2013

Decode Team

Instituto Politécnico de tomar

Portugol

Equivalências de estruturas entre Portugol e Lua

Índice

[Nota Geral: 3](#_Toc359443883)

[Algumas notas sobre Lua: 3](#_Toc359443884)

[Estrutura Início 3](#_Toc359443885)

[Início: 3](#_Toc359443886)

[Estrutura Fim 3](#_Toc359443887)

[Fim: 3](#_Toc359443888)

[Variáveis 4](#_Toc359443889)

[Equivalência entre TIPOS de variáveis 4](#_Toc359443890)

[Definição e atribuição de variáveis 4](#_Toc359443891)

[Estruturas input/output 4](#_Toc359443892)

[Input – Ler 4](#_Toc359443893)

[Output – Escrever 5](#_Toc359443894)

[Estruturas de Decisão 5](#_Toc359443895)

[Condição *“if”* e *“if else”* 5](#_Toc359443896)

[Condição *“while”* 7](#_Toc359443897)

[Condição *“do while”* 8](#_Toc359443898)

[Exemplos práticos 8](#_Toc359443899)

[Estrutura Conector 9](#_Toc359443900)

[Conector 9](#_Toc359443901)

[Funções 10](#_Toc359443902)

[Definir funções 10](#_Toc359443903)

[Definir função 10](#_Toc359443904)

[Chamada de funções 11](#_Toc359443905)

[Exemplos do uso de funções 11](#_Toc359443906)

[Estrutura de retorno 11](#_Toc359443907)

[Return 11](#_Toc359443908)

[Operadores 12](#_Toc359443909)

[Aritméticos 12](#_Toc359443910)

[Lógicos 12](#_Toc359443911)

[Relacionais 12](#_Toc359443912)

[ANEXO 13](#_Toc359443913)

[Algoritmo com o uso da condição “if” 13](#_Toc359443914)

[Fluxograma 13](#_Toc359443915)

[Código 13](#_Toc359443916)

[Esquema detalhado 14](#_Toc359443917)

[Algoritmo com o uso da condição “if else” 15](#_Toc359443918)

[Fluxograma 15](#_Toc359443919)

[Código: 15](#_Toc359443920)

[Esquema detalhado 16](#_Toc359443921)

[Algoritmo com o uso da condição “while” 17](#_Toc359443922)

[Fluxograma 17](#_Toc359443923)

[Código 17](#_Toc359443924)

[Esquema detalhado 18](#_Toc359443925)

[Algoritmo com o uso da condição “ do while” 19](#_Toc359443926)

[Fluxograma 19](#_Toc359443927)

[Código 19](#_Toc359443928)

[Esquema detalhado 20](#_Toc359443929)

[Algoritmo com o uso de uma função 21](#_Toc359443930)

[Fluxogramas 21](#_Toc359443931)

[Código 21](#_Toc359443932)

[Esquema detalhado 22](#_Toc359443933)

# Nota Geral:

Devido à especificação da linguagem, a tradução só é possível depois de ser executado o fluxograma.

# Algumas notas sobre Lua:

* É case sensitive.
* As funções são habitualmente definidas na parte inicial do programa, antes das instruções que farão a execução principal do programa.
* O código deve ser guardado num ficheiro com extensão .lua
* Para fazer comentários em Python utiliza-se --
  + Exemplo: --Comentário

# Estrutura Início

## Início:

Início

O Lua não tem equivalência ao Inicio.

# Estrutura Fim

## Fim:

Fim

O Lua não tem equivalência ao Fim.

# Variáveis

## Equivalência entre TIPOS de variáveis

|  |  |  |
| --- | --- | --- |
| TIPO | Portugol | Python |
| Inteiro | Inteiro | Number |
| Real | Real | Number |
| Texto | Texto | String |
| Lógico | Logico | boolean |

Tabela - Tipos de variáveis

**Nota:** Lua não diferencia valores inteiros de valores reais.

## Definição e atribuição de variáveis

Nota: Ao utilizar variáveis em Lua não é necessário informar o tipo desta variável como fazemos em outras linguagens.

# Estruturas input/output

## Input – Ler

variavel

io.write'<texto> '

variavel = io.read()

## Output – Escrever

expressao

Para escrever no ecrã:

Print '<texto>'

Ou

Print ( '<texto>',<variável>)

# Estruturas de Decisão

## Condição *“if”* e *“if else”*

condição

FALSE

TRUE

Instruções 2

Instruções 1

Para TRUE, escrever:

**if** <condição> **then**  
  
<instruções>  
  
**end**

Para FALSE:

Se Instruções 2 for igual a (conector) não fazer nada.

Senão, escrever:

**else**  
  
<instruções>

Exemplos práticos

#### Condição “if”

**if** <condição> **then**  
  
<instruções>  
  
**end**

#### Condição “if else”

io.write'Digite um número:'

n = io.read()

if n%2==0 then

print 'Numero Par'

else

print 'Numero Impar'

end

## Condição *“while”*

condição

true

false

instruções

**while** <condição> **do**  
  
<instruções>  
  
**end**

**Nota:** Em Lua o ciclo While é composto sempre pelo ‘do’.

## Condição *“do while”*

instruções

TRUE

condição

FALSE

Instruções

Nota: Não tem equivalência

### Exemplos práticos

#### Condição “while”

i=1

while i<=10 do

print(i)

i=i+1

end

#### Condição “do while”

i=0

repeat

io.write'Digite um número: '

i = io.read()

until tonumber(i)>=0

# Estrutura Conector

## Conector

Se for uma condição “*do while*” escrever:

repeat

# Funções

## Definir funções

Exemplo( a , b, . . .)

