Descomplicando

Lua

Autor: 0x29a

Matemática

Lua é uma linguagem bem simples e dinâmica, ela é utilizada por diversas linguagens, principalmente por C e C++ para automação de tarefas simples. Além de dela ser simples, ela é extremamente “tipada”, ou seja, qualquer manuseio errado de dados, levam a um erro.

Sabendo que uma string é todo tipo de dado dentro de “” (aspas, seja simples ou dupla), vejamos um exemplo no JavaScript puro, onde é possível somar um texto com um número inteiro:

```
let a = 1;

let b “1”;

console.log(a + b);

# Result: 11
```

Já com lua, isso não acontece, gera um erro grotesco. Por isso vou passar primeiro todas as operações matemáticas de Lua:

Soma: +

Subtração: -

Multiplicação: *

Divisão: /

Resto da divisão: %

Exponencial: ^

Sabendo dos operadores, no Lua, existem dois tipos de números, mas eles não possuem uma classificação, que são os inteiros e os reais. O que diferem um do outro, o próprio nome já diz, o real sempre tem quebras como: 1.2, 2.4, já o inteiro nunca haverá quebras, afinal, ele é um inteiro: 5, 10, 23, 100...

Dado todo esse conhecimento, vale lembrar que Lua abrange algumas regras básicas da matemática, como os parênteses que indicam que tem que ser a primeira operação a ser realizada e por aí vai!

Então vamos ver a seguinte situação: Calcular a área de um quadrado, dado o tamanho de um dos lados como 24m:

```
lado = 24

areaQuadrado = lado * lado

print(areaQuadrado)
```

Funções

Sempre que tivemos alguma situação chata e repetitiva, ficamos enjoados e deixamos para depois e talvez você “engavete” isso e nunca mais toca no assunto. As funções vieram justamente para isso, vamos dar um exemplo: Todo momento em nosso código vamos repetir uma operação de soma entre dois números; assim invés de ficar escrevendo a operação toda hora, porque não escrevemos uma função?

Nela, podemos chamar a hora que quiser, no momento em que quiser, basta estar no escopo (no mesmo código declarado em algum lugar que conseguimos chamá-la, por isso geralmente escrevemos de cima pra baixo rsrs). Falando diretamente de nossa função agora, vamos precisar receber na hora em que chamar ela, dois valores, sendo eles A e B, após receber, vamos ter que retornar a soma de A com B, com isso, em Lua temos o famoso ***RETURN*** que basicamente retorna algo dentro de uma função e assim conseguimos coletar o retorno da função em uma variável por exemplo!

Chega de teoria e vamos para a prática, para criamos uma função, basta declarar do seguinte jeito:

```
function nomeDaFuncao()
    -- conteudo
end
```

Dentro do dela, conseguimos criar a ação que a torna viva, como vamos receber dois valores, vamos recebê-los pelo argumento, que se encontra dentro dos parênteses, ali dentro é onde definimos quais informações queremos receber quando ela for chamada, ficando assim de acordo com nosso exemplo:

```lua
function nomeDaFuncao(a, b)
    -- conteudo
end
```

Agora precisamos retornar a soma de A com B, para isso podemos tanto somar e colocar numa variável e retornar a variável com a soma, quanto até mesmo retornar direto à soma, que já é um nível like a pro!

```lua
function nomeDaFuncao(a, b)
    return a + b
end
```

Mas você só pode retornar no “bruto” assim, quando você tiver a plena certeza que você vai receber somente valores numéricos tanto no A quanto no B!

Agora, podemos chamar ela em uma variável ou direto em um print passando dois valores na hora de chamar, que teremos a soma de ambos:

```lua
function nomeDaFuncao(a, b)
    return a + b
end

soma = nomeDaFuncao(1, 1)
print(soma)
```

```
C:\Users\carto\Documents\Lua>lua 0x5.lua
2

C:\Users\carto\Documents\Lua>
```

IF

Sempre que precisamos conferir algo, utilizamos o ***IF***, traduzido do inglês, fica “se”, então vamos supor que temos uma variável ***B*** que carrega o valor 5 (numérico), e a variável ***X*** que carrega o mesmo valor que ***B***, e a gente precisa conferir se ambas são iguais, como fazemos isso?

Antes de responder isso, vamos ver como é a estrutura de um ***IF***:

```
if (statement) then
    -- Seu código
end
```

No ***STATEMENT*** é onde você cria a regra para verificar, de acordo com os nossos operadores lógicos em Lua, para verificar se um é igual a outro, basta colocar **“==”**, então se ***B*** for igual a ***X***, basta colocar ***“(b == x)”***, e continuar seu código ali no campo onde está comentado.

```
if (b == x) then
    print("B é igual a X")
end
```

E então, todo nosso código ficaria assim:

```
b = 5
x = 5
if (b == x) then
    print("B é igual a X")
end
```

Operadores lógicos

Igual	**=**
Diferente	**~=**
Maior	**>**
Menor	**<**
Menor ou igual	**<=**
Maior ou igual	**>=**

Mas quando eu iria usar algum operador lógico? Quase sempre, poucas tarefas vão poder ser concluídas sem usar um ***IF***, pensamos na seguinte situação: Temos uma loja, vendemos somente um item, este item custa *80R$*, e a gente tem o dever de conferir se o dinheiro do cliente consegue pagar o valor estipulado. Podemos resolver isso com um ***IF*** e o operador lógico **“>=”** (maior ou igual).

Mas porque não utilizar somente o operador lógico **“>”** (maior)? Porque se usarmos o maior, só vamos conseguir receber valores ACIMA do preço, ou seja, o custo seria 1 REAL mais caro que o valor estipulado, então o ***maior ou igual*** vem para resolver isso, porque se o cliente estiver com o dinheiro certinho, 80R$, ele consegue pagar, porque ele vai ser igual ao valor estipulado, e então nosso código ficaria assim:

```
moneyClient = 80
price = 80

if (moneyClient >= price) then
  moneyClient = moneyClient - price
  print("Item pago!")
end
```

No fim, após o ***IF***, deixamos o cliente com ***0R$***, pois ele tem o valor correto e na subtração iria sobrar 0 (zero);

Exercícios

1 - Crie uma função que receba três parâmetros, sendo todos numéricos e some os dois (2) primeiros, por fim, confira se a soma deu o valor do terceiro e retorne ***true*** caso o último parâmetro seja **igual** aos dois primeiros parâmetros somados e ***false*** para o **contrário**.

2 - Em uma variável onde **só** pode receber os valores numéricos um (1) e dois (2), crie uma função que receba esta variável e se o valor for um (1), retorne ***true***, caso contrário ***false***.

3 - Crie uma variável onde tenha um número de **sua** escolha, e mostre na tela as seguintes **operações**: (variável * variável), (variável + variável), (variável - variável) e ((variável / 2) + ((variável -2) - variável * variável)).