Linguagem de Programação: Lua

1. Elementos Sintáticos

1.1 Comandos

Lua oferece um conjunto de comandos convencionais, semelhantes às linguagens Pascal e C, que incluem:

- Atribuições
- Estruturas de Controle
- Chamadas de Função
- Declarações de Variáveis

1.2 Trechos

Um trecho é definido como uma sequência de comandos executados sequencialmente, podendo ser seguido por um ponto-e-vírgula (;).

Definição Sintática:

trecho -> comando ";"

1.3 Blocos

Blocos são listas de comandos.
Sintaticamente, um bloco é a mesma coisa que um trecho.

Definição Sintática:

bloco -> trecho

Um bloco pode ser explicitamente delimitado para produzir um único comando:

comando -> **do** bloco **end**

Notas:

- Blocos controlam o **escopo de variáveis.**
- Permitem a **adição de comandos** como **return** ou **break** no código.

1.4 Atribuições

Lua suporta atribuições múltiplas, onde uma lista de variáveis recebe valores de uma lista de expressões, separadas por vírgulas.

Definição Sintática:

comando -> listavar "=" listaexp
listavar -> var {"," var}
listaexp -> exp {"," exp}

Antes da atribuição ser realizada, a lista de valores é *ajustada* para o comprimento da lista de variáveis. Se há mais valores do que o necessário, os valores em excesso são descartados.

1.5 Estruturas de Controle

As estruturas de controle **if**, **while** e **repeat** possuem o significado usual e a sintaxe familiar.

Definição Sintática:

comando -> **if** exp then bloco {**elseif** exp then bloco} [else bloco] end

1.5.1 If

Inicia-se com **if** e uma expressão, com um bloco delimitado por then e finaliza com end. Pode incluir elseif e else.

1.6 Comando For

O comando **for** possui duas variações: uma **numérica** e outra **genérica**.

1.6.1 Variação Numérica

Inicia com for, seguido por:

- Nome da variável
- Palavra reservada =
- Expressões inicial, final e, opcionalmente, passo.

Bloco delimitado por do e end.

Definição Sintática:

comando -> **for** nome `=´ exp `**,**´ exp [`**,**´ exp] **do** bloco **end**

1.7 Expressões

As expressões em Lua abrangem diversos padrões sintáticos.

Definição Sintática:

exp -> expprefixo
| **nil | false | true**
| Numero
| Cadeia
| funcao
| "..."
| exp opbin exp
| opunaria exp
expprefixo -> var | chamadadefuncao | "(" exp ")"

Tipos de Expressões:

- **Primitivas**:nil,false,true.
- **Numéricas**: números.
- **Textuais**: cadeias de caracteres.
- **Funções**: padrão funcao.

- **Tabelas**: construtores conforme construtortabela.
- **Operações Unárias**: padrão opunaria exp.
- **Operações Binárias**: padrão exp opbin exp.
- **Prefixos**: variáveis, chamadas de função ou expressões entre parênteses.

1.7.1 – Operadores Aritméticos

Lua possui os operadores aritméticos padrão: +, -, *, /, ^, % (pertencem a opbin) e o unário -(não pertence a opbin).

Aceita números e strings numéricas.
Aceita qualquer expoente (ex: x^(-0.5) → inverso da raiz de x).
O módulo é definido por a % b == a - math.floor(a/b)*b, ou seja, resto da divisão arredondada para menos infinito.

1.7.2 – Operadores Relacionais

São: <, >, <=, >=, ~=, ==, (todos pertencem a opbin)

Retornam sempre **true** ou **false**.

A igualdade compara tipos antes dos valores — tipos diferentes sempre resultam em **false**.
Operadores de ordem (<, >, <=, >=) comparam números entre si e strings entre si; caso contrário, chamam metamétodos lt e le.

1.7.3 – Operadores Lógicos

São **and, or** (pertencem a opbin) **e not** (não pertence a opbin)**.**
Somente **false e nil** são considerados falsos.

- **not** retorna sempre true ou false.
- **and** retorna o primeiro operando falso ou o segundo se o primeiro for verdadeiro.
- **or** retorna o primeiro valor verdadeiro ou o segundo se o primeiro for falso.
  Usam avaliação de curto-circuito.

1.7.4 – Concatenação

O operador de concatenação é ’ .. ’ (pertence ao opbin)
Se ambos os operandos são números ou strings, são convertidos para string.

1.7.5 – Precedência

A precedência de operadores, da menor prioridade para a maior:

Or, and, <, >, <=, >=, ~=, ==, .., +, -, *, /, %, not, #, - (unary), ^

Parênteses mudam as precedências de uma expressão. Os operadores de concatenação ('..') e de exponenciação ('^') são associativos à direita.

1.7.6 – Chamadas de Função

Definição Sintática:

chamadadefuncao -> expprefixo args

Argumentos da função:

Args -> f‘cadeia’

Todas as expressões fornecidas como argumento são avaliadas antes da chamada. Uma chamada da forma f'cadeia' (ou f"cadeia" ou f[[cadeia]]) é um açúcar sintático para f('cadeia'); ou seja, a lista de argumentos consiste somente em uma cadeia de caracteres literal.

Referências

- **Manual de Referência de Lua 5.1**. Disponível em: <https://www.lua.org/manual/5.1/pt/manual.html#2.4>. Acesso em: 19 jan. 2026.