

GF3002

脚本语言

刘志伟

刘志伟

lua迭代器和泛型for

实验1

余庆祥

2220631136

BIH

2022/10/23

|  |  |
| --- | --- |
| **1、实验名称** | **实验1 lua迭代器和泛型for** |
| **2、实验日期** | **2022.10.23** |
| **3、实验目的** | |
| 1. 理解迭代器的概念。 2. 掌握迭代器的编写。 3. 掌握泛型for的原理及应用。 | |
| **4、实验环境** | |
| win7以上兼容系统，lua解释器 | |
| **5、实验的步骤和方法** | |
| 1. 理解迭代器和泛型for的使用原理。 2. 通过泛型for结构实现求得任意一个输入数组中所有的元素之和的功能。 3. 将迭代器用无状态迭代器和复杂状态的迭代器改写 4. 写迭代器函数遍历所有VIP玩家（这里VIP玩家定义为点数大于等于15的玩家），功能类似于 pairs。分别使用三种方式遍历 | |
| **6、数据记录和计算** | |
| （1）理解迭代器和泛型for的使用原理代码：  t = {'a', 'b', 'c', 'd'}  for i, v in ipairs(t) do  print(v)  end  function values(t)  local i = 0  return function ()  i = i + 1  return t[i]  end  end  t = {10, 20, 30}  iter = values(t)  while true do  local element = iter()  if element == nil then  break  end  print(element)  end  for element in values(t) do  print(element)  end  （2）通过泛型for结构实现求得任意一个输入数组中所有的元素之和的功能代码：  t = {1, 2, 3, 4, 5, 6}  function add(...)  local sum = 0  for k, v in ipairs{...} do  sum = sum + v  end  return sum  end  print(add(unpack(t)))  （3）将迭代器用无状态迭代器和复杂状态的迭代器改写代码：  无状态迭代器代码：  function next(t, i)  i = i + 1  if t[i] then  return i, t[i]  end  end  function values(t)  return next, t, 0  end  for k, v in values(t) do  print(k, v)  end  复杂状态迭代器代码：  function iter(state)  state.i = state.i + 1  if state.t[state.i] then  return state.i, state.t[state.i]  end  end  function values(a)  return iter, {t = a, i = 0}  end  for i, v in values(t) do  print(i,v)  end  真正状态迭代器代码：  function ipairs(t, f)  local i = 0  i = i + 1  while t[i] do  f(i)  i = i + 1  end  end  ipairs(t, function(i)  print(i, t[i])  end)  （4）第四题代码：  使用闭包代码：  function my\_pairs(t)  local k  return function ()  k = next(t, k)  if k ~= nil then  while t[k].Point < 15 do  k = next(t, k)  if k == nil then  return nil  end  end  return next(t, k)  end  end  end  print("使用闭包:")  for k, v in my\_pairs(users) do  print(k, v.Name, v.Point)  end  使用无状态迭代器代码：  function iter(t, i)  i = next(t, i)  if t[i] then  while t[i].Point < 15 do  i = next(t, i)  if i == nil then  return nil  end  end  return i, t[i]  end  end  function my\_pairs(t)  return iter, t, next(t)  end  print("使用无状态迭代器代码:")  for k, v in my\_pairs(users) do  print(k, v.Name, v.Point)  end  使用table作为恒定状态的复杂状态迭代器代码：  function iter(state)  if state.t[state.i] then  state.i = next(state.t, state.i)  while state.t[state.i].Point < 15 do  state.i = next(state.t, state.i)  if state.i == nil then  return nil  end  end  return state.i, state.t[state.i]  end  end  function my\_pairs(param\_t)  return iter, {t = param\_t, i = next(param\_t)}  end  print("使用table作为恒定状态的复杂状态迭代器:")  for k, v in my\_pairs(users) do  print(k, v.Name, v.Point)  end  使用真正的迭代器代码：  function my\_pairs(t, f)  local i = next(t)  while t[i] do  while t[i].Point < 15 do  i = next(t, i)  if i == nil then  return nil  end  end  f(i)  i = next(t, i)  end  end  print("真正迭代器：")  my\_pairs(users, function(i)  print(i, users[i].Name, users[i].Point)  end) | |
| **7、实验结果或结论（**实验结果怎么样？你从这个实验你学会了什么？得出了什么结论？） | |
| （1）理解迭代器和泛型for的使用原理结果图：    （2）通过泛型for结构实现求得任意一个输入数组中所有的元素之和的功能结果图：    （3）将迭代器用无状态迭代器和复杂状态的迭代器改写结果图：    （4）第四题结果图： | |
| **8、备注或说明**  **）** | |
|  | |