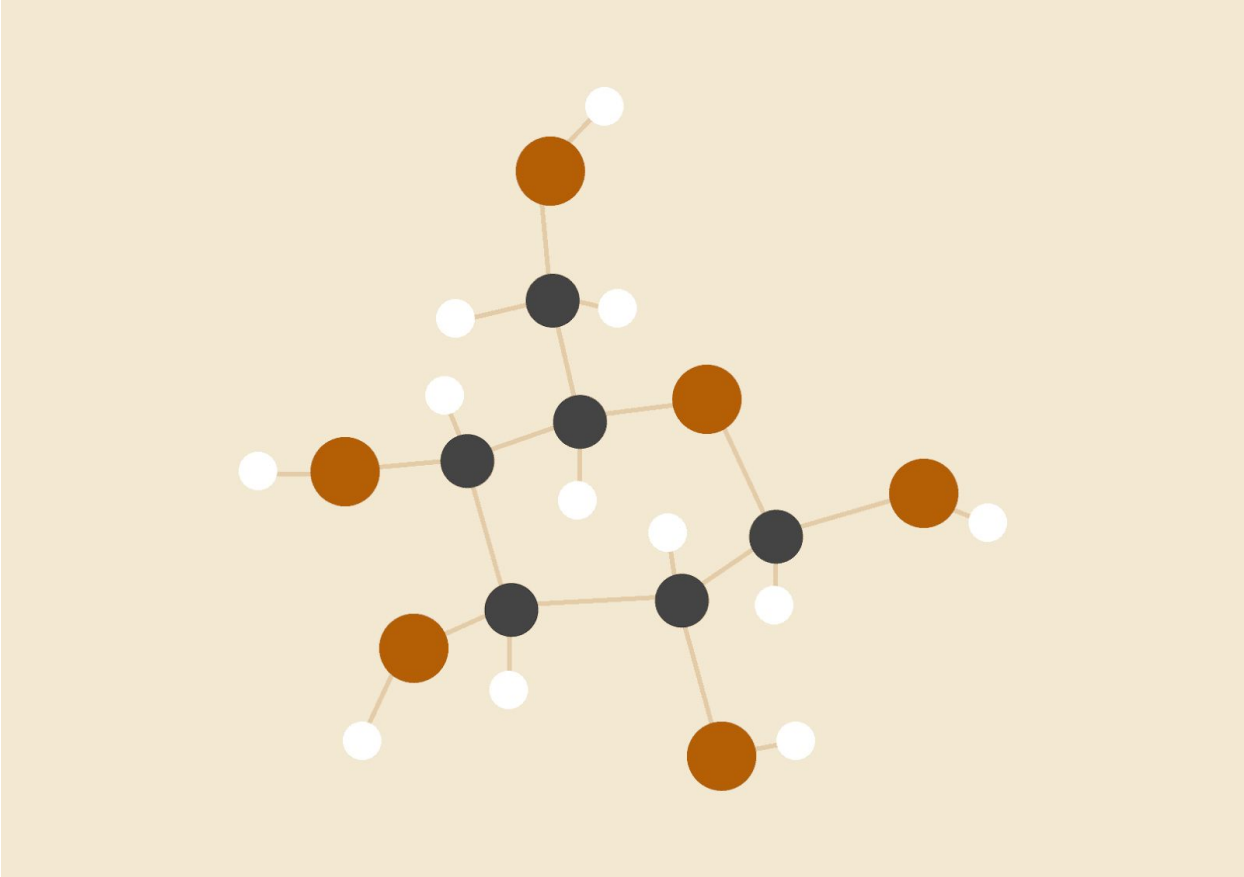
Relatório EP (AL-AS)

