# Estudo sobre a Linguagem de Programação Lua

**Antonio de Oliveira Viana[1] e Neri Brandão Rocha[1]**

[1]Departamento de Ciência da Computação – Universidade Federal de Roraima (UFRR)
Boa Vista – RR – Brazil

anttoniio@gmail.com, neribrandao2@gmail.com

***Resumo.*** *Uma linguagem de programação estabelecida e robusta, rápida, portátil, embutível, poderosa e simples, pequena, livre, e de importância global, para o desenvolvimento de um banco de dados com informações de alunos, calcula média de um escopo de alunos e exibe situação de aprovação e reprovação e gráfico das médias. Lua.*

## 1. História do surgimento da linguagem

A linguagem de programação Lua surgiu em 1993, com a necessidade interna do Departamento de Informática de realizar tarefas difíceis na área da Informática. O primeiro projeto foi realizado numa parceria entre o Departamento de Informática da PUC e a Petrobrás. O apoio de que a Petróleo Brasileiro S/A precisava estava numa das operações mais delicadas da empresa: as escavações. Com essa nova linguagem, a Petrobrás pôde estabelecer um roteiro (pelo computador) detalhado de como seriam conduzidas as escavações. A empresa passou a economizar tempo e conseguiu organizar mais os detalhes específicos das escavações, como temperatura certa da água e local de atuação das escavadeiras (disse o professor Roberto Ierusalimschy, associado do Departamento de Informática da PUC-Rio). Depois do projeto inicial, surgiram vários outros envolvendo a nova linguagem de programação, como o desenvolvimento, em 1994, de um sistema de intranet (sistema fechado de Internet numa empresa) no Departamento de Informática da PUC-Rio. O professor Ierusalimschy disse, empolgado, que "o projeto começou numa época em que a Internet não era nem comentada no país. Dois anos depois, o sistema foi estendido para todos os estudantes do campus".

Surgida no TeCGraf? (Grupo de Tecnologia em Computação Gráfica, parceria da PUC-Rio com a Petrobrás), a linguagem de programação Lua foi criada por Roberto Ierusalimschy (engenheiro de sistemas com pós-doutorado na Universidade de Waterloo, no Canadá, professor associado do departamento de informática da PUC-Rio e consultor do TeCGraf?) juntamente com seus companheiros Waldemar Celes (professor de Ciências da Computação da PUC-Rio) e Luiz Henrique de Figueiredo (Matemático). O nome Lua veio do fato de que quando eles criaram, em 1993, planejavam uma linguagem maior, chamada SOL (Simple Object Language), mas depois desistiram dela e pensaram em

reduzi-la. Então, alguém sugeriu: já que vocês vão fazer algo menor do que o Sol... E veio o nome Lua.

## 2. Domínios de aplicação

Lua é usada em muitas aplicações industriais (e.g., Adobe's Photoshop Lightroom), com ênfase em sistemas embutidos (e.g., o middleware Ginga para TV digital) e jogos (e.g., World of Warcraft e Angry Birds). Lua é atualmente a linguagem de script mais usada em jogos. Lua tem um sólido manual de referência e existem vários livros sobre a linguagem. Várias versões de Lua foram lançadas e usadas em aplicações reais desde a sua criação em 1993.

## 3. Paradigmas suportados pela linguagem

Lua é uma linguagem de programação poderosa, eficiente e leve, projetada para estender aplicações. Ela permite programação procedural, programação orientada a objetos, programação funcional, programação orientada a dados e descrição de dados.

Lua combina sintaxe procedural simples com poderosas construções para descrição de dados baseadas em tabelas associativas e semântica extensível. Lua é tipada dinamicamente, é executada via interpretação de bytecodes para uma máquina virtual baseada em registradores, e tem gerenciamento automático de memória com coleta de lixo incremental. Essas características fazem de Lua uma linguagem ideal para configuração, automação (scripting) e prototipagem rápida.

## 4. Variáveis e tipos de dados

Há oito tipos básicos em Lua: nil, boolean, number, string, function, userdata, thread, e table. Nil é o tipo do valor nil, cuja propriedade principal é ser diferente de qualquer outro valor; ele geralmente representa a ausência de um valor útil. Boolean é o tipo dos valores false e true. Tanto nil como false tornam uma condição falsa; qualquer outro valor a torna verdadeira. Number representa tanto números inteiros como números reias (ponto flutuante de precisão dupla). String representa sequências imutáveis de bytes. Lua é 8 bits pura: cadeias podem conter qualquer valor de 8 bits, incluindo zeros ('\0') dentro delas.

Lua pode chamar (e manipular) funções escritas em Lua e funções escritas em C.

O tipo userdata é oferecido para permitir que dados C arbitrários sejam guardados em variáveis Lua. Um valor userdata é um ponteiro para um bloco de memória bruta. Há dois tipos de userdata: userdata completo, onde o bloco de memória é gerenciado por Lua, e userdata leve, onde o bloco de memória é gerenciado pelo hospedeiro. Userdata não possui operações pré-definidas em Lua, exceto atribuição e teste de identidade. Através do uso de metatabelas, o

programador pode definir operações para valores userdata completos. Valores userdata não podem ser criados ou modificados em Lua, somente através da API C. Isso garante a integridade de dados que pertencem ao programa hospedeiro.

