CENTRO FEDERAL DE EDUCAÇÃO TECNOLÓGICA
DE MINAS GERAIS

Linguagem de Programação

RODRIGO LOPES E VITOR SANTANA

Seminário da Linguagem Lua

Belo Horizonte

2021

RODRIGO LOPES E VITOR SANTANA

Seminário da Linguagem Lua

Seminário da disciplina de Linguagem de Programação onde são discutidos conceitos básicos da Linguagem Lua e diferentes abordagens utilizadas para essa linguagem .

Orientador(a): Prof. Andrei Rimsa

Belo Horizonte

2021

Sumário:

Introdução

A linguagem Lua foi desenvolvida no Brasil na Puc-Rio (Pontifícia Universidade Católica do Rio de Janeiro) em 1993, por um grupo de Tecnologia em Computação Gráfica (Tecgraf).

A Lua foi projetada para estender aplicações, normalmente ela trabalha em conjunto com outras linguagens, ela é uma linguagem de scripting muito poderosa. Ela é uma linguagem ideal para configuração, automação (scripting) e prototipagem rápida.

Motivação

A Lua é uma linguagem Multiparadigmas, com isso ela aceita a programação orientada a objetos, programação funcional, programação procedural,programação orientada a dados e descrição de dados.

A linguagem foi desenvolvida primeiramente para um projeto da Petrobras, entretanto devido a sua clareza, facilidade de aprendizado e eficiência ela é utilizada em vários ramos da programação:

- **Dispositivos Móveis:** o **Ginga** é o nome do middleware que, integrado ao sintonizador de TV digital, proporcionará interação entre o telespectador e a televisão digital aberta brasileira.
- **Games:** Um exemplo é o jogo World WarCraft que utiliza a linguagem Lua em sua implementação, porém vários outros jogos contemporâneos foram desenvolvidos em Lua.
- **Ferramentas:** Lua foi utilizada em ferramentas de edição tais como Adobe Photoshop Lightroom, e modeladores 3D como Celestia.

Características da Linguagem

Esta linguagem trata de uma linguagem de script em alto nível, possuindo tipagem dinâmica, como Python, sendo também leve e reflexiva.

Lua é uma linguagem multiparadigma. Ela é naturalmente imperativa, porém ela possui certas ferramentas que permitem a emulação de uma linguagem de orientação a objetos (classes), com o auxílio de ferramentas como meta-tabelas, e uma programação funcional (mas não estritamente funcional como Haskell), com o uso funções em suas estruturas. Similar a linguagens como Java, Lua é executada via interpretação de bytecodes, assim como possui mecanismos de auto-coleta de lixo.

Lua também é uma linguagem relativamente portátil, podendo ser carregada em qualquer máquina que possua um compilador padrão de C. E ,por fim, Lua é uma linguagem bem potente, sendo capaz de realizar avaliações preguiçosas graças ao auxílio de meta-mecanismos potentes.

Exemplos

Um exemplo simples

```lua
-- Comentário podem ser feito desta maneira
--[[
    E pode comentar em várias linha,
    desta forma
]]

print ("Hello World")
```

```
Hello World
```

Outros 14 exemplos

- Exemplo 1

  Nesse exemplo será apresentado como utilizar um laço de Repetição utilizando o **while.**

```lua
print("-----Exemplo 1-----")
iterador = 1

while iterador <= 10 do

    print(iterador)
    iterador = iterador + 1

end
```

```
-----Exemplo 1-----
1
2
3
4
5
6
7
8
9
10
```

- Exemplo 2

  Nesse exemplo será apresentado como utilizar um laço de repetição utilizando o **for.**

```lua
print("Exemplo 2")

-- A logica do for é que o i = 1, verifica se i<=10 e soma 1 a cada iteração

for i = 1,10,1 do
    print(i)
end
```

```
Exemplo 2
1
2
3
4
5
6
7
8
9
10
```