INE 5426



Bruno George de Moraes

Marcelo José Dias

Renan Pinho Assi

Vinicius Schwinden Berkenbrock

08.10.2019

Introdução

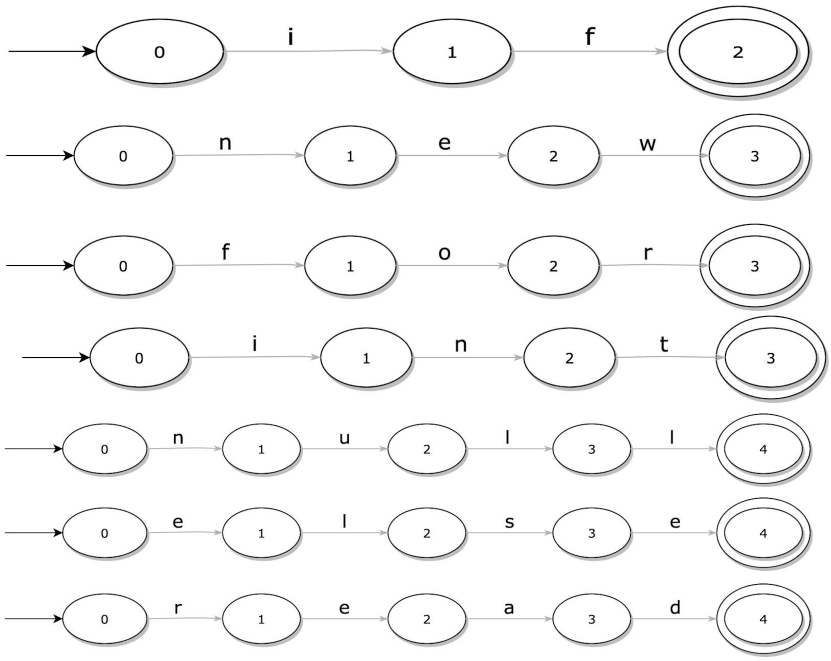
Relatório sobre Exercício-Programa (EP) na disciplina INE 5426, cujo objetivo é a construção de um Analisador Léxico e Sintático e responder algumas tarefas sobre a gramática `CC-2019-2`.

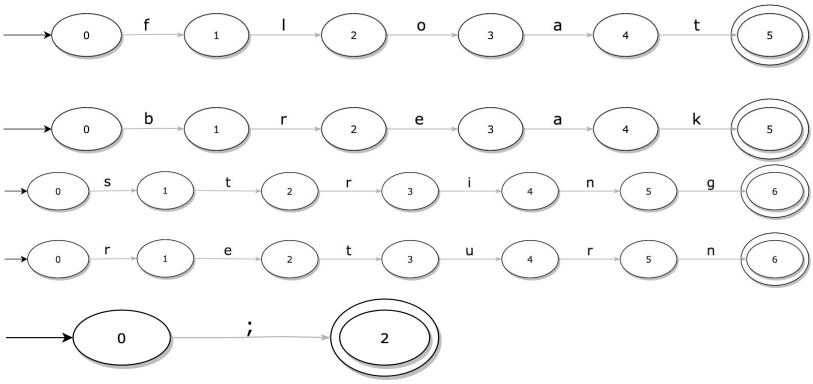
Respostas Analisador Léxico

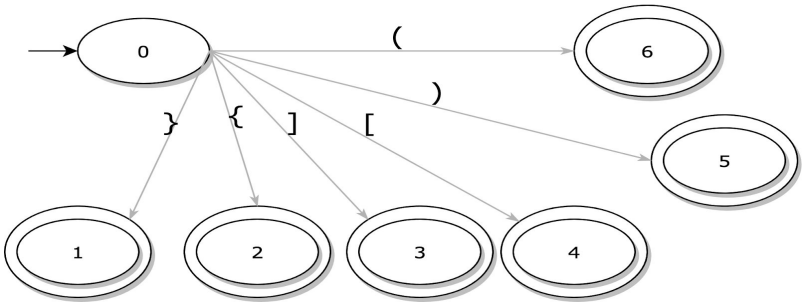
1) Descrição tokens:

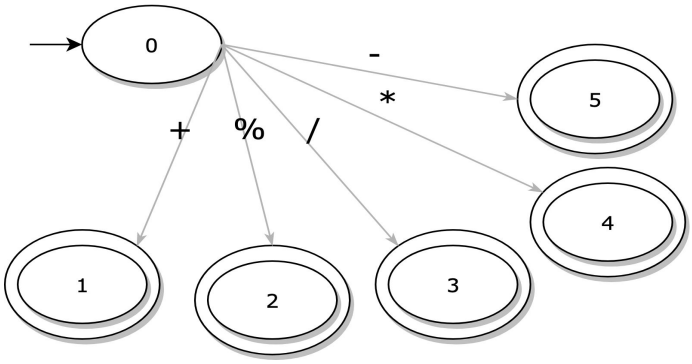
IDENT, // ID
OPENBRACE, CLOSEBRACE, OPENBRKT, CLOSEBRKT, OPENPARENT,
CLOSEPARENT, // {}[]()
INT, // int_constant
STR, // string_constant
FLT, // float_constant
INTEGER_T, FLOAT_T, STRING_T, NULL, // tipos: int, float, string, null
PRINT, RETURN, READ, // print, return, read
IF, ELSE, FOR, BREAK,
NEW, ASSERT, // new, =
LT, LE, EQ, GT, GE, NE, // < <= == > >= <>
ADD, MINUS, MULTIPLY, DIVIDE, MODULUS, // + - * / %
SEPARATOR, // ;
ERROR,
EMPTY, // Auxiliar pro sintatico
DOLLAR // fim pilha

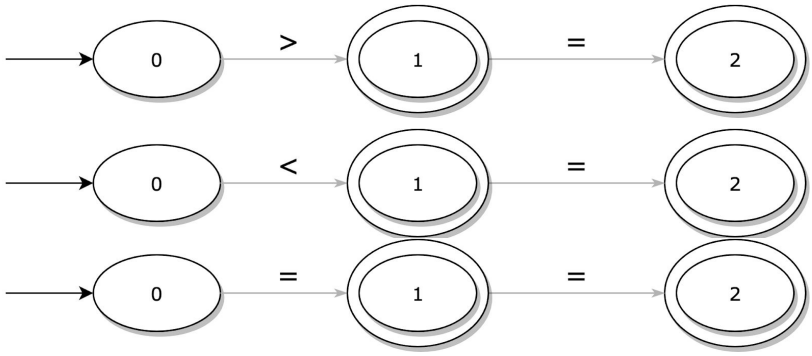
2) Diagramas Transição (Autômatos) tokens

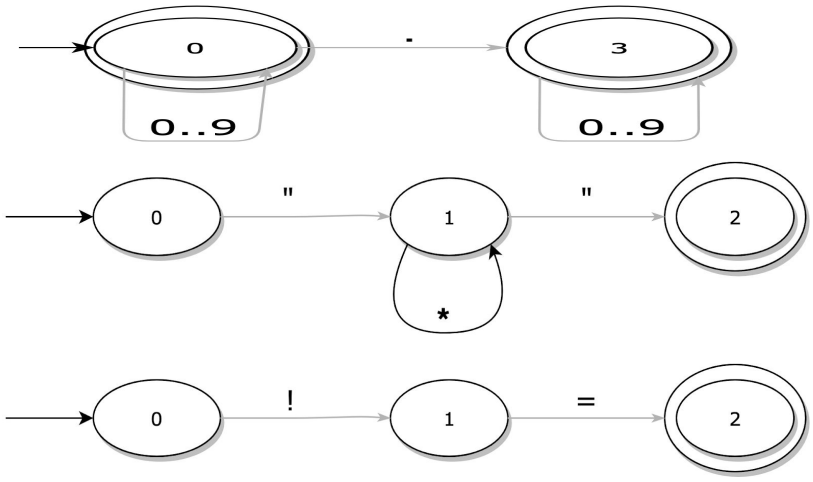



0 f 1 l 2 o 3 a 4 t 5
0 b 1 r 2 e 3 a 4 k 5
0 s 1 t 2 r 3 i 4 n 5 g 6
0 r 1 e 2 t 3 u 4 r 5 n 6
0 ; 2


0
(
6
)
5
}
{
]
[
1
2
3
4


0
-
5
*
+
%
/
4
1
2
3


0 > 1 = 2
0 < 1 = 2
0 = 1 = 2



3) Descrição da implementação analisador léxico

Primeiramente abrimos o arquivo que contém a linguagem, a partir dele substituímos onde tem strings delimitadas por “” por uma constante string. Após isso adicionamos os espaços em branco entre todos os símbolos da entrada, seguido de uma remoção de espaços em branco duplos, com isso unimos símbolos como “= =”.