O tipo thread representa fluxos de execução independentes e é usado para implementar co-rotinas. Não confunda fluxos de execução Lua com processos leves do sistema operacional. Lua dá suporte a co-rotinas em todos os sistemas, até mesmo naqueles que não dão suporte a processos leves.

O tipo table implementa arrays associativos, isto é, arrays que podem ser indexados não apenas com números, mas com qualquer valor Lua exceto nil e NaN (Not a Number, um valor numérico especial usado para representar resultados indefinidos ou não representáveis, tais como 0/0). Tabelas podem ser heterogêneas; isto é, elas podem conter valores de todos os tipos (exceto nil). Qualquer chave com valor nil não é considerada parte da tabela. De modo recíproco, qualquer chave que não é parte da tabela possui um valor nil associado. Tabelas são o único mecanismo de estruturação de dados em Lua; elas podem ser usadas para representar arrays comuns, sequências, tabelas de símbolos, conjuntos, registros, grafos, árvores, etc. Para representar registros, Lua usa o nome do campo como um índice. A linguagem dá suporte a essa representação fornecendo a.nome como açúcar sintático para a[“nome”]. Há várias maneiras convenientes para criar tabelas em Lua.

## 4. Comandos de controle

As estruturas de controle if, while, e repeat possuem o significado usual e a sintaxe familiar:

comando ::= while exp do bloco end

comando ::= repeat bloco until exp

comando ::= IF exp then bloco {elseif exp then bloco} [else bloco] end

Lua também possui um commando for, em duas variações.

A expressão da condição de uma estrutura de controle pode retornar qualquer valor. Tanto false quanto nil são considerados falso. Todos os valores diferentes de nil e false são considerados verdadeiro (em particular, o número 0 e a cadeia vazia também verdadeiro).

No laço repeat-until, o bloco interno não termina na palavra chave until, mas somente após a condição. Assim, a condição pode se referir a variáveis locais declaradas dentro do corpo do laço.

O comando goto transfere o controle do programa para um rótulo. Por razões sintáticas, rótulos em Lua são considerados comandos também:

comando ::= goto Nome

comando ::= rótulo

rótulo ::= ‘::’ Nome ‘::’

Um rótulos é visível em todo o bloco onde ele é definido, exceto dentro de blocos aninhados onde um rótulo com o mesmo nome é definido e dentro de funções aninhadas. Um goto pode fazer um desvio para qualquer rótulo visível desde que ele não entre no escopo de uma variável local.

Rótulos e comandos vazios são chamados de comandos nulos, uma vez que eles não realizam ações.

O comando break termina a execução de um laço while, repeat, ou for, fazendo um desvio para o próximo comando após o laço:

comando ::= break

Um break termina o laço mais interno.

O comando return é usado para retornar valores de uma função ou de um trecho (que é uma função disfarçada). Funções podem retornar mais do que um valor, assim a sintaxe para o comando return é

comando ::= return [listaexps] [‘;’]

O comando return pode somente ser escrito como o último comando de um bloco. Se for realmente necessário um return no meio de um bloco, então um bloco interno explícito pode ser usado, como na expressão idiomática do return end, porque agora return é o último comando de seu bloco (interno).

## 5. Escopo (regras de visibilidade)

Um bloco é uma lista de comandos, que são executados sequencialmente:

bloco ::= {comando}

Lua possui comandos vazios que permitem você separar comandos com ponto-e-vírgula, começar um bloco com um ponto-e-vírgula ou escrever dois ponto-e-vírgula em sequência:

comando ::= ‘;’

Chamadas de funções e atribuições podem começar com um abre parêntese.

Essa possibilidade leva a uma ambiguidade na gramática de Lua. Considere o seguinte fragmento:

a = b + c

(print o rio.write) (‘done’)

A gramática poderia vê-lo de duas maneiras:

a = b + c (print o rio.write) (‘done’)

a = b + c; (print o rio.write) (‘done’)

O parser corrente sempre vê tais construções da primeira maneira, interpretando o abre parêntese como o começo dos argumentos de uma chamada. Para evitar essa ambiguidade, é uma boa prática sempre preceder com um ponto-e-vírgula comandos que começam com um parêntese:

;(print o rio.wuite) (‘done’)

Um bloco pode ser explicitamente delimitado para produzir um único comando:

comando ::= do bloco end

Blocos explícitos são úteis para controlar o escopo de declarações de variáveis. Blocos explícitos são também algumas vezes usados para adicionar um comando return no meio de outro bloco.

## 6. Exemplo prático de uso da linguagem de programação

Um banco de dados com informações de alunos, calculo de média de um escopo de alunos, e exibindo situação de aprovação ou reprovação e gráfico das médias. Foi utilizado o paradigma procedural, pois apesar de simples é eficiente e suficiente para a aplicação. O ambiente de desenvolvimento utilizado foi ZeroBrane Studio.

## 7. Conclusões

Lua é linguagem estabelecida e robusta, Lua é rápida, Lua é portátil, Lua é embutível, Lua é poderosa (e simples), Lua é pequena, Lua é livre e Lua tem importância global.

## 8. Referências

A linguagem de Programação Lua. https://www.lua.org/portugues.html. <Acessado> 19.07.2017 as 23h.

Trabalho de Linguagens de Programação sobre LUA. https://lplua.wordpress.com/historico/. <Acessado> 19.07.2017 as 21h.