Pontifícia Universidade Católica
do Rio de Janeiro

Introdução à Engenharia

ENG1000

Aula 06 – Funções

2016.1

Prof. Augusto Baffa
<abaffa@inf.puc-rio.br>

Organização de Código

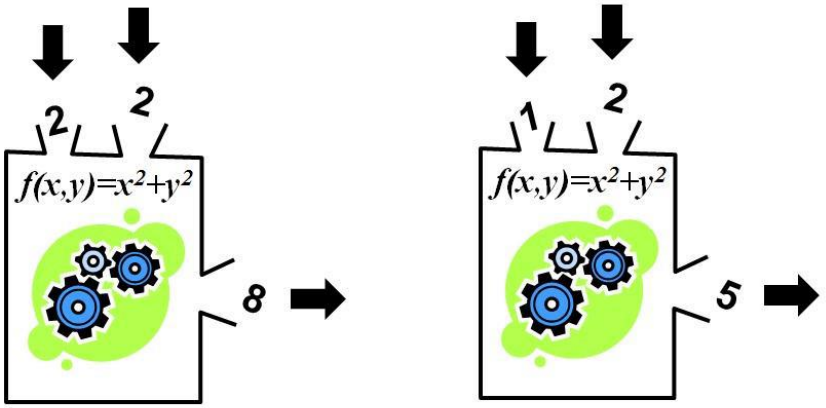
- Um programa representa a implementação de uma solução de um determinado problema.

- **É fundamental o programa seja escrito de forma organizada**, a fim de facilitar a manutenção, o re-uso, a adaptação do código, durante o processo de desenvolvimento ou no futuro.

- Uma maneira de organizar o código é realizando a modularização do programa em **funções**.

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;

Funções

- **Um programa em Lua é dividido em pequenas funções:**
  - Bons programas são compostos por diversas pequenas funções.
  - Como o próprio nome diz, uma função representa uma funcionalidade.
  - A vantagem de se ter o código modularizado em funções é que o código fica mais fácil de entender, de manter, de atualizar e de reusar.

- Nós já estamos usando funções auxiliares para capturar dados oriundos do teclado (**io.read**) e também para imprimir dados na tela como saída (**io.write**).

Criando Novas Funções

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

```
function nome_da_funcao (parâmetros, parâmetro)
  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 essa função 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. Cada um deve ter seu tipo e nome declarados

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)
```

Parâmetros e Valor de Retorno

- Uma chamada de uma função pode aparecer dentro de uma expressão maior. Por exemplo, se quiséssemos calcular a metade do volume do cilindro:

```
volume = volume_cilindro(raio,altura)/2.0
```

- Também pode ser utilizada uma expressão válida na passagem de parâmetros:

```
volume = volume_cilindro(raio, 2*altura)
```

Escopo de Variáveis

- Uma variável declarada dentro de uma função 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



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)
```

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

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


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

Vai escrever 10.0 na tela

Vai escrever 5.0 na tela

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

Vai escrever 10.0 na tela

Vai escrever 10.0 na tela

Exercícios

Lista de Exercícios 02 - Funções

http://www.inf.puc-rio.br/~abaffa/eng1000/