Temos também um dicionário que contém os símbolos e palavras reservadas da gramática. Para realizar o teste se uma palavra está na dita gramática usamos separados por espaço em branco e conferimos se as substrings estão neste dicionário.

A saída do analisador léxico e uma lista com todos os tokens em ordem.

Respostas Analisador Sintático

Questao 1, CCC-2019-2;

```
PROGRAM → STATEMENT | epson
STATEMENT → VARDECL; | ATRIBSTAT; | PRINTSTAT; | READSTAT; | RETURNSTAT;
| IFSTAT; | FORSTAT; | {STATELIST} | break ; | ;
VARDECL     → int ident VAR2 |
              float ident VAR2 |
              string ident VAR2
VAR2        → [int_constant] VAR2 |
              epson
ATRIBSTAT → LVALUE = EXPRESSION |
            LVALUE = ALLOCEXPRESSION
PRINTSTAT → print EXPRESSION
READSTAT → read LVALUE
RETURNSTAT → return
```

```
IFSTAT      → if( EXPRESSION ) STATEMENT |
              if( EXPRESSION ) STATEMENT else STATEMENT
FORSTAT → for( ATRIBSTAT; NUMEXPRESSION; ATRIBSTAT ) STATEMENT
STATELIST → STATEMENT |
            STATEMENT STATELIST
ALLOCEXPRESSION → new int [ EXPRESSION ] ALLOCEXP |
                  new float [ EXPRESSION ] ALLOCEXP |
                  new string [ EXPRESSION ] ALLOCEXP
ALLOCEXP    → [ EXPRESSION ] ALLOCEXP |
              epson
EXPRESSION  → NUMEXPRESSION |
              NUMEXPRESSION < NUMEXPRESSION |
              NUMEXPRESSION > NUMEXPRESSION |
              NUMEXPRESSION <= NUMEXPRESSION |
              NUMEXPRESSION >= NUMEXPRESSION |
              NUMEXPRESSION == NUMEXPRESSION |
              NUMEXPRESSION != NUMEXPRESSION
NUMEXPRESSION     → TERM |
                    TERM + TERM NUM2|
                    TERM - TERM NUM2
NUM2        → + TERM NUM2|
              - TERM NUM2|
              epson
TERM        → UNARYEXPR |
              UNARYEXPR * UNARYEXPR TERM2 |
              UNARYEXPR \ UNARYEXPR TERM2 |
              UNARYEXPR % UNARYEXPR TERM2
TERM2       → * UNARYEXPR TERM2|
              \ UNARYEXPR TERM2|
              % UNARYEXPR TERM2|
              epson
UNARYEXPR   → + FACTOR | - FACTOR| FACTOR
FACTOR      → int_constant | float_constant | string_constant | null |
              LVALUE |
              ( EXPRESSION )
LVALUE      → ident |
              ident [EXPRESSION] LVALUEEXP
LVALUEEXP   → [ EXPRESSION ] LVALUEEXP | epson
```

Questao 2,

Não há recursão à esquerda na linguagem CCC-2019-2, pois nenhum não-terminal deriva direta ou indiretamente o próprio não-terminal.

Questão 3,

Foi necessário realizar a fatoração à esquerda na gramática em vários casos em que produções à direita começavam com o(s) mesmo(s) símbolo(s) de entrada.

Exemplo:

De:

```
IFSTAT      → if( EXPRESSION ) STATEMENT |
              if( EXPRESSION ) STATEMENT else STATEMENT
```

Para:

```
IFSTAT      → if( EXPRESSION ) STATEMENT IF2
IF2         → else STATEMENT | e
```

No caso abaixo o NUMEXPRESSION se repete no início das produções e deve ser fatorada:

```
De: EXPRESSION    → NUMEXPRESSION |
                    NUMEXPRESSION < NUMEXPRESSION |
                    NUMEXPRESSION > NUMEXPRESSION |
                    NUMEXPRESSION <= NUMEXPRESSION |
                    NUMEXPRESSION >= NUMEXPRESSION |
                    NUMEXPRESSION == NUMEXPRESSION |
                    NUMEXPRESSION != NUMEXPRESSION
Para:
      EXPRESSION → NUMEXPRESSION EXP2
      EXP2 → < NUMEXPRESSION |
             > NUMEXPRESSION |
             <= NUMEXPRESSION |
             >= NUMEXPRESSION |
             == NUMEXPRESSION |
             != NUMEXPRESSION |
             epson
```

