CENTRO FEDERAL DE EDUCAÇÃO TECNOLÓGICA DE MINAS GERAIS
DEPARTAMENTO DE ENGENHARIA DA COMPUTAÇÃO

VITOR BRANDÃO RAPOSO

FERNANDO FARIA SOARES

COMPILADORES

Trabalho 1: Analisador Léxico & Tabela de Símbolos

BELO HORIZONTE - MG

2022

1 INTRODUÇÃO

Em nosso primeiro trabalho realizado para a disciplina de Compiladores, realizamos a de um analisador léxico simples com o esqueleto de uma tabela de símbolos que será futuramente utilizada pelo analisador sintático. O objetivo dessa parte do trabalho é que o analisador léxico criado por nós consiga ler os lexemas (conjunto de caracteres) e identificar os tokens gerados com suas respectivas TAGs. Os tokens são uma parte essencial para o funcionamento de um compilador, pois irão reconhecer o que está no código e determinar como o programa irá ser conduzido.

2 METODOLOGIA

Baseamo-nos fortemente nas ideais utilizadas no projeto de Front-end completo disponibilizado ao final do livro "*Compiladores: Princípios, técnicas e ferramentas*" utilizado em nossa disciplina.

Assim, nossa aplicação e divida nas seguintes classes:

- **Toplevel**: classe main responsável por rodar o programa;
- **Lexer**: responsável por iniciar o analisador léxico;
- **Token**: classe estrutural de Token;
- **Tag**: responsável por listar todas as categorias de tokens;
- **Num**: classe estrutural de números;
- **Word**: classe estrutural das palavras reservadas e símbolos com múltiplos caracteres
- **Env**: responsável por criar e manter as múltiplas instâncias de tabelas de símbolos, denominadas environments;
- **Id**: classe estrutural dos identificadores;
- **Type**: classe estrutural de categorias de variáveis (char, int, float);
- **Expr**, **Array**, **Node**: classes associadas a tabela de símbolo criadas para futuramente apoiar o analisador sintático. Atualmente nao estamos utilizando para fins práticos.

3 TESTES

Para o realizar a compilação do Analisador Léxico, é necessário mudar o caminho o qual encontra os testes no computador, nesse caso é:

C:/Coding/Compiladores/Analisador_Lexico/teste1

A modificação tem que ser feita no Top-level.java, na parte em negrito:

“

```
public static void main(String[] args) throws IOException {

    Token currentToken;
    Lexer lexer = new Lexer("C:/Coding/Compiladores/Analisador_Lexico/teste1");

    System.out.println("Analise Lexica:");
    System.out.println("-----------------"); “
```

Abaixo estão os testes, os quais possuem a saída de acordo com a TAG do token. Por exemplo, sabemos que a tag da palavra reservada “routine” é 256. Na criação de um token, após identificá-lo de acordo com o conjunto de caracteres (lexema), direcionamos uma TAG para ele. No primeiro teste, o analisador léxico lê os caracteres, forma o lexema “routine” e mostra qual a sua TAG de acordo com a tabela de símbolos já criada.

Teste 1:

```
Analise Lexica:
-----------------
routine : TOKEN.256
declare : TOKEN.259
int : TOKEN.260
a : TOKEN.282
301 : TOKEN.301
b : TOKEN.282
300 : TOKEN.300
int : TOKEN.260
resul : TOKEN.282
300 : TOKEN.300
float : TOKEN.261
a : TOKEN.282
301 : TOKEN.301
x : TOKEN.282
300 : TOKEN.300
begin : TOKEN.257
a : TOKEN.282
:= : TOKEN.302
12 : TOKEN.286
a : TOKEN.282
300 : TOKEN.300
x : TOKEN.282
:= : TOKEN.302
12 : TOKEN.286
300 : TOKEN.300
read : TOKEN.270
298 : TOKEN.298
a : TOKEN.282
```

```
Tabela de Simbolos:
-----------------
{Resultado=Resultado, int=int, not=not, end=end, until=until, or=or, repeat=repeat, x=x, result=result, resul=resul, if=if,
hen=then, char=char, c=c, declare=declare, begin=begin, b=b, a=a, float=float, and=and}
```

