

**== Laboratório 1: Exercícios de Lua ==**

Para esquentar...

Execute no terminal o seguinte programa para verificar se o interpretador Lua está funcionando corretamente:

```
lua -e 'print "hello world!"'
```

Caso tudo ocorra bem, faça os seguintes exercícios:

1. Dado um arquivo de texto onde as linhas ímpares possuem o nome de uma pessoa e as linhas pares o sobrenome correspondente ao nome da linha anterior, escreva um programa em Lua que leia todas as linhas do arquivo, armazene as informações em uma tabela associativa e imprima a tabela resultante na tela. Note que cada elemento da tabela é um par chave/valor igual ao par nome/sobrenome de cada contato, ou seja, `tabela[nome]=sobrenome`. Utilize o `for` genérico e a função iteradora `io.lines(nomedoarquivo)` para ler as linhas do arquivo. Assuma que o arquivo texto já existe, por exemplo, "arquivo.txt", e que está no mesmo diretório do programa a ser desenvolvido. Modularize o código e apresente o resultado na tela.
2. Crie uma tabela de números inteiros e a ordene utilizando o algoritmo Quicksort. A inserção de elementos, assim como a ordenação, deve ser feita através de funções que recebam como entrada uma tabela. Usem a função "table.insert (tabela, elemento)" para inserção de elementos na tabela. Exiba o resultado na tela.

Dica: Para truncar a divisão, use a função piso "math.floor(x)".

O pseudocódigo do Quicksort é dado abaixo:

```
quickSort (array, indice_inicio, indice_fim)
  inicio, fim ← recede indice_inicio e indice_fim
  pivo ← recebe array [(inicio + fim)/2]
  enquanto inicio <= fim faça
    enquanto array [inicio] < pivo faça
      incrementa inicio de 1
    fim do enquanto
    enquanto array [fim] > pivo faça
      decrementa fim de 1
    fim do enquanto
    se inicio <= fim então
      troca array [inicio], array [fim]
      incrementa inicio de 1
      decrementa fim de 1
    fim do se
  fim do enquanto
  se indice_inicio < fim então
    quickSort (array, indice_inicio, fim)
  fim do se
```

```
se inicio < indice_fim então
    quickSort (array, inicio, indice_fim)
fim do se
fim do quicksort
```

3. Altere o programa da questão anterior para que seja possível criar tabelas associativas, inserir as tabelas associativas em uma tabela do tipo array e, por fim, ordenar a tabela em função de um dos elementos das tabelas associativas. A ideia é criar uma tabela que represente um carrinho de compra de produtos programados como tabelas associativas.

Por exemplo, os produtos `{id="arroz", preço=20, peso=5}` e `{id="feijao", preço=5, peso=1}` devem ser inseridos em um carrinho representado por uma tabela. Sendo assim, `carrinho={{id="arroz", preço=20, peso=5}, {id="feijao", preço=5, peso=1}}`.

Após a inserção dos produtos no carrinho, o programa deve ser capaz de ordenar de maneira decrescente os produtos em função do preço. Faça o programa modular e apresente o resultado final na tela.

4. Assumindo que o carrinho suporta um determinado peso máximo menor do que a soma do peso de todos os produtos disponíveis, crie um programa que maximize a soma de preços dos produtos a serem inseridos no carrinho. Faça uma função que resolva este problema e imprima o resultado na tela.

Dica: Utilize uma abordagem gulosa, ou seja, comece pelos produtos com os maiores preços, tomando cuidado para não ultrapassar o peso máximo permitido. Este problema é conhecido como Problema da "Mochila" (*Knapsack problem*).

5. Altere o programa anterior para que, ao invés do preço, seja considerada a relação preço/peso. Será que é uma boa ideia de mudança?
6. Escreva um programa que possibilite os usuários escolher uma operação matemática simples como soma, subtração etc. a partir de um índice de tabela. Para isso, crie as funções matemáticas e as insira em uma tabela. Exiba os resultados obtidos assumindo dois números inteiros como entrada das funções matemáticas. Modularize o código e apresente os resultados na tela.

