# LINGUAGEM LUA


NOMES: NILSON JUNIO

RAPHAEL BRAGANÇA

JOAO PEDRO PAMPLONA

# HISTÓRIA E INFLUÊNCIA

- Lua é uma linguagem desenvolvida no Brasil por brasileiros, criada na PUC em 1993
- Na época, precisava-se de uma linguagem: portável, capaz de descrever dados facilmente, amigável com C e de sintaxe fácil.
- Alguns de seus parentes mais próximos são icon, pascal, scheme, por sua concepção, e python, por sua facilidade de utilização por não-programadores.

# CARACTERÍSTICAS

- Multiparadigma
- Tipagem dinâmica forte
- Rápida
- Interpretada
- Portátil
- Facilidade de incorporação de uma aplicação Lua em outras linguagens
- Linguagem de script
- É uma linguagem de programação interpretada
- Simplicidade, tamanho pequeno e eficiência

# EXPRESSIVIDADE – METATABLE: LUA

```lua
Habilidades = {
    Flechada = {
        dano = 15,
        distancia = "15 metros"
    },
    Rajada = {
        dano = 30,
        distancia = "20 metros"
    },
    atacar = function()
        print("ATACAR!")
    end
}
Arqueiro = {
new = function(n)
    local novo_personagem = {
        nome = n,
        classe = "Arqueiro",
        vida = 35}
    setmetatable(novo_personagem, {__index = Habilidades})
    return novo_personagem
end
}
```

```lua
arq = Arqueiro.new("raphael")
imprimir = ""
for k,v in pairs(arq) do
    imprimir = imprimir..k..":"..v.."\n"
end
imprimir = imprimir.."Habilidades: ".."\nFlechada - "
for k,v in pairs(arq.Flechada) do
    imprimir = imprimir..k.." = "..v.." | "
end
imprimir = imprimir.."\n  Rajada - "
for k,v in pairs(arq.Rajada) do
    imprimir = imprimir..k.." = "..v.." | "
end
print(imprimir)
arq.atacar()
-------------------------------- EXPLICAÇÃO ↓ ----------------------
print("================= EXPLICAÇÃO ↓ ====================")
print("Habilidade é uma "..type(arq.Habilidades))
print("Flechada é uma "..type(arq.Rajada))
print("Rajada é uma "..type(arq.Rajada))
print("Atacar é uma ".. type(arq.atacar))
```

```
nome:raphael
vida:35
classe:Arqueiro
Habilidades:
Flechada - dano = 15 | distancia = 15 metros |
  Rajada - dano = 30 | distancia = 20 metros |
ATACAR!
================= EXPLICAÇÃO ↓ ====================
Habilidade é uma nil
Flechada é uma table
Rajada é uma table
Atacar é uma function
```

# EXPRESSIVIDADE - METATABLE: C

```c
#include <stdio.h>
typedef struct Habilidade
{
  char* nome;
  int dano;
  int distancia;
} Habilidade;
typedef struct Arqueiro
{
  char* nome;
  int vida;
  char* classe;
  struct Habilidade* habilidades;
}Arqueiro;

Habilidade Flechada()
{
  struct Habilidade flechada;
  flechada.nome = "Flechada";
  flechada.dano = 15;
  flechada.distancia = 15;
  return flechada;
}
Habilidade Rajada()
{
  struct Habilidade rajada;
  rajada.nome = "Rajada";
  rajada.dano = 30;
  rajada.distancia = 20;
  return rajada;
}
Arqueiro new_Arqueiro(char* n)
{
  struct Arqueiro arq;
  arq.nome = n;
  arq.vida = 35;
  arq.classe = "Arqueiro";
  arq.habilidades[0] = Flechada();
  arq.habilidades[1] = Rajada();
  return arq;
}

void imprimirArqueiro(Arqueiro arq)
{
  printf("Nome: %s\nVida: %d\nClasse: %s\nHabilidades:\n%s - dano = %d |"
  "distancia = %d metros |\n  %s - dano = %d | distancia = %d |",
  arq.nome,arq.vida, arq.classe,arq.habilidades[0].nome, arq.habilidades[0].dano,arq.habilidades[0].distancia,
  arq.habilidades[1].nome, arq.habilidades[1].dano, arq.habilidades[1].distancia);
}
int main()
{
  struct Arqueiro arq = new_Arqueiro("raphael");
  imprimirArqueiro(arq);
  return 0;
}
```

```
Nome: raphael
Vida: 35
Classe: Arqueiro
Habilidades:
Flechada - dano = 15 | distancia = 15 metros |
  Rajada - dano = 30 | distancia = 20 |
```

