Lua 学习笔记

郑华

2018 年 5 月 13 日

1 变量

赋值与定义 不需要指明类型，系统会根据给出的值进行确定

除此之外可以如下赋值

```
a, b, c = 0, 1, 2 -->a=0, b=1, c=2
```

局部变量 局部变量加个 local 关键字，不是则不加

2 控制结构

for ->

```
for i = 1, 10, 1 do --初始值 结束值 步长（+x）
 print(i)
end


--范型 for 循环
-- print all values of array 'a'
for i,v in ipairs(a) do print(v) end --i为当前的下标 v为当前下标的临时局部值

-- print all keys of table 't
for k in pairs(t) do print(k) end

--[[
1. 控制变量是局部变量
2. 不要修改控制变量的值
--]]
```

while ->

```
while i<x do
    print(i)
end
```

if ->

```
if i<x then
    print(i)
else
    print(i+1)
end
```

return -> 可以返回多个变量

```
function func(valueUsed)
    return returbValueOne,returnValueTwo... -->有几个返回值写几个
```

可变参数 Lua 函数可以接受可变数目的参数，和 C 语言类似在函数参数列表中使用三点（...）表示函数有可变的参数。Lua 将函数的参数放在一个叫 arg 的表中，除了参数以外，arg 表中还有一个域 n 表示参数的个数。

```
--有时候我们可能需要几个固定参数加上可变参数
function g (a, b, ...) end

--[[
CALL PARAMETERS
g(3) a=3, b=nil, arg={n=0}
g(3, 4) a=3, b=4, arg={n=0}
g(3, 4, 5, 8) a=3, b=4, arg={5, 8; n=2}
--]]

--重写 print 函数:
printResult = ""
function print(...)
    for i,v in ipairs(arg) do
        printResult = printResult .. tostring(v) .. "\t"
    end
    printResult = printResult .. "\n"
end
```

3 字符串

连接用符号- .. ->

```
print("Hello" .. 'HH') --> HelloHH
```

字符到数字的智能转换 ->

```
print("10"+11) --> HelloHH
```

字符串查找替换 会全部替换

```
a = "one␣string␣one"
b = string.gsub(a,"one","other")

print(a) -->one string one
print(b) -->other string other
```

注释 -> 单行注释是“–”

多行注释是“–[[–]]”

类型函数 type ->

```
print(type("Hello␣World")) --> string
print(type(10.4*3)) --> number
print(type(print)) --> function
print(type(true)) --> boolean
print(type(nil)) --> nil
```

查找函数 find ->

```
s,e = string.find("Hello␣Lua␣World!","World") -->Lua可以返回多个变量

print(s,e) --->s为目标字符串在给定字符串的起始位置，e则为终止位置 11 15
```

4 数组或表

下标从 1 开始不是 0 ->

```
days={"Sunday", "Monday", "Tuesday", "Wednesday",
"Thursday", "Friday", "Saturday"}

print(days[4]) --> Wednesday


tab = {sin(1), sin(2), sin(3), sin(4),
sin(5),sin(6), sin(7), sin(8)}

a = {x=0, y=0} <--> a = {}; a.x=0; a.y=0

w = {x=0, y=0, label="console"}
x = {sin(0), sin(1), sin(2)}
w[1] = "another␣field"

--不管用何种方式创建 table，我们都可以向表中添加或者删除任何类型的域，构造函数仅仅影响表的
    初始化。
x.f = w       --原来没有的东西如果写出来，则自动会加进到X

print(w["x"])     --> 0 即访问的是 表中的X项，如果是w[x] -->nil
print(w[1]) --> another field
print(x.f[1]) --> another field
w.x = nil -- remove field "x"X
```

为什么有的东西可以用下标访问，有些不能 原来，只有在表不提供任何关键字时，才会按照下标进行寻找，否则，必须按照提供的关键字访问

```
local a = {x = 12, mutou = 99, [3] = "hello"}
print(a["x"]);

local a = {x = 12, mutou = 99, [3] = "hello"}
print(a.x);


local a = {[1] = 12, [2] = 43, [3] = 45, [4] = 90}

--如果说，大家习惯了数组，用数字下标，又不想自己一个个数字地定义，比如:
local a = {12, 43, 45, 90}
print(a[1]);
```

5 Function

定义与使用函数 ->

```
function fact(n)
    if n == 0 then
        return 1
    else
        return n*fact(n-1)
    end
end

print("enter␣a␣number")

a = io.read("*number")
print(fact(a))
```

6 Lua 调用其他文件

调用 Lua ->

```
--lib.lua file
function norm(x,y)
    local n2 = x^2 + y^2
    return math.sqrt(n2)
end

function twice(x)
    return 2*x
end

--lua 调用file
dofile("lib.lua")
n = norm(3.4,1.0)
print(twice(n))
```