


1COP020 - Lista de Exercícios 14

1.  **Exercício Prático:** Utilize as ferramentas *flex* e *bison* para criar um *parser* de expressões matemáticas. As expressões podem ser formadas por:

operadores binários: + - * / ^ %

operadores unários: + -

delimitadores: ()

funções: sen cos tan abs

números inteiros

variáveis: podem ser formadas por letras minúsculas e números mas devem sempre começar por uma letra.

números reais: a parte inteira e a parte fracionária são separadas por um ponto. São números reais válidos:

3.14

.666

4.

0.0

Exemplos de expressões válidas são:

sen(x)

-x*sen(x)

abc+78

-10*45^2

(1+3)*78

17.78+3

(dim0+dim1)*fatormultiplicativo3

(sen(x))^2+(cos(x))^2

As cadeias de entrada para o seu programa estarão armazenadas em um arquivo, sendo que haverá uma cadeia por linha. Espaços em branco e quebras de linha devem ser removidas pelo analisador léxico sem acusar erro. *Tokens* considerados inválidos devem ser indicados como erro pelo analisador léxico e a análise sintática da cadeia em questão não é realizada.

O arquivo contendo as cadeias deve ser lido da entrada padrão e a saída do programa deve ser impressa na saída padrão. Supondo que o arquivo de entrada se chame `cadeias.txt` e que o programa executável se chame `l14e1`, a execução será da seguinte forma:

```
./l14e1 < cadeias.txt
```

Crie a sua gramática de forma a respeitar as precedências dos operadores matemáticos. Se a expressão matemática estiver correta, o seu programa deve imprimir a mensagem:

EXPRESSAO CORRETA

Se expressão matemática contiver um erro de sintaxe, o seu programa deve indicar em que ponto ocorreu o erro, bem como o *token* causador do erro. Considere a seguinte entrada:

`()`

A seguinte saída deve ser gerada:

Erro de sintaxe na coluna [2]:)

Observe que deve ser indicada a coluna onde ocorreu o erro. No exemplo apresentado, o erro aconteceu na coluna 2 e foi causado pelo *token* `)`

Considere agora a seguinte entrada:

`+3*`

A seguinte saída deve ser gerada:

A expressao terminou de forma inesperada.

Observe que no exemplo apresentado, a entrada é constituída por uma cadeia sintaticamente correta até onde está construída, mas que terminada de forma abrupta. Esse término abrupto não deixa de ser um erro sintático, mas deve ser reportado de forma específica.

Em uma cadeia, quando houver erro léxico, todos os caracteres inválidos que aparecerem na mesma devem ser apresentados. É importante salientar que nenhuma forma de análise sintática deve ser realizada em uma cadeia que contiver um erro léxico, isto é, caracteres inválidos. Como exemplo, considere a seguinte entrada:

`#3+3@$1+1!=`

A seguinte saída deve ser gerada:

Caractere(s) invalido(s) -> [#,@,\$,!,=]

O número de mensagens geradas deve ser igual a quantidade de cadeias de entrada. Considere que o seu programa recebeu a seguinte entrada, a qual contém 10 linhas:

`+666*1`

`-sen cos`

`+666-`

`#3+3@$1+1!=`

`@1+1`

`+666^.666`

`*-@123#$`

`(())*-@123#$`

`(())`

O seu programa deve gerar a seguinte saída, a qual também contém 10 linhas:

```
EXPRESSAO CORRETA
Erro de sintaxe na coluna [6]: cos
A expressao terminou de forma inesperada.
A expressao terminou de forma inesperada.
Caractere(s) invalido(s) -> [#,@,$,!]=]
Caractere(s) invalido(s) -> [@]
EXPRESSAO CORRETA
Caractere(s) invalido(s) -> [@,#,$]
Caractere(s) invalido(s) -> [@,#,$]
Erro de sintaxe na coluna [3]: )
```