

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: portátil, 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

```
1  Habilidades = {
2      Flechada = {
3          dano = 15,
4          distancia = "15 metros"
5      },
6      Rajada = {
7          dano = 30,
8          distancia = "20 metros"
9      },
10     atacar = function()
11         print("ATACAR!")
12     end
13 }
14 Arqueiro = {
15     new = function(n)
16         local novo_personagem = {
17             nome = n,
18             classe = "Arqueiro",
19             vida = 35
20         }
21         setmetatable(novo_personagem, {__index = Habilidades})
22         return novo_personagem
23     end
24 }
```

```
24     arq = Arqueiro.new("raphael")
25     imprimir = ""
26     for k,v in pairs(arq) do
27         imprimir = imprimir..k..": "..v.."\n"
28     end
29     imprimir = imprimir.."Habilidades: ".."\nFlechada - "
30     for k,v in pairs(arq.Flechada) do
31         imprimir = imprimir..k.." = "..v.." | "
32     end
33     imprimir = imprimir.."\n Rajada - "
34     for k,v in pairs(arq.Rajada) do
35         imprimir = imprimir..k.." = "..v.." | "
36     end
37     print(imprimir)
38     arq.atacar()
39     ----- EXPLICAÇÃO ↓ -----
40     print("===== EXPLICAÇÃO ↓ =====")
41     print("Habilidade é uma "..type(arq.Habilidades))
42     print("Flechada é uma "..type(arq.Rajada))
43     print("Rajada é uma "..type(arq.Rajada))
44     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

```
1  #include <stdio.h>
2  typedef struct Habilidade
3  {
4      char* nome;
5      int dano;
6      int distancia;
7  } Habilidade;
8  typedef struct Arqueiro
9  {
10     char* nome;
11     int vida;
12     char* classe;
13     struct Habilidade* habilidades;
14 } Arqueiro;
15
16 Habilidade Flechada()
17 {
18     struct Habilidade flechada;
19     flechada.nome = "Flechada";
20     flechada.dano = 15;
21     flechada.distancia = 15;
22     return flechada;
23 }
24
25 Habilidade Rajada()
26 {
27     struct Habilidade rajada;
28     rajada.nome = "Rajada";
29     rajada.dano = 30;
30     rajada.distancia = 20;
31     return rajada;
32 }
33
34 Arqueiro new_Arqueiro(char* n)
35 {
36     struct Arqueiro arq;
37     arq.nome = n;
38     arq.vida = 35;
39     arq.classe = "Arqueiro";
40     arq.habilidades[0] = Flechada();
41     arq.habilidades[1] = Rajada();
42     return arq;
43 }
44
45 void imprimirArqueiro(Arqueiro arq)
46 {
47     printf("Nome: %s\nVida: %d\nClasse: %s\nHabilidades:\n%s - dano = %d | "
48           "distancia = %d metros |\n  %s - dano = %d | distancia = %d |",
49           arq.nome, arq.vida, arq.classe, arq.habilidades[0].nome, arq.habilidades[0].dano, arq.habilidades[0].distancia,
50           arq.habilidades[1].nome, arq.habilidades[1].dano, arq.habilidades[1].distancia);
51 }
52
53 int main()
54 {
55     struct Arqueiro arq = new_Arqueiro("raphael");
56     imprimirArqueiro(arq);
57     return 0;
58 }
```

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

EXPRESSIVIDADE – CO-ROTINA: 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

```
1  #include <stdio.h>
2  // CÁLCULO DE VERIFICAÇÃO DE CONTROLE
3  int teste = 0;
4  int corotina(int n);
5
6  int main() {
7      int parametroAux = 0;
8      int calculo = 0;
9      int entrada = 0;
10     int n = 0;
11
12     entrada = 3;
13     // printf("Entre com um dado de 1 á 10: ");
14     // scanf("%i",&entrada); // para inputar o dado do usuário
15     calculo = entrada * parametroAux;
16
17     while (calculo <= 30) {
18         parametroAux = corotina(parametroAux);
19         calculo = entrada * parametroAux;
20     }
21
22     printf("Numero de teste: %i \n", teste);
23     if (calculo < 100) {
24         printf("Verificacao realizada com sucesso \n");
25     }
26     else {
27         printf("A entrada nao atende o processo de Verificao \n"); }
28     return 0;
29 }
```

```
31 int corotina(int parametro) {
32     if (parametro == 0) {
33         parametro = 2;
34         teste = 1;
35         printf("Primeiro parametro = %i \n", parametro);
36         return parametro;
37     }
38     if (parametro == 2) {
39         parametro = 5;
40         teste = 2;
41         printf("Segundo parametro = %i \n", parametro);
42         return parametro;
43     }
44     if (parametro == 5) {
45         parametro = 7;
46         teste = 3;
47         printf("Terceiro parametro = %i \n", parametro);
48         return parametro;
49     }
50     else {
51         parametro = 100;
52         teste = -1;
53         return parametro;
54     }
55 }
```

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_\(linguagem_de_programa%c3%a7%c3%a3o\)](https://pt.wikipedia.org/wiki/lua_(linguagem_de_programa%c3%a7%c3%a3o))
- <http://www.lua.org>