|  |  |
| --- | --- |
| Restrições contextuais – “LUA” | |
| 1 | A linguagem é *case sensitive*. |
| 2 | Todos os identificadores precisam ter sido “declarados” (algum valor/definição) antes de serem utilizados. Como Lua não possui declaração de variáveis, o primeiro uso de um identificador será sua “declaração”. Os demais aparecimentos são atribuições. |
| 3 | Em A = B (*assignmentCommand*), A deve ser uma variável. |
| 4 | Em A = B (*assignmentCommand*), o tipo de B precisa ser igual ao tipo de A. |
| 5 | Variáveis não sofrem mudança de tipo. Se inicialmente uma variável recebe um *int/float*, seu tipo será esse até o fim da execução do programa. |
| 6 | Todos os operadores devem ser aplicados a operandos do mesmo tipo. |
| 7? | Operadores +, -, \*, /, ==, ~=, >, <, >=, <= devem ser aplicados a operandos *int* ou *float*.  Mas só temos esses tipos... então vale colocar isso? |
| 8 | Operadores ==, ~=, >, <, >=, <= retornam valor de tipo int. |
| 9 | Operadores +, -, \* e / retornam o tipo dos seus operandos. |
| 10 | Não pode haver mais de uma função com o mesmo *spelling*. |
| 11 | Não pode haver mais de um procedimento com o mesmo *spelling*. |
| 12 | *Return* só pode ser utilizado dentro do escopo de uma função. |
| 13 | *Return* deve ser o último comando de uma *functionDeclaration.* |
| 14 | Uma função só pode ser chamada se declarada anteriormente. |
| 15 | Um procedimento só pode ser chamado se declarado anteriormente. |
| 16 | O número de argumentos passados na chamada de uma função deve ser igual ao número de parâmetros da função associada. |
| 17 | O número de argumentos passados na chamada de um procedimento deve ser igual ao número de parâmetros da função associada. |
| 18 | A expressão condicional de um *if* precisa ser do tipo inteiro. |
| 19 | A expressão condicional de um *while* precisa ser do tipo inteiro. |
| 20 | *Break* só pode ser utilizado dentro do escopo de um *while*. |

**Considerações:**

1. Existem variáveis globais e variáveis locais
2. Para os casos que foi necessário enviar o *obj* pelo método *visitor*, o objeto enviado foi um *arrayList* de inteiros, onde:

Posição 0 do array ~ Passar por uma *functionDeclaration*

Posição 1 do array ~ Passar por um *while.*

1. Valores Booleanos são considerados inteiros, onde: 0~*false* e qualquer outro número~*true*.
2. Funções e procedimentos são declarados através da mesma estrutura na gramática. A diferença é a presença do retorno. Já a chamada de uma função é diferente de uma chamada de procedimento.

**Observações da linguagem:**

1. O programador deve cuidar de argumentos que não são dos mesmos tipos dos parâmetros em Lua. Se por exemplo a seguinte função for declarada:

*function soma (x,y) print (x+y) end*O resultado para soma (1,2) = 3O resultado para soma (1, “4”) = 5, onde “4” significa uma String

O resultado para soma (“4”, “3”) = 7  
  
Logo, o tipo não é um fator fixo. Se o corpo da função permite sua aplicabilidade em diversos tipo, vai funcionar para esses tipos em questão.

**? Para verificar se tem ou não um certo item que é uma lista... verifica se ele é null ou se é vazio?? Se for vazia ela entra no for sem problema, né?**