- Exemplo 3

  Nesse exemplo será apresentado como criar funções.

```
print("Exemplo 3")

-- Imprime os numeros pares entre 1 e num

function imprimePares(num)
  i = 1
  while i <= num do
    if i%2==0 then
      print(i)
    end
    i = i + 1
  end

end

imprimePares(10)
```

```
Exemplo 3
2
4
6
8
10
```

- Exemplo 4

  Nesse exemplo será apresentado como retornar valores com a função.

```
print("Exemplo 4")

-- Verifica se "num" é primo, caso seja retorna true se não retorna
false

function Primo(num)
  i = 1
  qtdDivis = 0
  while i <= num do
    if num % i == 0 then
      qtdDivis = qtdDivis + 1
    end
    i = i + 1
  end

  if qtdDivis <= 2 then
    return true
  else
    return false
  end

end

-- Verificando se os numeros 11 e 10 são primos
print (Primo(11))
print (Primo(10))
```

```
Exemplo 4
true
false
```

- Exemplo 5

  Nesse exemplo será apresentado como criar um vetor.

```lua
print("Exemplo 5 \n")

--Criação do Vetor
vetor = {10,20,30,40,50}

-- O operador "#" retorna o tamanho do vetor

-- Percorrer o vetor imprimindo cadao valor de cada posicao do vetor
for i=1,#vetor,1 do
  print(i .. " posicao contem o valor: " .. vetor[i])
end
```

```
Exemplo 5

1 posicao contem o valor: 10
2 posicao contem o valor: 20
3 posicao contem o valor: 30
4 posicao contem o valor: 40
5 posicao contem o valor: 50
>
```

- Exemplo 6

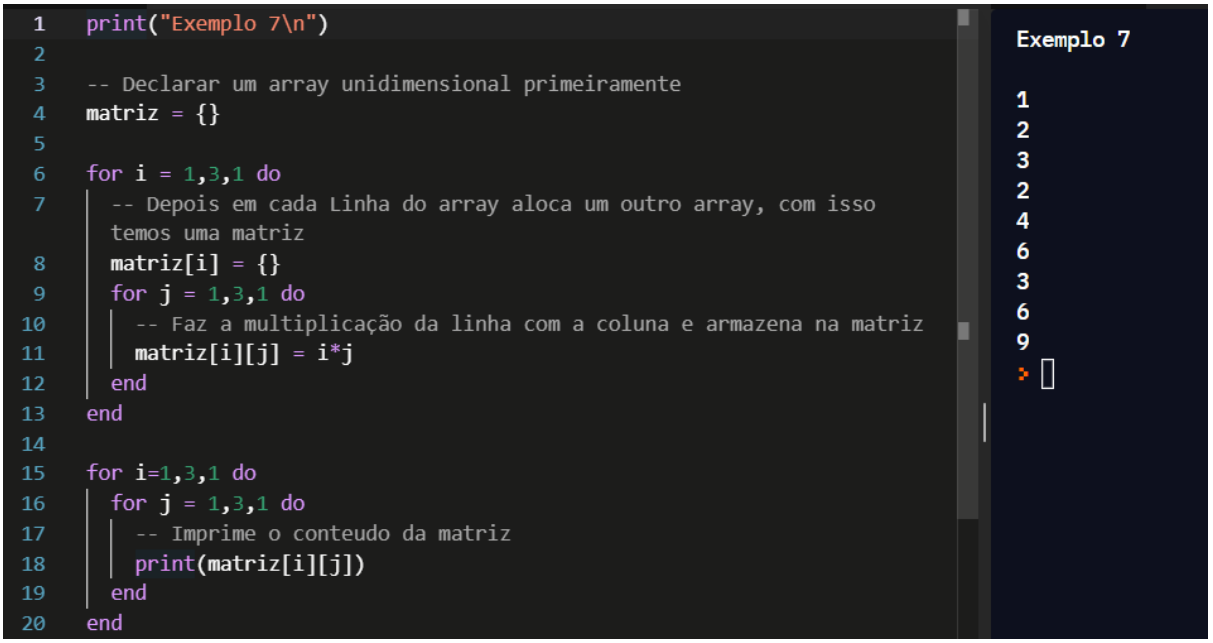
  Nesse exemplo será apresentado uma outra maneira da criação de vetores.

```lua
print("Exemplo 6\n")

-- Declaração do vetor
-- Não é necessario declarar o tamanho do vetor, pois a própria linguagem efetua o ajuste da tabela à medida que a tabela é utilizada
vetor = {}

-- Adiciona 5 elementos no vetor

for i = 1,5,1 do

  print("Digite o " .. i .. " elemento do vetor: " )
  vetor[i] = io.read("*number")

end

print("-----------------------------")

-- Imprime os valores colocados no vetor
for i = 1,5,1 do

  print(i.." posicao contem o valor : ".. vetor[i])

end
```

```
Exemplo 6

Digite o 1 elemento do vetor:
10
Digite o 2 elemento do vetor:
20
Digite o 3 elemento do vetor:
30
Digite o 4 elemento do vetor:
40
Digite o 5 elemento do vetor:
50
-----------------------------
1 posicao contem o valor : 10
2 posicao contem o valor : 20
3 posicao contem o valor : 30
4 posicao contem o valor : 40
5 posicao contem o valor : 50
>
```