Gramática fatorada:

```
PROGRAM     → STATEMENT |
              epson
STATEMENT   → VARDECL; |
              ATRIBSTAT; |
              PRINTSTAT; |
              READSTAT; |
              RETURNSTAT; |
              IFSTAT |
              FORSTAT |
              {STATELIST} |
              break ; |   ;
VARDECL     → int ident VAR2 |
```

```
              float ident VAR2 |
              string ident VAR2
VAR2          → [int_constant] VAR2 |
              epson
ATRIBSTAT     → LVALUE = ATREXP
ATREXP        → EXPRESSION |
              ALLOCEXPRESSION
PRINTSTAT     → print EXPRESSION
READSTAT      → read LVALUE
RETURNSTAT    → return
IFSTAT        → if( EXPRESSION ) STATEMENT IF2
IF2           → else STATEMENT |
              epson
FORSTAT       → for( ATRIBSTAT; EXPRESSION; ATRIBSTAT ) STATEMENT
STATELIST     → STATEMENT STATE2
STATE2        → STATELIST |
              epson
ALLOCEXPRESSION → new ALLOC2
ALLOC2        → int[ EXPRESSION ] ALLOC3 |
              float[ EXPRESSION ] ALLOC3 |
              string[ EXPRESSION ] ALLOC3
ALLOC3        → [ EXPRESSION ] ALLOC3 |
              epson
EXPRESSION    → NUMEXPRESSION EXP2
EXP2          → < NUMEXPRESSION |
              > NUMEXPRESSION |
              <= NUMEXPRESSION |
              >= NUMEXPRESSION |
              == NUMEXPRESSION |
              != NUMEXPRESSION |
              epson
NUMEXPRESSION → TERM NUM2
NUM2          → + TERM NUM2|
              - TERM NUM2|
              epson
TERM          → UNARYEXPR TERM2
TERM2         → * UNARYEXPR TERM2 |
              \ UNARYEXPR TERM2 |
              % UNARYEXPR TERM2 |
              epson
UNARYEXPR     → + FACTOR | - FACTOR| FACTOR
FACTOR        → int_constant |
              float_constant |
              string_constant |
              null |
              LVALUE |
              ( EXPRESSION )
LVALUE        → ident ALLOC3
```

Questão 4, Transforme em LL(1)

Abaixo segue a tabela de referência no formato NãoTerminal, Terminal -> Produção :

