Aluna: Rafaelle Arruda
Professor: Braulio Mello

Data: 07/04/2021
Disciplina: Construção de Compiladores

Atividade Orientada

Em primeiro momento, criamos uma entrada.txt que informa o que é possível gerar atrás da gramática de forma léxica, de acordo com a imagem 1 abaixo.

```
se
ou_se
enquanto
escreve
na_tela
novov
quebra
=
==
!=
:
{
}
(
)
/
-
+
*
,
+ior
-ior

<S> ::= .<A> | 0<C>
<A> ::= a<A> | b<A> | c<A> | d<A> | e<A> | f<A> | g<A> | h<A> | i<A> | j<A> | k<A> | l<A> | m<A> |
        n<A> | o<A> | p<A> | q<A> | r<A> | s<A> | t<A> | u<A> | v<A> | w<A> | x<A> | y<A> | z<A> | ~<B>
<B> ::= $
<C> ::= 1<C> | 2<C> | 3<C> | 4<C> | 5<C> | 6<C> | 7<C> | 8<C> | 9<C> | 0<C> | $
```

Imagem 1. Entrada de dados de uma arquivo.txt

Em questão de implementação, a forma que os tokens podem ser adicionadas na tabela de símbolos. Começa com o seguinte passo, cada vez que encontrar um separador, ele encontrar uma palavra e então adicionar a tabela de símbolos informando a linha e o estado que reconhece esse token, como a imagem 2 a seguinte mostra um trecho da parte de um código que está sendo trabalhado.

```
def analisador_lexico_sintatico(self):
    tabela = self.pegarAutomato()
    fitaS = []
    Ts = []
    codigoFonte = list(open('codigo.txt'))
    separador = [' ', '\n', '\t']
    palavra = ''
    count = 0
    estado = 0
    for linha in codigoFonte:
        count += 1
        for caracter in linha:
            if caracter in separador and palavra:
                Ts.append({'Linha': str(count), 'Estado': str(estado), 'Rotulo': palavra.strip('\n')})
                estado = 0
                palavra = ''
            else:
                try:
                    estado = tabela[estado][caracter][0]
                except KeyError:
                    estado = -1
                if caracter != ' ':
                    palavra += caracter
```

Imagem 2. Código fonte do projeto.

Logo temos uma GLC que criada para atualizar no Parser e irá gera a tabela de parsing para a análise sintática que está na imagem 3 abaixo.

```
"Start Symbol" = <S>
<S> ::= 'novov' <VAR0>
<VAR0> ::= '.name.' '=' '0constant' <VAR0> | <OP>
<OP> ::= <LOGIC> <OP> | <PRINT> <OP> | <SCANF> <OP> | <>
<LOGIC> ::= 'se' '(' <OPEX> ')' '{' <EXP> '}' <ELSE> | 'enquanto' '(' <OPEX> ',' <INC> ')' '{' <EXP> '}'
<ELSE> ::= 'ou_se' '(' <OPEX> ')' '{' <EXP> '}' | <>
<SCANF> ::= 'escreve' '(' '.name.' ')' ':' '0constant'
<PRINT> ::= 'na_tela' '(' '.name.' ')' | 'na_tela' '(' '0constant' ')'
<OPEX> ::= <VAR> | <CONSTANT>
<VAR> ::= '.name.' '==' '.name.' | '.name.' '!=' '.name.' | '.name.' '+ior' '.name.' | '.name.' '-ior' '.name.'
<CONSTANT> ::= '.name.' '==' '0constant' | '.name.' '!=' '0constant' | '.name.' '+ior' '0constant' | '.name.' '-ior' '0constant'
<EXP> ::= '.name.' '=' <SIMB> '*' <SIMB> | '.name.' '=' <SIMB> '/' <SIMB> | '.name.' '=' <SIMB> '-' <SIMB> | '.name.' '=' <SIMB> '+' <SIMB> | 'quebra'
| <>
<SIMB> ::= '-' '.name.' | '.name.' | '-' '0constant' | '0constant'
<INC> ::= '.name.' '+' | '.name.' '-' | '.name.' '+' '0constant' | '.name.' '-' '0constant' | <>
```

Imagem 3. Tabela GLC