Teste 2:

```
Analise Lexica:
-----------------
routine : TOKEN.256
int : TOKEN.260
a : TOKEN.282
301 : TOKEN.301
b : TOKEN.282
301 : TOKEN.301
c : TOKEN.282
300 : TOKEN.300
float : TOKEN.261
d : TOKEN.282
301 : TOKEN.301
95 : TOKEN.95
var : TOKEN.282
begin : TOKEN.257
read : TOKEN.270
298 : TOKEN.298
a : TOKEN.282
299 : TOKEN.299
300 : TOKEN.300
b : TOKEN.282
:= : TOKEN.302
a : TOKEN.282
296 : TOKEN.296
a : TOKEN.282
300 : TOKEN.300
```

```
Tabela de Simbolos:
-----------------
{formula=formula, int=int, not=not, end=end, until=until, or=or, repeat=repeat, if=if, routine=routine, while=while, do=do,
, char=char, c=c, declare=declare, begin=begin, b=b, a=a, float=float, and=and, val=val}
```

Teste 3:

```
Analise Lexica:
-----------------
routine : TOKEN.256
declare : TOKEN.259
int : TOKEN.260
a : TOKEN.282
301 : TOKEN.301
aux : TOKEN.282
300 : TOKEN.300
float : TOKEN.261
b : TOKEN.282
300 : TOKEN.300
begin : TOKEN.257
B : TOKEN.282
:= : TOKEN.302
0 : TOKEN.286
300 : TOKEN.300
read : TOKEN.270
298 : TOKEN.298
a : TOKEN.282
299 : TOKEN.299
300 : TOKEN.300
read : TOKEN.270
298 : TOKEN.298
b : TOKEN.282
299 : TOKEN.299
300 : TOKEN.300
if : TOKEN.263
298 : TOKEN.298
a : TOKEN.282
291 : TOKEN.291
299 : TOKEN.299
```

```
Tabela de Simbolos:
-----------------
{Numeros=Numeros, int=int, not=not, end=end, until=until, or=or, repeat=repeat, if=if, B=B, routine=routine, while=while, do=do,
ar, declare=declare, begin=begin, b=b, aux=aux, a=a, float=float, and=and}
```

Teste 4:

```
Analise Lexica:
----------------
routine : TOKEN.256
declare : TOKEN.259
int : TOKEN.260
pontuacao : TOKEN.282
301 : TOKEN.301
pontuacaoMaxina : TOKEN.282
301 : TOKEN.301
disponibilidade : TOKEN.282
300 : TOKEN.300
char : TOKEN.262
pontuacaoMinima : TOKEN.282
300 : TOKEN.300
begin : TOKEN.257
pontuacaoMinima : TOKEN.282
61 : TOKEN.61
50 : TOKEN.286
300 : TOKEN.300
pontuacaoMaxima : TOKEN.282
61 : TOKEN.61
100 : TOKEN.286
300 : TOKEN.300
write : TOKEN.271
298 : TOKEN.298
306 : TOKEN.306
Pontuacao : TOKEN.282
do : TOKEN.269
candidato : TOKEN.282
307 : TOKEN.307
299 : TOKEN.299
300 : TOKEN.300
read : TOKEN.270
298 : TOKEN.298
pontuacao : TOKEN.282
```

```
Tabela de Simbolos:
-----------------
{int=int, disponibilidade=disponibilidade, candidato=candidato, pontuacaoMaxina=pontuacaoMaxina, pontuaÃ=pontuaÃ, not=not, D
ssamento, Candidato=Candidato, or=or, repeat=repeat, pontuacao=pontuacao, if=if, aprovado=aprovado, routine=routine, while=w
ead=read, pontuacaoMaxima=pontuacaoMaxima, else=else, write=write, Ã=Ã, then=then, char=char, declare=declare, begin=begin,
```