### Definir função

**Function**<nome\_função>

<Instruções>

**end**Chamada de funções

Function(Nome)

NOME(PARAMETRO);

### Exemplos do uso de funções

function fact(k)

if tonumber(k) > 2 then

return k \* fact(k - 1)

else

return k

end

end

i=5

j=fact(i)

print(j)

# Estrutura de retorno

## Return

expressao

return expressao;

# Operadores

### Aritméticos

|  |  |  |
| --- | --- | --- |
| Nome | Portugol | Lua |
| Adição | a + b | a + b |
| Subtração | a – b | a - b |
| Divisão | a / b | a / b |
| Multiplicação | a \* b | a \* b |
| Resto da divisão inteira |  | a % b |
| Potenciação |  | ^ |

Tabela - Equivalência de operadores aritméticos

### Lógicos

|  |  |  |
| --- | --- | --- |
| Nome | Portugol | Lua |
| Disjunção | a && b | and |
| Conjunção | a || b | or |
| Negação |  | not |

Tabela - Equivalência de operadores lógicos

### Relacionais

|  |  |  |
| --- | --- | --- |
| Nome | Portugol | Lua |
| Igual | a == b | a == b |
| Diferente | a != b | a ~= b |
| Maior | a > b | a > b |
| Maior ou igual | a >= b | a >= b |
| Menor | a < b | a < b |
| Menor ou igual | a <= b | a <= b |

Tabela - Equivalência de operadores relacionais

# ***ANEXO***

Para uma compreensão mais abrangente do uso das estruturas, ficam alguns exemplos mais extensivos, com o uso de várias estruturas em algoritmos completos.

## Algoritmo com o uso da condição “if”

**Problema:** Verificar se um número introduzido pelo utilizador é par.

### Fluxograma

Inicio

n

n%2=0

TRUE

FALSE

“Par”

Fim

### Código

io.write'Digite um número:'

n = io.read()

if n%2==0 then

print 'Par'

end

### Esquema detalhado



**Sem Correspondência**

**io.write'Digite um número:'**

**n = io.read()**

**if n%2==0 then**

**print 'Par'**

**Sem Correspondência**

**End**

## Algoritmo com o uso da condição “if else”

**Problema:** Verificar se um número introduzido pelo utilizador é par ou ímpar.

### Fluxograma

### 

TRUE

Fim

“Par”

n%2=0

n

Inicio

FALSE

“Impar”

### Código:

io.write'Digite um número:'

n = io.read()

if n%2==0 then

print 'Numero Par'

else

print 'Numero Impar'

end

### Esquema detalhado

**Sem Correspondência**

**io.write'Digite um número:'**

**n = io.read()**

**if n%2==0 then**

**else:**

**print 'Numero Impar'**

**print 'Numero Par'**

**Sem Correspondência**

**End**

“Impar”

FALSE

TRUE

Fim

“Par”

n%2=0

n

Inicio

## Algoritmo com o uso da condição “while”

**Problema:** Escrever um número de 1 a 10.

### Fluxograma

Inicio

i=1

i <= 10

FALSE

TRUE

i

i=i + 1

Fim

### Código

i=1

while i<=10 do

print(i)

i=i+1

end

### Esquema detalhado

**Sem Correspondência**

**i= 1**

**while i<=10 do**

**print(i)**

**i=i+1**

**End**

i=i + 1

i=1

Fim

i

i <= 10

Inicio

FALSE

TRUE

## Algoritmo com o uso da condição “ do while”

**Problema:** Pedir um número positivo.

### Fluxograma

Inicio

i = 0

i

TRUE

i < 0

FALSE

Fim

### Código

i=0

repeat

io.write'Digite um número: '

i = io.read()

until tonumber(i)>=0

### Esquema detalhado

**Sem Correspondência**

**i=0**

**repeat**

**io.write'Digite um número:'**

**i = io.read()**

**until tonumber(i)>=0**

**Sem Correspondência**

Inicio

i = 0

i

TRUE

i < 0

FALSE

Fim

## Algoritmo com o uso de uma função

**Problema:** Factorial de um número.

### Fluxogramas

**Código principal**

**Função fact(k)**

fact(k)

Inicio

TRUE

FALSE

k > 2

i = 5

j = fact ( i )

k

k \* fact(k - 1)

j

Fim

Fim

### Código

function fact(k)

if tonumber(k) > 2 then

return k \* fact(k - 1)

else

return k

end

end

i=5

j=fact(i)

print(j)

### Esquema detalhado

**Sem Correspondência**

**i=5**

**j=fact(i)**

**print (j)**

**Sem Correspondência**

Inicio

i = 5

j = fact ( i )

j

Fim

**def fact(k):**

**if tonumber(k)>2 then**

**return (k \* fact(k - 1) )**

**else:**

**return k**

**End**