- Exemplo 7

Nesse exemplo será apresentado uma forma de criar uma matriz bidimensional .

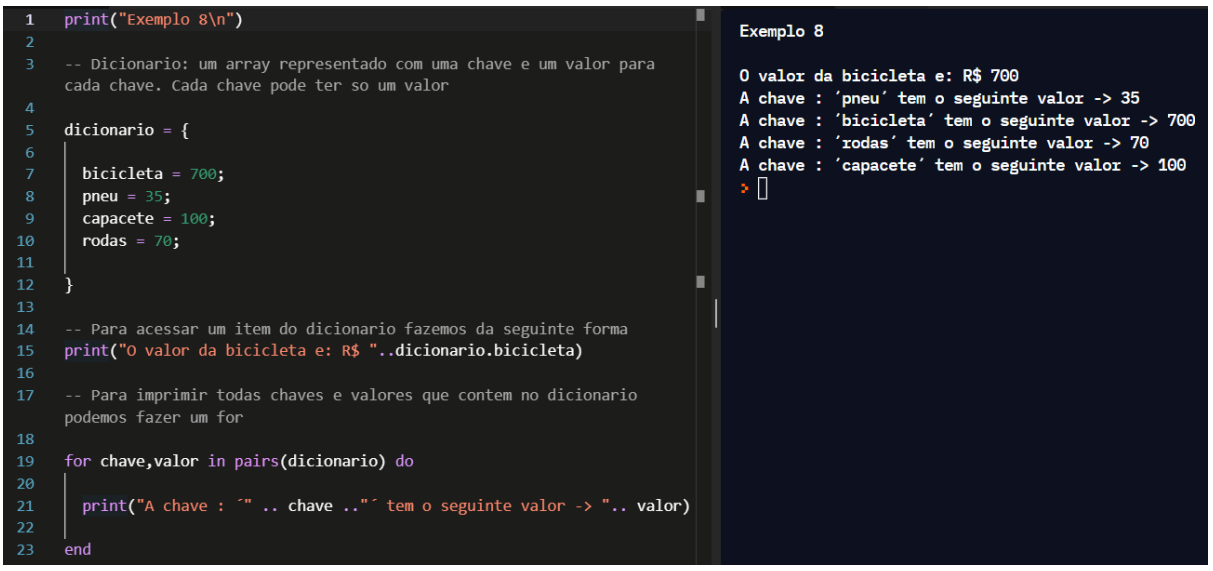


```lua
print("Exemplo 7\n")

-- Declarar um array unidimensional primeiramente
matriz = {}

for i = 1,3,1 do
  -- Depois em cada Linha do array aloca um outro array, com isso temos uma matriz
  matriz[i] = {}
  for j = 1,3,1 do
    -- Faz a multiplicação da linha com a coluna e armazena na matriz
    matriz[i][j] = i*j
  end
end

for i=1,3,1 do
  for j = 1,3,1 do
    -- Imprime o conteudo da matriz
    print(matriz[i][j])
  end
end
```

```
Exemplo 7

1
2
3
2
4
6
3
6
9
```

- Exemplo 8

Nesse exemplo será mostrado como criar um dicionário, visto que é uma ferramenta muito importante.

