第18章 迭代器和泛型for

18.1 迭代器和闭包

迭代器是一种可以让我们遍历一个集合中所有元素的代码结构。

Lua中，通常使用函数表示迭代器：每一次调用函数时，含税会返回集合的下一个元素。

迭代器需要在连续的调用之间保存一些状态，这样才能知道当前迭代所处的位置及如何从当前位置步进到下一个位置。

闭包为保存状态提供了一种良好的机制。

一个闭包就是一个可以访问其自身环境中一个或多个局部变量的函数。

一个闭包结构通常涉及两个函数：闭包本身和一个用于创建该闭包及其封装变量的工厂。

编程：实现ipairs

编程：遍历来自标准输入的所有单词的迭代器

使用Lua语言编程的最终用户一般不会去定义迭代器。

18.2 泛型for的语法

上述迭代器有一个缺点，需要为每个新的循环创建一个新的闭包。

泛型for保存了三个值：一个迭代函数，一个不可变状态和一个控制变量。

for做的第一件事是对in后面的表达式求值。这些表达式应该返回三个值供for保存：迭代函数，不可变状态和控制变量的初始值。

在使用简单迭代器时，工厂只会返回迭代函数，不可变状态和控制变量都是nil。

在上述的初始化步骤完成后，for使用不可变状态和控制变量为参数来调用迭代函数。

for var\_1,...,var\_n in explist do block end等价于

do

local \_f,\_s,\_var = explist

while true do

local var\_1, ... , var\_n = \_f(\_s, \_var)

\_var = var\_1

if \_var == nil then break end

block

end

end

18.3 无状态迭代器

无状态迭代器就是一种自身不保存任何状态的迭代器。

编程：实现ipairs

函数pairs的迭代函数是Lua语言中的一个个基本函数next。

编程：实现pairs

编程：不调用pairs而直接使用next

编程：遍历链表的迭代器

18.4 按顺序遍历表

编程：编写一个按照键的顺序来遍历表的迭代器

18.5 迭代器的真实含义