PROGRAM,OPENBRACE->STATEMENT

PROGRAM,BREAK->STATEMENT

PROGRAM,INTEGER_T->STATEMENT

PROGRAM,FLOAT_T->STATEMENT

PROGRAM,STRING_T->STATEMENT

PROGRAM,IDENT->STATEMENT

PROGRAM,PRINT->STATEMENT

PROGRAM,READ->STATEMENT

PROGRAM,RETURN->STATEMENT

PROGRAM,IF->STATEMENT

PROGRAM,FOR->STATEMENT

PROGRAM,SEPARATOR->STATEMENT

PROGRAM,DOLLAR->ε

STATEMENT,OPENBRACE->OPENBRACE STATELIST CLOSEBRACE

STATEMENT,BREAK->BREAK SEPARATOR

STATEMENT,INTEGER_T->VARDECL SEPARATOR

STATEMENT,FLOAT_T->VARDECL SEPARATOR

STATEMENT,STRING_T->VARDECL SEPARATOR

STATEMENT,IDENT->ATRIBSTAT SEPARATOR

STATEMENT,PRINT->PRINTSTAT SEPARATOR

STATEMENT,READ->READSTAT SEPARATOR

STATEMENT,RETURN->RETURNSTAT SEPARATOR

STATEMENT,IF->IFSTAT

STATEMENT,FOR->FORSTAT

STATEMENT,SEPARATOR->SEPARATOR

VARDECL,INTEGER_T->INTEGER_T IDENT VAR2

VARDECL,FLOAT_T->FLOAT_T IDENT VAR2

VARDECL,STRING_T->STRING_T IDENT VAR2

VAR2,OPENBRKT->OPENBRKT INT CLOSEBRKT VAR2

VAR2,SEPARATOR->ε

ATRIBSTAT,IDENT->LVALUE ASSERT ATREXP

ATREXP,ADD->EXPRESSION

ATREXP,MINUS->EXPRESSION

ATREXP,INT->EXPRESSION

ATREXP,FLT->EXPRESSION

ATREXP,STR->EXPRESSION

ATREXP,NULL->EXPRESSION

ATREXP,IDENT->EXPRESSION

ATREXP,OPENPARENT->EXPRESSION

ATREXP,NEW->ALLOCEXPRESSION

PRINTSTAT,PRINT->PRINT EXPRESSION

READSTAT,READ->READ LVALUE

RETURNSTAT,RETURN->RETURN

IFSTAT,IF->IF OPENPARENT EXPRESSION CLOSEPARENT STATEMENT IF2

IF2,ELSE->ELSE STATEMENT

IF2,DOLLAR->ε

IF2,OPENBRACE->ε

IF2,BREAK->ε

IF2,INTEGER_T->ε

IF2,FLOAT_T->ε

IF2,STRING_T->ε

IF2,IDENT->ε

IF2,PRINT->ε

IF2,READ->ε

IF2,RETURN->ε

IF2,IF->ε

IF2,FOR->ε

IF2,SEPARATOR->ε

IF2,CLOSEBRACE->ε

FORSTAT,FOR->FOR OPENPARENT ATRIBSTAT SEPARATOR EXPRESSION SEPARATOR ATRIBSTAT CLOSEPARENT STATEMENT

STATELIST,OPENBRACE->STATEMENT STATE2

STATELIST,BREAK->STATEMENT STATE2

STATELIST,INTEGER_T->STATEMENT STATE2

STATELIST,FLOAT_T->STATEMENT STATE2

STATELIST,STRING_T->STATEMENT STATE2

STATELIST,IDENT->STATEMENT STATE2

STATELIST,PRINT->STATEMENT STATE2

STATELIST,READ->STATEMENT STATE2

STATELIST,RETURN->STATEMENT STATE2

STATELIST,IF->STATEMENT STATE2

STATELIST,FOR->STATEMENT STATE2

STATELIST,SEPARATOR->STATEMENT STATE2

STATE2,OPENBRACE->STATELIST

STATE2,BREAK->STATELIST

STATE2,INTEGER_T->STATELIST

STATE2,FLOAT_T->STATELIST

STATE2,STRING_T->STATELIST

STATE2,IDENT->STATELIST

STATE2,PRINT->STATELIST

STATE2,READ->STATELIST

STATE2,RETURN->STATELIST

STATE2,IF->STATELIST

STATE2,FOR->STATELIST

STATE2,SEPARATOR->STATELIST

STATE2,CLOSEBRACE->ε

ALLOCEXPRESSION,NEW->NEW ALLOC2

ALLOC2,INTEGER_T->INTEGER_T OPENBRKT EXPRESSION CLOSEBRKT ALLOC3

ALLOC2,FLOAT_T->FLOAT_T OPENBRKT EXPRESSION CLOSEBRKT ALLOC3

ALLOC2,STRING_T->STRING_T OPENBRKT EXPRESSION CLOSEBRKT ALLOC3