# EXPRESSIVIDADE – CO-ROTINA: LUA

```lua
corotina = coroutine.create(
  function ()
    parametro = 2
    teste = 1
    print("Primeiro parametro =", parametro)
    coroutine.yield()

    parametro = 5
    teste = 2
    print("Segundo  parametro =", parametro)
    coroutine.yield()

    parametro = 7
    teste = 3
    print("Terceiro parametro =", parametro)
    coroutine.yield()

    parametro = 100
    teste = "Erro"
  end
)
-- CÁLCULO DE VERIFICAÇÃO DE CONTROLE
-- Entrada de 1 á 10
entrada = 3
-- print("Entre com um dado de 1 á 10: ")
-- io.read() -- para inputar o dado do usuário
parametro = 0
calculo =  entrada * parametro

while (calculo <= 30) do
    coroutine.resume(corotina)
    calculo =  entrada * parametro
end

print("Numero de teste:", teste)
if calculo < 100 then
    print("Verificação realizada com sucesso")
else
    print("A entrada não atende o processo de Verificação")
end
```

```
Primeiro parametro =    2
Segundo  parametro =    5
Terceiro parametro =    7
Numero de teste:    Erro
A entrada não atende o processo de Verificação
```

# EXPRESSIVIDADE - CO-ROTINA: C

```c
#include <stdio.h>
// CÁLCULO DE VERIFICAÇÃO DE CONTROLE
int teste = 0;
int corotina(int n);

int main() {
    int parametroAux = 0;
    int calculo = 0;
    int entrada = 0;
    int n = 0;

    entrada = 3;
  // printf("Entre com um dado de 1 á 10: ");
  // scanf("%i",&entrada); // para inputar o dado do usuário
    calculo =  entrada * parametroAux;

  while (calculo <= 30) {
        parametroAux = corotina(parametroAux);
        calculo =  entrada * parametroAux;
  }

    printf("Numero de teste: %i \n", teste);
    if (calculo < 100) {
      printf("Verificacao realizada com sucesso \n");
    }
    else {
        printf("A entrada nao atende o processo de Verificao \n"); }
  return 0;
}
```

```c
int corotina(int parametro) {
    if (parametro == 0) {
        parametro = 2;
        teste = 1;
        printf("Primeiro parametro = %i \n", parametro);
        return parametro;
    }
    if (parametro == 2) {
        parametro = 5;
        teste = 2;
        printf("Segundo parametro = %i \n", parametro);
        return parametro;
    }
    if (parametro == 5) {
        parametro = 7;
        teste = 3;
        printf("Terceiro parametro = %i \n", parametro);
        return parametro;
    }
    else {
        parametro = 100;
        teste = -1;
    return parametro;
  }
}
```

```
Primeiro parametro = 2
Segundo parametro = 5
Terceiro parametro = 7
Numero de teste: -1
A entrada nao atende o processo de Verificao
```

# BIBLIOGRAFIA


- *ierusalimschy, roberto; figueiredo, luiz henrique de; celes, waldemar (2006). lua reference manual. rio de janeiro: lua.org. 103 páginas*
- *ierusalimschy, roberto (2006). programming in lua. rio de janeiro: lua.org. 252 páginas.*
- *jung, kurt; brown, aaron (2007). beginning lua programming. indianapolis: wiley publishing. 644 páginas.*
- https://pt.wikipedia.org/wiki/lua_(linguagem_de_programa%c3%a7%c3%a3o)
- http://www.lua.org

# LINGUAGEM GROOVY

# HISTÓRIA E INFLUÊNCIA

- Groovy é uma linguagem de programação orientada a objetos desenvolvida para a plataforma Java e foi lançada em janeiro de 2007.
- É conhecida como uma alternativa à linguagem Java que possui uma sintaxe concisa, familiar e fácil de aprender.
- É uma linguagem tanto estática quanto dinâmica e possui funcionalidades similares às linguagens Python, Ruby e Perl.

