|  |  |
| --- | --- |
| 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, a primeira atribuição de um identificador será sua “declaração”. Os demais aparecimentos são atribuições. |
| 3 | Em A = B (*assignmentCommand*), A deve ser uma variável. |
| 4 | 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. |
| 5 | Todos os operadores devem ser aplicados a operandos do mesmo tipo. |
| 6 | Operadores +, -, \*, /, ==, ~=, >, <, >=, <= devem ser aplicados a operandos *int* ou *float*. |
| 7 | Operadores ==, ~=, >, <, >=, <= retornam valor de tipo int. |
| 8 | Operadores +, -, \* e / retornam o tipo dos seus operandos. |
| 9 | Não pode haver mais de uma função com o mesmo *spelling*. |
| 10 | Não pode haver mais de um procedimento com o mesmo *spelling*. |
| 11 | Uma função só pode ser chamada se declarada anteriormente. |
| 12 | Um procedimento só pode ser chamado se declarado anteriormente. |
| 13 | O número de argumentos passados na chamada de um procedimento deve ser igual ao número de parâmetros da definição do mesmo. |
| 14 | O número de argumentos passados na chamada de uma função deve ser igual ao número de parâmetros da definição da mesma. |
| 15 | Toda expressão de condição deve ser do tipo inteiro. (condição do *if* e *while*) |
| 16 | *Break* só pode ser utilizado dentro do escopo de um *while*. |
| 17 | Um parâmetro não pode ter seu valor alterado, somente lido. |
| 18 | Não é possível ter uma variável e função com mesmo nome. |

**Considerações:**

1. Existem variáveis globais e variáveis locais
2. Valores Booleanos são considerados inteiros, onde: 0~*false* e 1~*true*.
3. 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.
4. Como o *return* está definido pela gramática para aparecer no final de uma função, a verificação de sua localização é responsabilidade sintática e não semântica.