Reconhecedor de gramáticas

A maior aplicação das gramáticas livres de contexto ocorre na **formalização sintática** das linguagens de programação de alto nível. A característica que torna as gramáticas livres de contexto especialmente adequadas à formalização sintática das linguagens de programação é a sua capacidade de representação de **construções aninhadas**, que são freqüentemente encontradas em linguagens dessa categoria;

As **construções aninhadas** costumam ocorrer em linguagens de programação, por exemplo, na construção de **expressões aritméticas**, em que subexpressões são delimitadas, através do uso de parênteses.

Para exemplificar isto, considere uma pequena gramática (escrita na notação BNF – *Backus-Naur Form*) para definição de variáveis, comandos de atribuição com expressões aritméticas.

```
<programa>    ::= <declarações> <comandos>
<declarações> ::= <declaração> | <declaração> <declarações>
<declaração>  ::= int <variavel>;
<comandos>    ::= <comandos> ; <atribuição> | <atribuição>
<atribuição>  ::= <variavel> = <expr>
<expr>        ::= <expr> + <termo> |
                  <expr> - <termo> |
                  + <termo> |
                  - <termo> |
                  <termo>
<termo>       ::= <termo> * <fator> | <termo> / <fator> | <fator>
<fator>       ::=  <numero> |
                   (<expr>)
<variável>    ::= <letra> <variável> | <letra>
<numero>      ::= <digito> <número> | <digito>
<letra>       ::= a | b | c | d | e | f | g | h | i | j | k | l |
                  m | n | o | p | q | r | s | t | u | v | w | x |
                  y | z
<digito>      ::= 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9
```

Considere que o símbolo inicial da gramática é `<programa>` e os `<não-terminais>` são nomes entre parênteses angulares < e > e os símbolos **`terminais`** estão em **`negrito`**.

Objetivo

O objetivo do trabalho é implementar um **Reconhecedor** para gramática acima, para tanto a gramática não pode ter recursividade à esquerda e deve estar fatorada, assim você deve rescrever as produções, eliminando a recursividade à esquerda, para que seja possível implementar o reconhecedor, ou seja, reescrever a gramática para notação EBNF (BNF estendida).

A entrada do Reconhecedor será realizada por um **arquivo texto** com um trecho de código seguindo a gramática acima, como por exemplo:

Um exemplo de código (em arquivo) pode ser visto abaixo:

```
int
 x;      int soma;

x= 2 + 3; // aqui eh um comentario
soma = 3*8
```

Note que a entrada pode constar **comentários** de uma linha (**//**) e **caracteres delimitadores** (espaços em branco, quebra de linhas, tabulação e retorno de carro) que devem ser desconsiderados na análise. A saída do reconhecedor será na tela do computador, e dever informar se foi possível derivar a sentença para gramática do trabalho e a quantidade de linhas do arquivo processada, por exemplo para o exemplo acima teremos a seguinte saída:

```
Entrada aceita sem erros.
5 linhas analisadas
```

Se a entrada não estiver sintaticamente correta, ou seja, caso não seja possível derivar a sentença a partir da gramática, como no exemplo abaixo:

```
int x;
int soma;

x=(2+((3)); // aqui eh um comentário, mas tem um erro na expressao
soma = 3*8
```

Deve ser apresentada uma mensagem **informando a linha em que ocorreu o erro e o reconhecedor finaliza o processo de análise**, por exemplo para a entrada acima temos:

```
Falha na derivacao
Erro na linha 4.
```

Observações importantes:

O programa deve estar bem documentado e pode ser feito em grupo de até **2 alunos**, não esqueçam de colocar o **nome dos integrantes** do grupo no arquivo fonte do trabalho e sigam as **Orientações para Desenvolvimento de Trabalhos Práticos** disponível no **Moodle**.

O trabalho será avaliado de acordo com os seguintes critérios:

- Funcionamento do programa, caso programa apresentarem ***warning*** ao serem compilados serão penalizados. Após a execução o programa deve finalizar com **retorno igual a 0**;
- O trabalho deve ser desenvolvido na **linguagem C** e será testado usando o compilador do **CodeBlocks 17.12**.
- O quão fiel é o programa quanto à descrição do enunciado, principalmente ao formato de do **arquivo de entrada**;
- Clareza e organização, programas com código confuso (linhas longas, variáveis com nomes não-significativos, ...) e desorganizado (sem indentação, sem comentários, ...) também serão penalizados.