ALLOC3,OPENBRKT->OPENBRKT EXPRESSION CLOSEBRKT ALLOC3

ALLOC3,SEPARATOR->ε

ALLOC3,ASSERT->ε

ALLOC3,MULTIPLY->ε

ALLOC3,DIVIDE->ε

ALLOC3,MODULUS->ε

ALLOC3,ADD->ε

ALLOC3,MINUS->ε

ALLOC3,LT->ε

ALLOC3,GT->ε

ALLOC3,LE->ε

ALLOC3,GE->ε

ALLOC3,EQ->ε

ALLOC3,NE->ε

ALLOC3,CLOSEPARENT->ε

ALLOC3,CLOSEBRKT->ε

EXPRESSION,ADD->NUMEXPRESSION EXP2

EXPRESSION,MINUS->NUMEXPRESSION EXP2

EXPRESSION,INT->NUMEXPRESSION EXP2

EXPRESSION,FLT->NUMEXPRESSION EXP2

EXPRESSION,STR->NUMEXPRESSION EXP2

EXPRESSION,NULL->NUMEXPRESSION EXP2

EXPRESSION,IDENT->NUMEXPRESSION EXP2

EXPRESSION,OPENPARENT->NUMEXPRESSION EXP2

EXP2,LT->LT NUMEXPRESSION

EXP2,GT->GT NUMEXPRESSION

EXP2,LE->LE NUMEXPRESSION

EXP2,GE->GE NUMEXPRESSION

EXP2,EQ->EQ NUMEXPRESSION

EXP2,NE->NE NUMEXPRESSION

EXP2,SEPARATOR->ε

EXP2,CLOSEPARENT->ε

EXP2,CLOSEBRKT->ε

NUMEXPRESSION,ADD->TERM NUM2

NUMEXPRESSION,MINUS->TERM NUM2

NUMEXPRESSION,INT->TERM NUM2

NUMEXPRESSION,FLT->TERM NUM2

NUMEXPRESSION,STR->TERM NUM2

NUMEXPRESSION,NULL->TERM NUM2

NUMEXPRESSION,IDENT->TERM NUM2

NUMEXPRESSION,OPENPARENT->TERM NUM2

NUM2,ADD->ADD TERM NUM2

NUM2,MINUS->MINUS TERM NUM2

NUM2,LT->ε

NUM2,GT->ε

NUM2,LE->ε

NUM2,GE->ε

NUM2,EQ->ε

NUM2,NE->ε

NUM2,SEPARATOR->ε

NUM2,CLOSEPARENT->ε

NUM2,CLOSEBRKT->ε

TERM,ADD->UNARYEXPR TERM2

TERM,MINUS->UNARYEXPR TERM2

TERM,INT->UNARYEXPR TERM2

TERM,FLT->UNARYEXPR TERM2

TERM,STR->UNARYEXPR TERM2

TERM,NULL->UNARYEXPR TERM2

TERM,IDENT->UNARYEXPR TERM2

TERM,OPENPARENT->UNARYEXPR TERM2

TERM2,MULTIPLY->MULTIPLY UNARYEXPR TERM2

TERM2,DIVIDE->DIVIDE UNARYEXPR TERM2

TERM2,MODULUS->MODULUS UNARYEXPR TERM2

TERM2,ADD->ε

TERM2,MINUS->ε

TERM2,LT->ε

TERM2,GT->ε

TERM2,LE->ε

TERM2,GE->ε

TERM2,EQ->ε

TERM2,NE->ε

TERM2,SEPARATOR->ε

TERM2,CLOSEPARENT->ε

TERM2,CLOSEBRKT->ε

UNARYEXPR,ADD->ADD FACTOR

UNARYEXPR,MINUS->MINUS FACTOR

UNARYEXPR,INT->FACTOR

UNARYEXPR,FLT->FACTOR

UNARYEXPR,STR->FACTOR

UNARYEXPR,NULL->FACTOR

UNARYEXPR,IDENT->FACTOR

UNARYEXPR,OPENPARENT->FACTOR

FACTOR,INT->INT

FACTOR,FLT->FLT

FACTOR,STR->STR

FACTOR,NULL->NULL

FACTOR,IDENT->LVALUE

FACTOR,OPENPARENT->OPENPARENT EXPRESSION CLOSEPARENT

LVALUE,IDENT->IDENT ALLOC3

Questão 5, Descrição da implementação do analisador sintático.

Primeiramente inicializamos o dicionário de produções com as produções da gramática proposta. Em seguida calculamos os Follows e depois a tabela de referência que em seu método inicializa os Firsts. Depois usamos o método PreditiveParser para montar a pilha da tabela de predição com os tokens de entrada.

REFERÊNCIAS

1. AHO, A.V.; LAM, M. S.; SETHI, R. ULLMAN, J.D. Compiladores – Princípios, Técnicas e Ferramentas, Pearson, 2008
2. DELAMARO, Márcio Eduardo. Como Construir um compilador. São Paulo, Novatec, 2004.