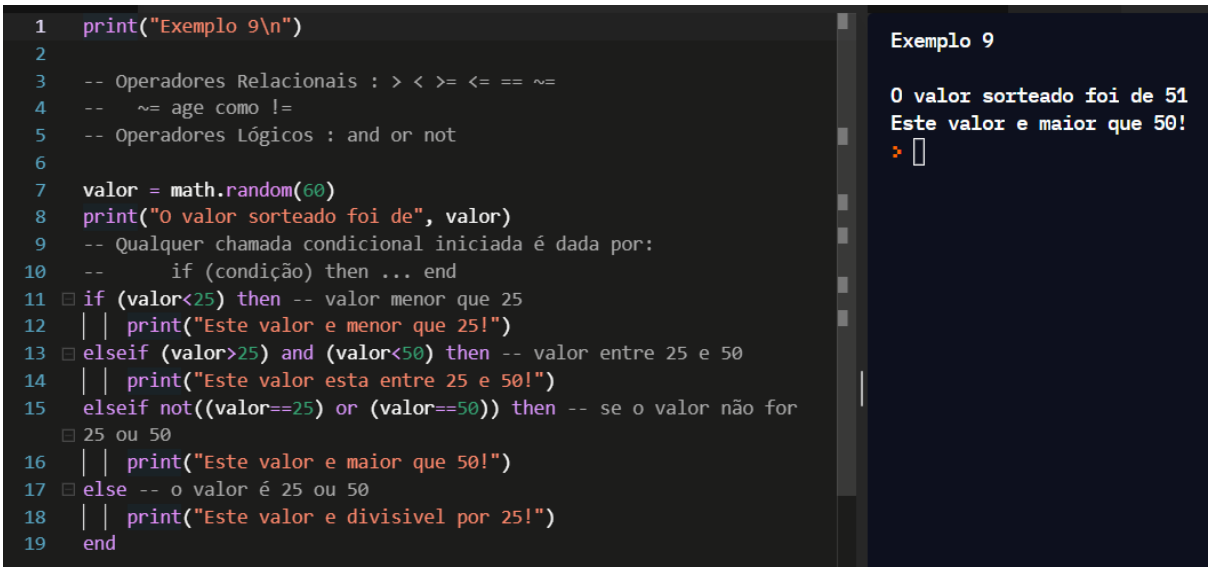


```lua
print("Exemplo 8\n")

-- Dicionario: um array representado com uma chave e um valor para cada chave. Cada chave pode ter so um valor

dicionario = {

  bicicleta = 700;
  pneu = 35;
  capacete = 100;
  rodas = 70;

}

-- Para acessar um item do dicionario fazemos da seguinte forma
print("O valor da bicicleta e: R$ "..dicionario.bicicleta)

-- Para imprimir todas chaves e valores que contem no dicionario podemos fazer um for

for chave,valor in pairs(dicionario) do

  print("A chave : ´" .. chave .."´ tem o seguinte valor -> ".. valor)

end
```

```
Exemplo 8

O valor da bicicleta e: R$ 700
A chave : ´pneu´ tem o seguinte valor -> 35
A chave : ´bicicleta´ tem o seguinte valor -> 700
A chave : ´rodas´ tem o seguinte valor -> 70
A chave : ´capacete´ tem o seguinte valor -> 100
```

