Lua 流程控制 Lua 运算符

Lua 函数

在Lua中，函数是对语句和表达式进行抽象的主要方法。既可以用来处理一些特殊的工作，也可以用来计算一些值。

Lua 提供了许多的内建函数，你可以很方便的在程序中调用它们，如print()函数可以将传入的参数打印在控制台上。

Lua 函数主要有两种用途：

- 1.完成指定的任务，这种情况下函数作为调用语句使用；
- 2.计算并返回值，这种情况下函数作为赋值语句的表达式使用。

函数定义

Lua 编程语言函数定义格式如下：

```
optional_function_scope function function_name( argument1, argument2, argument3..., argumentn)
    function_body
    return result_params_comma_separated
end
```

解析：

- **optional_function_scope:** 该参数是可选的制定函数是全局函数还是局部函数，未设置该参数默认为全局函数，如果你需要设置函数为局部函数需要使用关键字 **local**。
- **function_name:** 指定函数名称。
- **argument1, argument2, argument3..., argumentn:** 函数参数，多个参数以逗号隔开，函数也可以不带参数。
- **function_body:** 函数体，函数中需要执行的代码语句块。
- **result_params_comma_separated:** 函数返回值，Lua语言函数可以返回多个值，每个值以逗号隔开。

实例

以下实例定义了函数 **max()**，参数为 num1, num2，用于比较两值的大小，并返回最大值：

实例

```
--[[ 函数返回两个值的最大值 --]]
function max(num1, num2)

  if (num1 > num2) then
     result = num1;
  else
     result = num2;
  end

  return result;
end
-- 调用函数
print("两值比较最大值为 ",max(10,4))
print("两值比较最大值为 ",max(5,6))
```

以上代码执行结果为：

```
两值比较最大值为    10
两值比较最大值为    6
```

Lua 中我们可以将函数作为参数传递给函数，如下实例：

实例

```
myprint = function(param)
  print("这是打印函数 -   ##",param,"##")
end

function add(num1,num2,functionPrint)
  result = num1 + num2
  -- 调用传递的函数参数
  functionPrint(result)
end
myprint(10)
-- myprint 函数作为参数传递
add(2,5,myprint)
```

以上代码执行结果为：

```
这是打印函数 -   ##    10    ##
这是打印函数 -   ##    7    ##
```

多返回值

Lua函数可以返回多个结果值，比如string.find，其返回匹配串"开始和结束的下标"（如果不存在匹配串返回nil）。

```
> s, e = string.find("www.runoob.com", "runoob")
> print(s, e)
5    10
```

Lua函数中，在return后列出要返回的值的列表即可返回多值，如：

实例

```
function maximum (a)
    local mi = 1             -- 最大值索引
    local m = a[mi]          -- 最大值
    for i,val in ipairs(a) do
       if val > m then
          mi = i
          m = val
       end
    end
    return m, mi
end

print(maximum({8,10,23,12,5}))
```

以上代码执行结果为：

```
23    3
```

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。

```
  function add(...)
  local s = 0
   for i, v in ipairs{...} do   --> {...} 表示一个由所有可变长参数构成的数组
     s = s + v
   end
   return s
  end
  print(add(3,4,5,6,7))  --->25
```

我们可以将可变参数赋值给一个变量。

例如，我们计算几个数的平均值：

实例

```
function average(...)
   result = 0
   local arg={...}    --> arg 为一个表，局部变量
   for i,v in ipairs(arg) do
      result = result + v
   end
   print("总共传入 " .. #arg .. " 个数")
   return result/#arg
end

print("平均值为",average(10,5,3,4,5,6))
```

以上代码执行结果为：

```
总共传入 6 个数
平均值为    5.5
```

我们也可以通过 select('#', ...) 来获取可变参数的数量

实例

```
function average(...)
   result = 0
   local arg={...}
   for i,v in ipairs(arg) do
      result = result + v
   end
   print("总共传入 " .. select("#",...) .. " 个数")
   return result/select("#",...)
end

print("平均值为",average(10,5,3,4,5,6))
```

以上代码执行结果为：

```
总共传入 6 个数
平均值为    5.5
```

有时候我们可能需要几个固定参数加上可变参数，固定参数必须放在变长参数之前：

实例

```
function fwrite(fmt, ...)  ---> 固定的参数fmt
    return io.write(string.format(fmt, ...))
end

fwrite("runoob\n")       --->fmt = "runoob", 没有变长参数。
fwrite("%d%d\n", 1, 2)   --->fmt = "%d%d", 变长参数为 1 和 2
```

输出结果为：

```
runoob
12
```

通常在遍历变长参数的时候只需要使用 `{...}`，然而变长参数可能会包含一些 `nil`，那么就可以用 **select** 函数来访问变长参数了：**select('#', ...)** 或者 **select(n, ...)**

- `select('#', ...)` 返回可变参数的长度
- `select(n, ...)` 用于访问 n 到 **select('#',...)** 的参数

调用select时，必须传入一个固定实参selector(选择开关)和一系列变长参数。如果selector为数字n，那么select返回它的第n个可变实参，否则只能为字符串'#',这样select会返回变长参数的总数。例子代码：

实例

```
do
    function foo(...)
        for i = 1, select('#', ...) do  -->获取参数总数
            local arg = select(i, ...); -->读取参数
            print("arg", arg);
        end
    end

    foo(1, 2, 3, 4);
end
```

输出结果为：

```
arg   1
arg   2
arg   3
arg   4
```

Lua 流程控制 Lua 运算符