


# Tópicos Especiais em Linguagens de Programação

## Aula 03 – Introdução à Linguagem Lua

Edirlei Soares de Lima  
<edirlei@iprj.uerj.br>



# Linguagem Lua

- Lua é uma **linguagem de programação** projetada para dar suporte à programação procedimental em geral.
- Oferece um bom suporte para programação orientada a objetos, programação funcional e programação orientada a dados.
- É utilizada em diversos ramos da programação, como no desenvolvimento de jogos, controle de robôs, processamento de texto, etc.



# Linguagem Lua

- Exemplos de **empresas que desenvolvem jogos** usando a linguagem Lua:
  - LucasArts, Blizzard, Microsoft, BioWare...
- Lua é inteiramente projetada, implementada e desenvolvida na **PUC-Rio**.
  - Nasceu e cresceu no **Tecgraf**, o Grupo de Tecnologia em Computação Gráfica da PUC-Rio.
  - Atualmente é desenvolvida no laboratório **Lablua**.

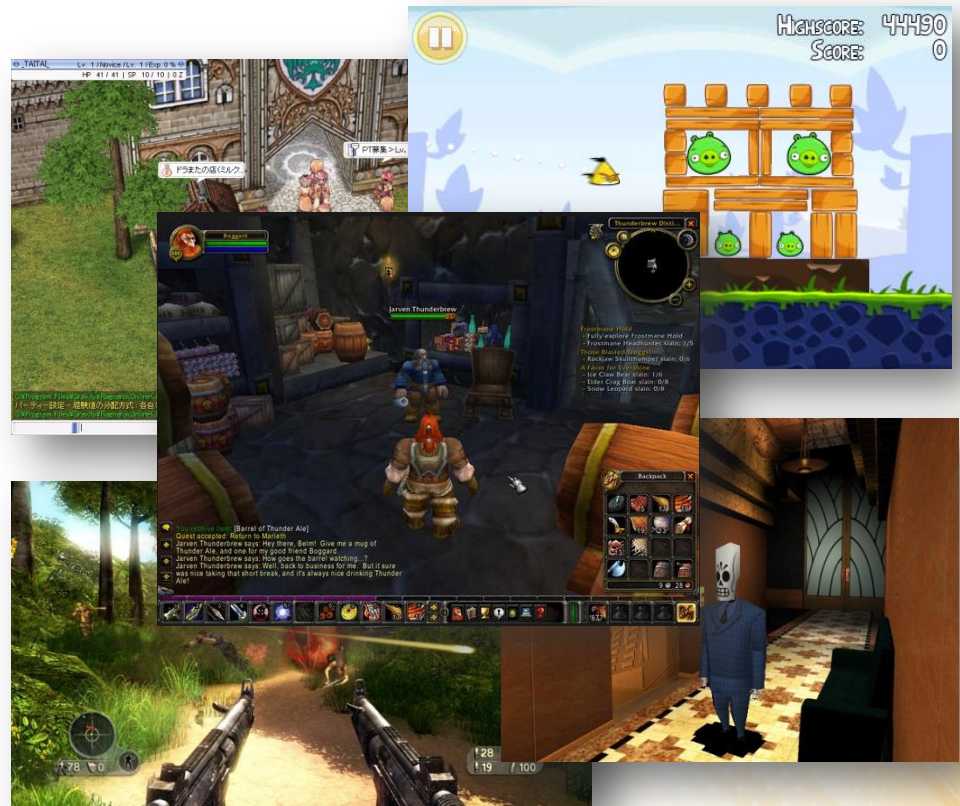


PUC  
RIO

# Linguagem Lua

- Exemplos de jogos que utilizam Lua:

- Angry Birds
- Civilization V
- Far Cry
- Grim Fandango
- Ragnarok
- Tibia
- World of Warcraft



# Por que escolher Lua?

- **É uma linguagem estabelecida e robusta:**
  - 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 tem um sólido **manual de referência** e existem vários livros sobre a linguagem.
- **É rápida:**
  - Outras linguagens de script aspiram ser "**tão rápidas quanto Lua**".
  - Vários benchmarks mostram Lua como a linguagem mais rápida dentre as linguagens de script interpretadas.

# Por que escolher Lua?

- **É portátil:**
  - Lua roda em todos os tipos de Unix e Windows, e também em dispositivos móveis (Android, iOS, BREW, Symbian, Windows Phone) e em microprocessadores embutidos (como ARM e Rabbit, para aplicações como Lego MindStorms).
- **É embutível:**
  - Lua tem uma API simples e bem documentada que permite uma integração forte com código escrito em outras linguagens.
- **É livre:**
  - Lua é software livre de código aberto, distribuída sob licença MIT.
  - Lua pode ser usada para quaisquer propósitos, incluindo propósitos comerciais, sem qualquer custo ou burocracia.