- Exemplo 9

  Nesse exemplo é abordado os operadores relacionais.



```lua
print("Exemplo 9\n")

-- Operadores Relacionais : > < >= <= == ~=
--   ~= age como !=
-- Operadores Lógicos : and or not

valor = math.random(60)
print("O valor sorteado foi de", valor)
-- Qualquer chamada condicional iniciada é dada por:
--      if (condição) then ... end
if (valor<25) then -- valor menor que 25
  print("Este valor e menor que 25!")
elseif (valor>25) and (valor<50) then -- valor entre 25 e 50
  print("Este valor esta entre 25 e 50!")
elseif not((valor==25) or (valor==50)) then -- se o valor não for 25 ou 50
  print("Este valor e maior que 50!")
else -- o valor é 25 ou 50
  print("Este valor e divisivel por 25!")
end
```

```
Exemplo 9

O valor sorteado foi de 51
Este valor e maior que 50!
```

- Exemplo 10

  Nesse exemplo é apresentado os escopos de variáveis da Linguagem.

```lua
print("Exemplo 10\n")
-- Essencialmente, existem três tipos de escopos:
--     globais, locais ou campos de tabela

-- Uma varíavel não explicitamente especificada será global
-- Pode ser acessada em qualque escopo

g = 5 -- Variável global
tabela = {campo = 14} -- um campo de um tabela

print("A variavel global \'g\' e inicializada com ", g)

do
  -- Uma variável local pode ser acessada somente por funções dentro de seu escopo
  local l = 1
  print("A variavel local \'l\' e inicializada com ", l)
  repeat
    g = g + l
  until (g == tabela.campo) -- campo indica valores dentro de tabela
end

print("A variavel global \'g\' finalizada com  ", g)
print("A variavel local \'l\' finalizada com  ", l)
```

```
Exemplo 10

A variavel global 'g' e inicializada com    5
A variavel local 'l' e inicializada com     1
A variavel global 'g' finalizada com    14
A variavel local 'l' finalizada com     nil
```

- Exemplo 11

  Nesse exemplo é apresentado o conceito de clausuras.

```lua
print("Exemplo 11\n")

-- Lua trabalha com o conceito de clausuras
-- Uma função interna pode comunicar com variáveis locais do escopo externo
function contador()
    local i = 0;
    return function()
        i = i + 1
        return i
    end
end

-- Clausuras são específicas de instancias da função externa
c1 = contador()

print(c1()) -- retorna 1
print(c1()) -- retorna 2

-- Uma nova clausura é iniciada para c2
c2 = contador()

print(c2()) -- retorna 1, pois c2 usa outra clausura
print(c1()) -- retorna 3, pois continua a clausura de c1
```

```
Exemplo 11

1
2
1
3
```

