Trabalho GA

Criar uma linguagem de expressões. O objetivo principal é definir uma linguagem que suporte expressões aritméticas, relacionais, lógicas e funções (tal como as funções suportadas no Excel).

Tipos de expressões suportadas

Na forma geral, uma expressão é definida como um das seguintes opções (notação BNF - *Backus Naur Form* ou *Backus Normal Form*):

```
<expressao> ::= <expressao> <operador> <expressao>
<expressao> ::= <operando>
<expressao> ::= <funcao>
<expressao> ::= <declaracao>

<declaracao> ::= <identificador> <operadorAtribuicao> <expressao>

<funcao> ::= <identificador_funcao> ( <expressao> [;<expressao>]* )
<funcao> ::= input ( <texto> ; <identificador>)
<funcao> ::= output (<expressao>)

<operando> ::= numero | texto | booleando | variavel
```

A ordem de precedência de tipos de expressão padrão é:

1. Expressão Aritmética
2. Expressão Relacional
3. Expressão Lógica

A ordem de predência de operadores aritméticos é:

1. Sinal
2. Potenciação
3. Multiplicação e divisão
4. Soma e subtração

> OBS.: Considere incorporar operador artimético para potenciação.

A ordem de predência de operadores relacionais é definida apenas pela ordem de escrita, da esquerda para a direita.

A ordem de predência de operadores lógicos é:

1. Negação
2. E lógico
3. OU lógico, OU exclusivo (XOR)

> OBS: o operador () pode modificar a ordem de precedência das operações.

Deve ser possível executar expressões como (considerando uma linguagem como Java):

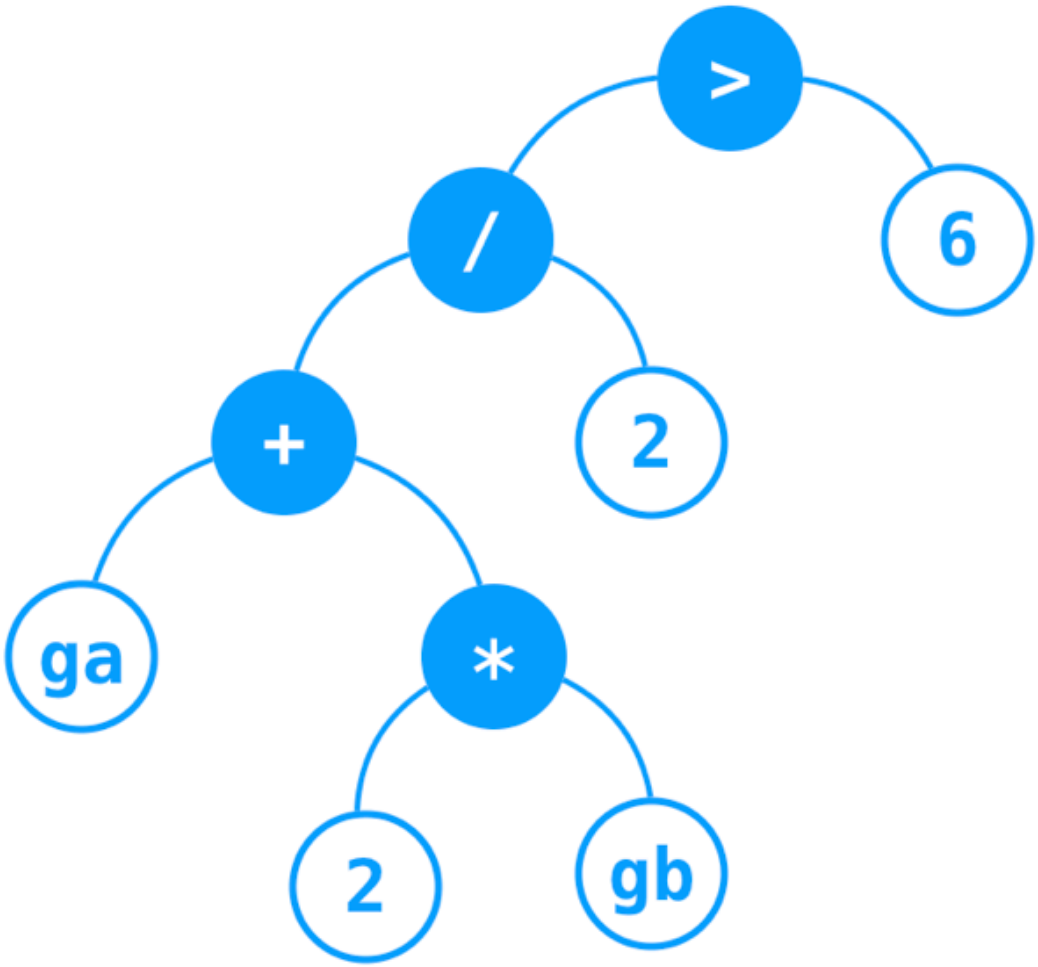
```
i / W * Sw
(ga + 2 * gb) / 3
!chuvendo || chuvendo && tenhoGuardaChuva
(ga + 2 * gb) / 2 > 6
```

Definição

- Defina operadores, palavras reservadas e a regra de composição de todas as estruturas de expressão necessárias para analisar uma expressão e permitir execução.
- Defina operações especiais, além do básico solicitado para as expressões.
- Sugestões de funções que para suporte na linguagem: INPUT, OUTPUT, IF, MAX, MIN, AVG e SUM. Defina outras operações interessantes.

Análise sintática e execução

- Desenvolver um software interpretador da linguagem de expressão definida.
- O interpretador deve receber uma string com a expressão e montar uma árvore de expressão. Depois, a partir da árvore de expressão, executar a expressão informada.
- Deve ser possível acessar posições de memória (variavéis).
    - Considere que um variável é representada por instrução de declaração com inicialização.
- Deve ser possível passar um arquivo texto com expressões para serem executadas linha-a-linha. Dessa forma, é possível declarar variáveis e executar expressões.
- Exemplo de árvore de expressão para a expressão `(ga + 2 * gb) / 2 > 6` na linguagens Java/JavaScript/C++/C#:



Exemplo de código numa linguagem hipotética

```
A = 0                   // declaraçao de A
B = 0                   // declaraçao de B
input('Digite GA';A)    // função input
input('Digite GB';B)    // função input
output((A + 2 * B) / 3) // função output com um parâmetro que é uma expressão

// exemplo de análise sintática (arvore de expressão)
// exp3 => exp2 / 3
// exp2 => A + exp1
// exp1 => 2 * B
```

Avaliação e entrega

- O trabalho vale 70% da nota do GA.
- Pode ser desenvolvido em grupo, o mesmo grupo formado para o trabalho de avaliação de linguagens.
- A entrega deve ser feita no Moodle, na tarefa especifica e contendo:

- Arquivo compactado no formato .zip com os códigos-fonte da implementação.
- Documentação explicando o uso da linguagem e do interpretador.
- Vídeo no formato MP4, com explicação da definição da linguagem, explicação da implementação e execução do interpretador.
- Duração do vídeo é de 15min.