# Um programa em Lua

```
local cels
local fahr

io.write("Digite a temperatura em Celsius: ")
cels = io.read()

fahr = 1.8 * cels + 32

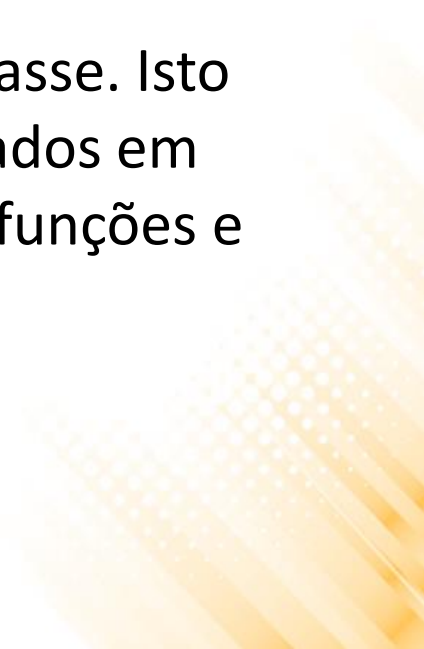
io.write("Temperatura em Fahrenheit: ", fahr, "\n")
```

# Variáveis em Lua

- **Variável** é um espaço reservado na memória do computador para armazenar um tipo de dado.
- Devem receber **nomes** para poderem ser referenciadas e modificadas quando necessário.
- Toda variável tem:
  - um nome
  - um tipo de dado
  - um valor
- **Restrição para nomes:** não é permitido começar o nome com um algarismo (0-9), alguns caracteres não são válidos (\*, -, /, +, ...), e palavras reservadas não podem ser utilizadas (if, for, while, ...).



# Variáveis em Lua

- Lua é uma linguagem **dinamicamente tipada**. Isto significa que variáveis não possuem tipos; somente valores possuem tipos.
    - Não existe definição de tipos na linguagem.
    - Todos os valores carregam o seu próprio tipo.
  - Todos os valores em Lua são valores de primeira classe. Isto significa que todos os valores podem ser armazenados em variáveis, passados como argumentos para outras funções e retornados como resultados.
- 

# Tipos de Variáveis da Linguagem Lua

Tipo	Exemplos de Valores
<code>number</code>	0, 1, 2.3, -2.3
<code>string</code>	"oi", "ola mundo", "teste 123", ""
<code>boolean</code>	true, false
<code>function</code>	0x1234567
<code>table</code>	0x2345678
<code>thread</code>	0x3456789
<code>userdata</code>	0x4567890
<code>nil</code>	nil

# Declaração de Variáveis em Lua

- Variáveis locais devem ser explicitamente declaradas
- Variáveis podem ser declaradas em conjunto
- Variáveis podem ser utilizadas sem serem declaradas (globais)

## Exemplos:

```
local a      -- declara uma variável local chamada a
local b      -- declara uma variável local chamada b
local d, e   -- declara duas variáveis locais
local d = 5  -- declaração e inicialização da variável
f = 10       -- declara e inicializa uma variável global
```

# Operadores Aritméticos

- **Operadores aritméticos** são usados para se realizar operações aritméticas com as variáveis e constantes.

Operação	Símbolo
Adição	+
Subtração	-
Multiplicação	*
Divisão	/
Resto da Divisão	%
Exponenciação	^

## Exemplos:

operador de atribuição

total = preco \* quantidade

media = (nota1 + nota2)/2

resultado = 3 \* (1 - 2) + 4 \* 2

resto = 4 % 2

res = b ^ 2

# Funções de Entrada e Saída em Lua

- **Função “write” da biblioteca “io”:** Permite a saída de dados, ou seja, a escrita de dados na tela.

```
io.write(constantes/variáveis/expressões...)
```

```
io.write(33)
```

tem como resultado a impressão da linha:

```
33
```

```
io.write("Valor = ", 33, " Total = ", 33 + 40)
```

saída:

```
Valor = 33 Total = 73
```

# Funções de Entrada e Saída em Lua

- Impressão de texto:

```
io.write("Curso de Programação de Jogos\nEm Lua")
```

exibe na tela a mensagem:

```
Curso de Programação de Jogos
```

```
Em Lua
```

# Funções de Entrada e Saída em Lua

- **Função “read” da biblioteca “io”:** Permite a entrada de dados, ou seja, a captura de valores fornecidos via teclado.

```
io.read()
```

```
local n  
n = io.read()
```

O valor digitado pelo usuário é armazenado na variável n

- **Importante:** o valor lido é sempre tratado como um **texto**. Em alguns casos é necessário convertê-lo para um número com o comando `tonumber`:

```
local n  
n = tonumber(io.read())
```

# Exemplo 01

- Escreva um programa que leia dois números inteiros e retorne a soma deles.

```
local numero1, numero2, resultado

io.write("Digite o primeiro numero: ")
numero1 = io.read()

io.write("Digite o segundo numero: ")
numero2 = io.read()

resultado = numero1 + numero2

io.write("Resultado da soma é ", resultado)
```

Nesses caso não convertemos os valores para números porque os operadores aritmético forçam a conversão automaticamente.