# CARACTERÍSTICAS

- Tipagem estática e tipagem dinâmica
- Clausuras
- Compilada Dinamicamente ou Interpretada
- Integra-se transparentemente com outros códigos e bibliotecas Java
- Sobrecarga de operadores
- Linguagem de programação e de script

# EXPRESSIVIDADE – OPERADORES DE SPREAD: GROOVY

```groovy
class Arqueiro {
    String nome
    int vida
    List<Habilidade> habilidades
}

class Habilidade {
    String nome
    int dano
    int distancia
}

def arqueiros = [
    new Arqueiro(nome: 'Joao Pedro', vida: 30, habilidades: [new Habilidade(nome: 'Rajada', dano: 10), new Habilidade(nome: 'Flechada', dano: 5)]),
    new Arqueiro(nome: 'Raphael', vida: 35, habilidades: [new Habilidade(nome: 'Chuva de Flechas', dano: 15), new Habilidade(nome: 'Flechada', dano: 5)]),
    new Arqueiro(nome: 'Nilson', vida: 33, habilidades: [new Habilidade(nome: 'Rajada', dano: 10), new Habilidade(nome: 'Chuva de Flechas', dano: 15)])
]


def nomes = arqueiros*.nome
def skills = arqueiros*.habilidades*.nome

println 'Nome dos Arqueiros: ' + nomes
println 'Habilidades: ' + skills
```

Saida

```
Nome dos Arqueiros: [Joao Pedro, Raphael, Nilson]
Habilidades: [[Rajada, Flechada], [Chuva de Flechas, Flechada], [Rajada, Chuva de Flechas]]
```

# EXPRESSIVIDADE – OPERADORES DE SPREAD: JAVA

```java
public class Spread {
    public static void main(String[] args) {
        List<Arqueiro> arqueiros = new ArrayList<Arqueiro>();
        Habilidade hab1 = new Habilidade("Rajada", 10);
        Habilidade hab2 = new Habilidade("Flechada", 5);
        Habilidade hab3 = new Habilidade("Chuva de Flechas", 15);
        Arqueiro arq1 = new Arqueiro("Joao Pedro", 30, Arrays.asList(hab1, hab2));
        Arqueiro arq2 = new Arqueiro("Raphael", 35, Arrays.asList(hab3, hab2));
        Arqueiro arq3 = new Arqueiro("Nilson", 33, Arrays.asList(hab1, hab3));

        arqueiros.add(arq1);
        arqueiros.add(arq2);
        arqueiros.add(arq3);

        System.out.print("Nomes dos Arqueiros: ");
        for(int i = 0; i < arqueiros.size(); i++){
            System.out.print(arqueiros.get(i).nome);
            if (i != arqueiros.size() - 1) {  System.out.print(", "); }
        }

        System.out.println(""); System.out.print("Habilidades: ");
        for(int i = 0; i < arqueiros.size(); i++){
            System.out.print("[");
            for(int j = 0; j < arqueiros.get(i).habilidades.size(); j++){
                System.out.print(arqueiros.get(i).habilidades.get(j).nome);
                if(j != arqueiros.get(i).habilidades.size() - 1) { System.out.print(", "); }
            }
            System.out.print("]");
            if (i != arqueiros.size() - 1) {  System.out.print(", "); }
        }
    }
}
```

Saída

```
Nomes dos Arqueiros: Joao Pedro, Raphael, Nilson
Habilidades: [Rajada, Flechada], [Chuva de Flechas, Flechada], [Rajada, Chuva de Flechas]
```

```java
public class Habilidade {
    String nome;
    int dano;

    public Habilidade(String nome, int dano){
        this.nome = nome;
        this.dano = dano;
    }
}
```

```java
public class Arqueiro {
    String nome;
    int vida;
    List<Habilidade> habilidades = new ArrayList<Habilidade>();

    public Arqueiro(String nome,
                    int vida, List<Habilidade> habilidades){
        this.nome = nome;
        this.vida = vida;
        this.habilidades = habilidades;
    }
}
```

# BIBLIOGRAFIA


- *http://groovy-lang.org/operators.html*
- *https://www.itexto.com.br/devkico/?p=654*
- https://pt.wikipedia.org/wiki/Groovy