## == Respostas ==

### \* Contribuição valiosa do ex-aluno Victor Raposo Ravaglia de Oliveira \*

#### 1.

```

/*****/
function lerNomesArquivo (nomeArquivo)
    local nomes = {}
    local totalLinhas = 0
    for linha in io.lines (nomeArquivo) do
        totalLinhas = totalLinhas + 1
        if (totalLinhas % 2 ~= 0) then
            nome = linha

            else
                nomes [nome] = linha
            end
        end
    end
    return nomes
end

function printNomes (nomes)
    for nome, sobrenome in pairs (nomes) do
        print (nome, "\t", sobrenome)
    end
end

local nomes = lerNomesArquivo ("arquivo.txt")
printNomes (nomes)

/*****/

```

#### 2.

```

/*****/
function quicksort(array, indice_inicio, indice_fim)
    local inicio = indice_inicio
    local fim = indice_fim
    local pivo = array[math.floor((indice_fim - indice_inicio)/2) + indice_inicio]
    while (inicio <= fim) do
        while (array[inicio] < pivo) do
            inicio = inicio + 1
        end
        while (array[fim] > pivo) do
            fim = fim - 1
        end
        if (inicio <= fim) then
            array[inicio], array[fim] = array[fim], array[inicio]
            inicio = inicio + 1
            fim = fim - 1
        end
    end
    if (indice_inicio < fim) then
        quicksort(array, indice_inicio, fim)
    end
    if inicio < indice_fim then
        quicksort(array, inicio, indice_fim)
    end
end

function preencherTabela (max)
    local tab = {}
    for i=1, max do
        print ("Entre com um inteiro:")
    end
end

```

```

        local number = io.read ("*number")
        table.insert (tab, number)
    end
    return tab
end

function imprimirTabela (tt)
    for idx, valor in ipairs (tt) do
        print (valor)
    end
end

local tab = preencherTabela (5)

print ("\n\nTabela desordenada:")
imprimirTabela (tab)

print ("\n\nTabela ordenada:")
quicksort (tab, 1, 5)
imprimirTabela (tab)

```

/\*\*\*\*\*/

### 3.

/\*\*\*\*\*/

```

function quicksort(array, indice_inicio, indice_fim)
    local inicio = indice_inicio
    local fim = indice_fim

    local pivo =
        array[math.floor((indice_fim - indice_inicio)/2) + indice_inicio]["preco"]

    while (inicio <= fim) do
        while (array[inicio]["preco"] < pivo) do
            inicio = inicio + 1
        end
        while (array[fim]["preco"] > pivo) do
            fim = fim - 1
        end
        if (inicio <= fim) then
            array[inicio], array[fim] = array[fim], array[inicio]
            inicio = inicio + 1
            fim = fim - 1
        end
    end
    if (indice_inicio < fim) then
        quicksort(array, indice_inicio, fim)
    end
    if inicio < indice_fim then
        quicksort(array, inicio, indice_fim)
    end
end

function lerItem ()
    local item = {}
    print ("Entre com um identificador:")
    local id = io.read ()
    item ["id"] = id
    print ("Entre com um preço:")
    local preco = tonumber (io.read ())
    item ["preco"] = preco
    print ("Entre com um peso:")
    local peso = tonumber (io.read ())
    item ["peso"] = peso
    return item
end

function imprimirTabela (tt)

```

```
        for idx, item in ipairs (tt) do
            print (idx, item.id, item.preco, item.peso)
        end
    end

-- Main --

local tab = {}
local item = {}
local numeroItens = 0

print ("Entre com o numero de itens:")
numeroItens = tonumber(io.read ())

for i=1, numeroItens do
    item = lerItem ()
    table.insert (tab, item)
end

print ("\n\nTabela desordenada:")
imprimirTabela (tab)

print ("\n\nTabela ordenada:")
quicksort (tab, 1, numeroItens)
imprimirTabela (tab)

/*****/
```