#)

[作为功能的类型](#)

[总称型、泛型和模板](#)

[动态类型](#)

[类型推断](#)

[8.7 小结](#)

[第 9 章 容器和字符串](#)

[9.1 容器种类多样](#)

[9.2 为什么存在不同种类的容器](#)

[数组与链表](#)

[链表的长处与短处](#)

[语言的差异](#)

[9.3 字典、散列、关联数组](#)

[散列表](#)

[树](#)

[元素的读取时间](#)

[没有万能的容器](#)

[9.4 什么是字符](#)

[字符集和字符的编码方式](#)

[计算机诞生以前的编码](#)

[EDSAC 的字符编码](#)

[ASCII 时代和 EBCDIC 时代](#)

[日语的编码](#)

[Shift_JIS 编码对程序的破坏](#)

[魔术注释符](#)

[Unicode 带来了统一](#)

[9.5 什么是字符串](#)

[带有长度信息的 Pascal 语言字符串和不带这一信息的 C 语言字符串](#)

[1 个字符为 16 比特的 Java 语言字符串](#)

[Python 3 中引入的设计变更](#)

[Ruby 1.9 的挑战](#)

[9.6 小结](#)

[第 10 章 并发处理](#)

[10.1 什么是并发处理](#)

[10.2 细分后再执行](#)

[10.3 交替的两种方法](#)

[协作式多任务模式——在合适的节点交替](#)

[抢占式多任务模式——一定时间后进行交替](#)

[10.4 如何避免竞态条件](#)

[竞态条件成立的三个条件](#)

[没有共享——进程和 actor 模型](#)

[不修改——const、val、Immutable](#)

[不介入](#)

[10.5 锁的问题及对策](#)

[锁的问题](#)

[借助事务内存来解决](#)

[事务内存的历史](#)

[事务内存成功吗](#)

[10.6 ##10.5小结**](#)

[第 11 章 对象与类](#)

[11.1 什么是面向对象](#)

[内涵因语言而异的面向对象](#)

[对象是现实世界的模型](#)

[什么是类](#)

[11.2 归集变量与函数建立模型的方法](#)

[11.3 方法 1：模块、包](#)

[什么是模块、包](#)

[用 Perl 语言的包设计对象](#)

[光有模块不够用](#)

[分开保存数据](#)

[向参数传递不同的散列](#)

[把初始化处理也放入包中](#)

[把散列和包绑定在一起](#)

[11.4 方法 2：把函数也放入散列中](#)

[first class](#)

[把函数放入散列中](#)

[创建多个计数器](#)

[把共享的属性放入原型中](#)

[这就是面向对象吗](#)

[11.5 方法 3：闭包](#)

[什么是闭包](#)

[为什么叫做闭包](#)

[11.6 方法 4：类](#)

[霍尔设想的类](#)

[C++ 语言中的类](#)

[功能说明的作用](#)

[类的三大作用](#)

[11.7 小结](#)

[第 12 章 继承与代码再利用](#)

[12.1 什么是继承](#)

[继承的不同实现策略](#)

[继承是把双刃剑](#)

[里氏置换原则](#)

[12.2 多重继承](#)

[一种事物在多个分类中](#)

[多重继承对于实现方式再利用非常便利](#)

[12.3 多重继承的问题——还是有冲突](#)

[解决方法 1：禁止多重继承](#)

[解决方法 2：按顺序进行搜索](#)

[解决方法 3：混入式处理](#)

[解决方法 4：Trait](#)

[12.4 小结](#)

版权声明

CODING WO SASAERU GIJUTSU by Hirokazu Nishio

Copyright © 2013 Hirokazu Nishio

All rights reserved.

Original Japanese edition published by Gijyutsu-Hyoron Co., Ltd., Tokyo

This Simplified Chinese language edition published by arrangement with Gijyutsu-Hyoron Co., Ltd., Tokyo in care of Tuttle-Mori Agency, Inc., Tokyo

本书中文简体字版由 Gijyutsu-Hyoron Co., Ltd. 授权人民邮电出版社独家出版。未经出版者书面许可，不得以任何方式复制或抄袭本书内容。

版权所有，侵权必究。

前言

当今程序设计语言多种多样，可供阅读的资料也非常多。但一个人的学习时间是有限的，全部都学并不现实。

另外，信息技术瞬息万变，特定语言及工具很快便已陈旧。如果不能意识到这一点而有选择性地学习一些相对稳定的知识，所学的内容将逐渐失去价值。

那么，该学习哪些知识并如何学习呢？笔者认为在学习中需要做到以下三点。

- 在比较中学习
- 在历史中学习
- 在实践中学习

第一条是指通过比较多种语言，总结出某种语言的独有特点，以及多种语言的共有特点。

第二条是指通过追溯语言的发展历史，了解语言是如何产生、变化和消失的，探寻语言发展演变的轨迹。

第三条是指亲自进行程序设计。边实践边思考如何编程，才能深入理解语言设计者的意图，同时也能发现自己原先理解不到位之处。

在阅读了各种程序设计书籍之后，相信读者们都曾产生过很多疑问。本书的目的就是解答大家的这些疑惑。本书假设读者对程序设计还不是很熟悉，侧重讲解“在比较中学习”和“在历史中学习”。如果大家在阅读本书后能掌握这些学习方法，那我将不胜欣喜。

欢迎访问：电子书学习和下载网站 (<https://www.shgis.cn>)

文档名称：《代码之髓：编程语言核心概念（图灵程序设计丛书）》[日]西尾泰和 著.epub

请登录 <https://shgis.cn/post/295.html> 下载完整文档。

手机端请扫码查看：