# Programando em Lua - Exemplo

- **Comentários:**

```
-- Programa para converter temperatura de Celsius em Fahrenheit

local cels    -- armazena temperatura em oC
local fahr    -- armazena temperatura em oF

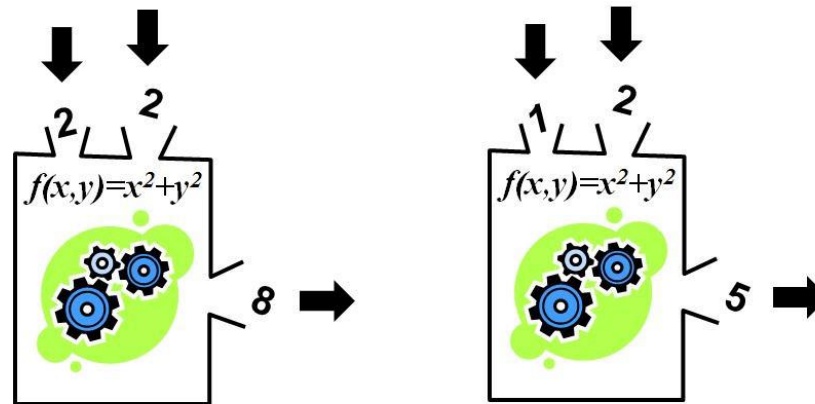
io.write("Digite a temperatura em Celsius: ")
cels = io.read()  -- captura valor fornecido via teclado

fahr = 1.8 * cels + 32  -- faz a conversão

-- exibe resultado na tela
io.write("Temperatura em Fahrenheit: ", fahr, "\n");
```

# Funções

- **Funções** em Lua são procedimentos que podem ser executados por outras partes do programa ou outras funções.



- **São utilizadas para:**
  - Simplificar e organizar o código;
  - Estender a linguagem de programação;

# Criando Novas Funções

Um programa Lua não pode ter duas funções com o mesmo nome.

```
function nome_da_funcao(parametro, parametro)
```

```
    variaveis locais
```

```
    instrucoes em Lua (comandos = expressoes e operadores)
```

```
    retorno
```

```
end
```

Se uma função não tem uma lista de parâmetros colocamos apenas o ().

Consiste no bloco de comandos que compõem a função.

# Criando Novas Funções

```
function celsius_fahrenheit(tc)
  local f
  f = 1.8 * tc + 32
  return f
end

local cels, fahr
io.write("Digite a temperatura em Celsius: ")
cels = io.read()
fahr = celsius_fahrenheit(cels)
io.write("Temperatura em Fahrenheit: ", fahr)
```

Podemos usar a função “celsius\_fahrenheit” em qualquer outro programa que precise de uma conversão deste tipo.

# Parâmetros e Valor de Retorno

- Exemplo:

```
function celsius_fahrenheit(tc)
```



Um único parâmetro de entrada

- Exemplo de função que recebe mais de um parâmetro:

```
function volume_cilindro(r, h)  
    local v  
    v = math.pi * (r ^ 2) * h  
    return v  
end
```



Dois parâmetros de entrada

# Parâmetros e Valor de Retorno

```
function volume_cilindro(r, h)
    local v
    v = math.pi * (r ^ 2) * h
    return v
end

local raio, altura, volume
io.write("Entre com o valor do raio: ")
raio = io.read()
io.write("Entre com o valor da altura: ")
altura = io.read()

volume = volume_cilindro(raio, altura)

io.write("Volume do cilindro = ", volume)
```

# Escopo de Variáveis

- Uma variável declarada dentro de uma função com a palavra “local” é chamada de **VARIÁVEL LOCAL**:
  - Ela somente é visível dentro da função que ela está declarada.
  - Assim que a função termina, os espaços de memória reservados para as suas variáveis locais são liberados e o programa não pode mais acessar esses espaços.

# Escopo de Variáveis

- **Variável Local:**

- Uma função pode ser chamada diversas vezes.
  - Para cada execução da função, os espaços das variáveis locais são automaticamente reservados, sendo então liberados ao final da execução.
- **Dentro de uma função não se tem acesso a variáveis locais definidas em outras funções.**
- **Os parâmetros de uma função também são variáveis automáticas com escopo dentro da função.**



# Escopo de Variáveis

```
function volume_cilindro(raio, altura)
  local volume
  volume = math.pi * (raio ^ 2) * altura
  return volume
end
```

Os nomes das variáveis locais são iguais mas a visibilidade é diferente.

```
local raio, altura, volume
io.write("Entre com o valor do raio: ")
raio = io.read()
io.write("Entre com o valor da altura: ")
altura = io.read()
```

```
volume = volume_cilindro(raio, altura)
```

```
io.write("Volume do cilindro = ", volume)
```


# Escopo de Variáveis

- Funções em Lua **recebem VALORES** e **retornam VALORES** (e não nomes de variáveis).
- **Os nomes podem coincidir, mas são variáveis distintas.**


```
function dobra_valor(x)
  x = x * 2
  return x
end
```

```
local x = 5.0
io.write(dobra_valor(x))
io.write(x)
```

Vai escrever 10.0 na tela



Vai escrever 5.0 na tela



# Exercícios

## **Lista de Exercícios 01 - Algoritmos e Variáveis**

<http://www.inf.puc-rio.br/~elima/jogos/>