- Exemplo 12

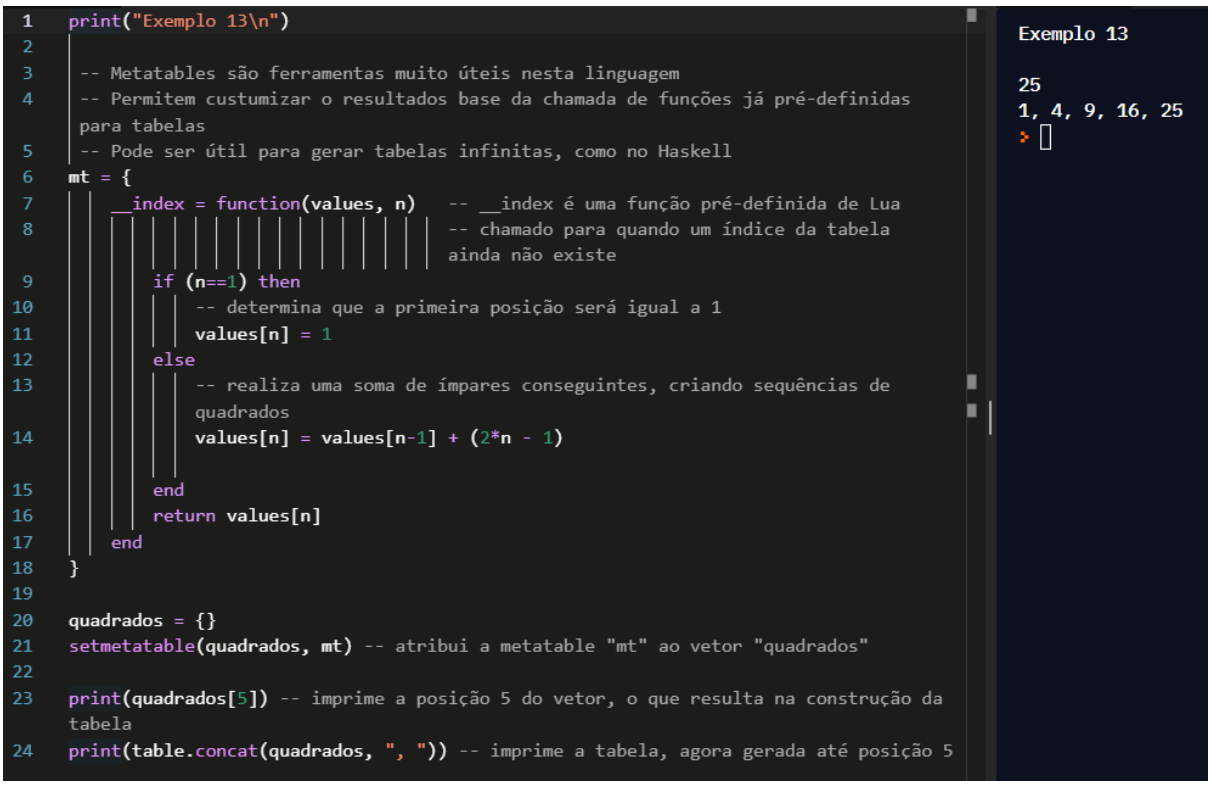
  Nesse exemplo é apresentado o conceito de co-rotina.

```lua
print("Exemplo 12\n")

-- Uma co-routina trata de trecho de códigos que podem serem interrompidos para serem executados posteriormente
co = coroutine.create(function()
    for i = 1, 10, 1 do
    print(i)
    print(coroutine.status(co))
    if i == 5 then
        coroutine.yield()  -- Interrompe a routina, mas não termina
    end
end end)
-- Inicia a co-routina
coroutine.resume(co)

-- Verifica o estado da routina, que foi interrompida em i=5
print(coroutine.status(co))

-- Efetua a routina novamente, terminando-a por completo
coroutine.resume(co)

-- Demonstra que a routina terminou
print(coroutine.status(co))
```

```
Exemplo 12

1
running
2
running
3
running
4
running
5
running
suspended
6
running
7
running
8
running
9
running
10
running
dead
```

