Para a implementação do analisador foi necessário remover a recursão à esquerda presente nessas produções:

if-stmt ::= if condition then stmt-list end |

if condition then stmt-list else stmt-list end

expression ::= simple-expr |

simple-expr relop simple-expr

Dessa maneira, obteve-se:

if-stmt ::= if condition then stmt-list if-stmt’

if-stmt’ ::= end | else stmt-list end

expression ::= simple-expr expression’

expression’ ::= λ | relop simple-expr

No qual se obtém a seguinte gramática

1. program ::= [ **var** decl-list] **begin** stmt-list **end**
2. decl-list ::= decl ";" { decl ";"}
3. decl ::= ident-list **is** type
4. ident-list ::= identifier {"," identifier}
5. type ::= **int** | **string**
6. stmt-list ::= stmt ";" { stmt ";"}
7. stmt ::= assign-stmt | if-stmt | do-stmt | read-stmt | write-stmt
8. assign-stmt ::= identifier ":=" simple\_expr
9. if-stmt ::= **if** condition **then** stmt-list if-stmt’
10. if-stmt’ ::= **end** | **else** stmt-list **end**
11. condition ::= expression
12. do-stmt ::= **do** stmt-list stmt-suffix
13. stmt-suffix ::= **while** condition
14. read-stmt ::= **in** "(" identifier ")"
15. write-stmt ::= **out** "(" writable ")"
16. writable ::= simple-expr
17. expression ::= simple-expr expression’
18. expression’ ::= λ | relop simple-expr
19. simple-expr ::= term | simple-expr addop term
20. term ::= factor-a | term mulop factor-a
21. fator-a ::= factor | **not** factor | "-" factor
22. factor ::= identifier | constant | "(" expression ")"
23. relop ::= "=" | ">" | ">=" | "<" | "<=" | "<>"
24. addop ::= "+" | "-" | **or**
25. mulop ::= "\*" | "/" | **and**
26. constant ::= integer\_const | literal
27. integer\_const ::= nozero {digit} | “0”
28. literal ::= " {" {caractere} "}"
29. identifier ::= (letter) {letter | digit } |
30. “\_” (letter | digit ) {letter | digit }
31. letter ::= [A-Za-z]
32. digit ::= [0-9]
33. nozero ::= [1-9]
34. caractere ::= um dos 256 caracteres do conjunto ASCII, exceto “{”, “}” e quebra de linha

|  |  |  |
| --- | --- | --- |
|  | FIRST | FOLLOW |
| program | var, begin | $ |
| decl-list | identifier | begin |
| decl | identifier | “;” |
| ident-list | identifier | is |
| type | int, string | “;” |
| stmt-list | identifier, if, do, in, out | end, else, while |
| stmt | identifier, if, do, in, out | “;” |
| assign-stmt | identifier | “;” |
| if-stmt | if | “;” |
| if-stmt’ | end, else | “;” |
| condition | identifier, literal, constant, “(”, not, “-” | then, “;” |
| do-stmt | do | “;” |
| stmt-suffix | while | “;” |
| read-stmt | in | “;” |
| write-stmt | out | “;” |
| writable | identifier, literal, constant, “(”, not, “-” | “)” |
| expression | identifier, literal, constant, “(”, not, “-” | “)”, then, “;” |
| expression’ | , relop | “)”, then, “;” |
| simple-expr | identifier, literal, constant, “(”, not, “-” | "=", ">", ">=", "<", "<=", "<>", “+”, “-”, or, “)”, then, “;” |
| term | identifier, literal, constant, “(”, not, “-” | “\*”, “/”, and, "=", ">", ">=", "<", "<=", "<>", “+”, “-”, or, “)”, then, “;” |
| factor-a | identifier, literal, constant, “(”, not, “-” | “\*”, “/”, and, "=", ">", ">=", "<", "<=", "<>", “+”, “-”, or, “)”, then, “;” |
| factor | identifier, constant, “(” | “\*”, “/”, and, "=", ">", ">=", "<", "<=", "<>", “+”, “-”, or, “)”, then, “;” |
| relop | "=" , ">" , ">=" , "<" , "<=", "<>" | identifier, constant, “(”, not, “-” |
| addop | “+”, “-”, or | identifier, constant, “(”, not, “-” |
| mulop | “\*”, “/”, and | identifier, constant, “(”, not, “-” |
| constant | integer\_const,literal | “\*”, “/”, and, "=", ">", ">=", "<", "<=", "<>", “+”, “-”, or, “)”, then, “;” |

A tabela do parser se encontra no arquivo excel separado, com o nome de TableParser.xlsx