Teste 5:

```
Analise Lexica:
-----------------
declare : TOKEN.259
integer : TOKEN.282
a : TOKEN.282
301 : TOKEN.301
b : TOKEN.282
301 : TOKEN.301
c : TOKEN.282
301 : TOKEN.301
maior : TOKEN.282
300 : TOKEN.300
char : TOKEN.262
outro : TOKEN.282
300 : TOKEN.300
begin : TOKEN.257
repeat : TOKEN.266
write : TOKEN.271
298 : TOKEN.298
306 : TOKEN.306
A : TOKEN.282
307 : TOKEN.307
306 : TOKEN.306
299 : TOKEN.299
300 : TOKEN.300
read : TOKEN.270
298 : TOKEN.298
a : TOKEN.282
299 : TOKEN.299
300 : TOKEN.300
write : TOKEN.271
298 : TOKEN.298
306 : TOKEN.306
B : TOKEN.282
```

```
Tabela de Simbolos:
-----------------
{int=int, maior=maior, not=not, end=end, until=until, S=S, N=N, valor=valor, or=or, Maior=Maior, repeat=repeat, ou
d, Outro=Outro, integer=integer, else=else, write=write, then=then, char=char, c=c, declare=declare, begin=begin,
```

Teste 6:

```
Analise Lexica:
----------------
routine : TOKEN.256
declare : TOKEN.259
declare : TOKEN.259
int : TOKEN.260
pontuacao : TOKEN.282
301 : TOKEN.301
pontuacaoMaxina : TOKEN.282
301 : TOKEN.301
disponibilidade : TOKEN.282
300 : TOKEN.300
char : TOKEN.262
pontuacaoMinima : TOKEN.282
300 : TOKEN.300
begin : TOKEN.257
begin : TOKEN.257
int : TOKEN.260
ab1 : TOKEN.282
300 : TOKEN.300
float : TOKEN.261
bc2 : TOKEN.282
300 : TOKEN.300
end : TOKEN.258
end : TOKEN.258
308 : TOKEN.308
Comentario : TOKEN.282
307 : TOKEN.307
307 : TOKEN.307
305 : TOKEN.305
305 : TOKEN.305
300 : TOKEN.300
300 : TOKEN.300
308 : TOKEN.308
```

```
Tabela de Simbolos:
----------------
{int=int, disponibilidade=disponibilidade, pontuacaoMaxina=pontuacaoMaxina, Comentario=Comentario, not=not, end=end, until=until, c
while=while, pontuacaoMinima=pontuacaoMinima, do=do, read=read, else=else, write=write, bc2=bc2, then=then, char=char, declare=decl
```

Teste 7:

```
Analise Lexica:
----------------
routine : TOKEN.256
declare : TOKEN.259
int : TOKEN.260
amor : TOKEN.282
300 : TOKEN.300
while : TOKEN.268
298 : TOKEN.298
amor : TOKEN.282
299 : TOKEN.299
do : TOKEN.269
while : TOKEN.268
298 : TOKEN.298
amor : TOKEN.282
299 : TOKEN.299
do : TOKEN.269
while : TOKEN.268
298 : TOKEN.298
amor : TOKEN.282
299 : TOKEN.299
do : TOKEN.269
while : TOKEN.268
298 : TOKEN.298
amor : TOKEN.282
299 : TOKEN.299
do : TOKEN.269
end : TOKEN.258
300 : TOKEN.300
end : TOKEN.258
300 : TOKEN.300
end : TOKEN.258
300 : TOKEN.300
end : TOKEN.258
300 : TOKEN.300
end : TOKEN.258
```

```
Tabela de Simbolos:
----------------
{int=int, amor=amor, not=not, end=end, until=until, or=or, repeat=repeat, if=if, routine=routine, while=while, do=do, read=read,
=begin, float=float, and=and}
```