- Exemplo 13

  Nesse exemplo é apresentado o conceito das Metatables

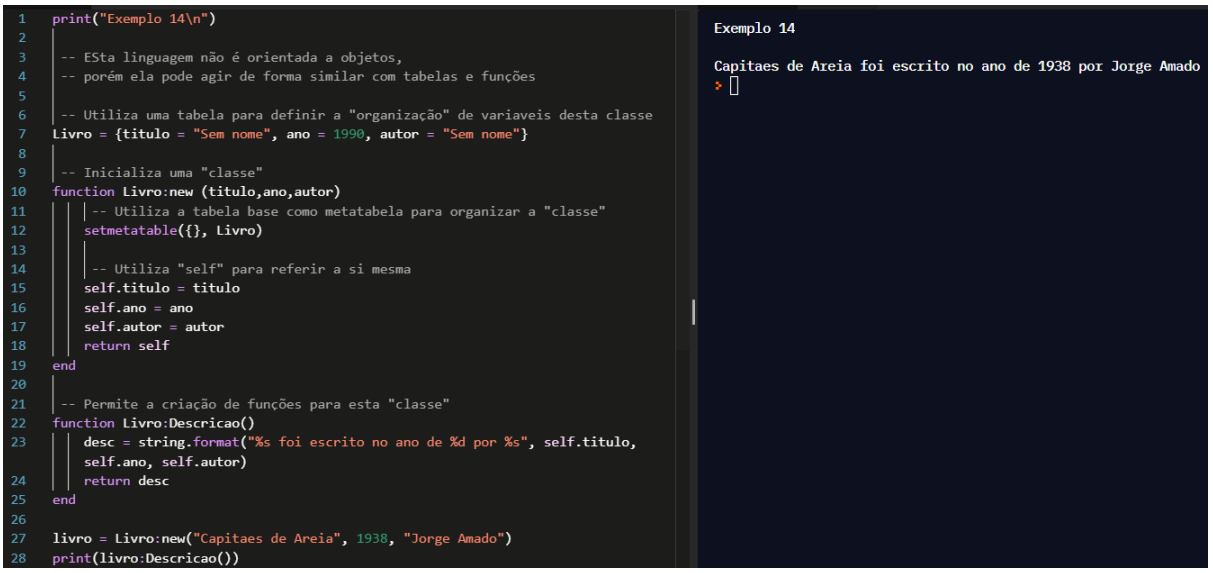


```lua
print("Exemplo 13\n")

-- Metatables são ferramentas muito úteis nesta linguagem
-- Permitem custumizar o resultados base da chamada de funções já pré-definidas para tabelas
-- Pode ser útil para gerar tabelas infinitas, como no Haskell
mt = {
    __index = function(values, n)   -- __index é uma função pré-definida de Lua
                                    -- chamado para quando um índice da tabela ainda não existe
        if (n==1) then
            -- determina que a primeira posição será igual a 1
            values[n] = 1
        else
            -- realiza uma soma de ímpares conseguintes, criando sequências de quadrados
            values[n] = values[n-1] + (2*n - 1)

        end
        return values[n]
    end
}

quadrados = {}
setmetatable(quadrados, mt) -- atribui a metatable "mt" ao vetor "quadrados"

print(quadrados[5]) -- imprime a posição 5 do vetor, o que resulta na construção da tabela
print(table.concat(quadrados, ", ")) -- imprime a tabela, agora gerada até posição 5
```

```
Exemplo 13

25
1, 4, 9, 16, 25
```

- Exemplo 14

  Nesse exemplo é apresentado a similaridade com a Orientação a Objetos utilizando tabelas e funções.



```lua
print("Exemplo 14\n")

-- ESta linguagem não é orientada a objetos,
-- porém ela pode agir de forma similar com tabelas e funções

-- Utiliza uma tabela para definir a "organização" de variaveis desta classe
Livro = {titulo = "Sem nome", ano = 1990, autor = "Sem nome"}

-- Inicializa uma "classe"
function Livro:new (titulo,ano,autor)
    -- Utiliza a tabela base como metatabela para organizar a "classe"
    setmetatable({}, Livro)

    -- Utiliza "self" para referir a si mesma
    self.titulo = titulo
    self.ano = ano
    self.autor = autor
    return self
end

-- Permite a criação de funções para esta "classe"
function Livro:Descricao()
    desc = string.format("%s foi escrito no ano de %d por %s", self.titulo, self.ano, self.autor)
    return desc
end

livro = Livro:new("Capitaes de Areia", 1938, "Jorge Amado")
print(livro:Descricao())
```

```
Exemplo 14

Capitaes de Areia foi escrito no ano de 1938 por Jorge Amado
```

Conclusão

Certamente, esta é uma linguagem de fácil manuseio e aprendizagem, e é certamente interessante como uma linguagem de origem brasileira acabou sendo bem única e utilizada por inúmeros projetos ao longo do mundo.

Atualmente, Lua ainda é muito utilizado em pesquisas de laboratórios, ferramentas contemporâneas e jogos populares que possuem grande controle da indústria. Sua última atualização ocorreu em dezembro do ano passado, e modificações no desenvolvimento desta linguagem ainda estão sendo realizadas. É seguro assumir que esta linguagem ainda terá sua marca no futuro, evidente pelo seu grande impacto em ferramentas e mídia do cotidiano.