Para a implementação do analisador foi necessário remover a repetição 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

decl-list ::= decl ";" { decl ";"}

ident-list ::= identifier {"," identifier}

stmt-list ::= stmt ";" { stmt ";"}

simple-expr ::= term | simple-expr addop term

term ::= factor-a | term mulop factor-a

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

decl-list ::= decl ";" decl-list'

decl-list' ::= λ | decl-list

ident-list ::= identifier ident-list'

ident-list' ::= λ | "," ident-list

stmt-list ::= stmt ";" stmt-list’

stmt-list’ ::= λ | stmt-list

simple-expr ::= term simple-expr’

simple-expr’ ::= λ | addop simple-expr

term ::= factor-a term’

term’ ::= λ | mulop term

No qual se obtém a seguinte gramática

1. program ::= **var** decl-list **begin** stmt-list **end**
2. program ::= **begin** stmt-list **end**
3. decl-list ::= decl **";"** decl-list’
4. decl-list’ ::= λ
5. decl-list’ ::= decl-list
6. decl ::= ident-list **is** type
7. ident-list ::= **identifier** ident-list’
8. ident-list’ ::= λ
9. ident-list’ ::= **","** ident-list
10. type ::= **int**
11. type ::= **string**
12. stmt-list ::= stmt **";"** stmt-list’
13. stmt-list’ ::= λ
14. stmt-list’ ::= stmt-list
15. stmt ::= assign-stmt
16. stmt ::= if-stmt
17. stmt ::= do-stmt
18. stmt ::= read-stmt

19. stmt ::= write-stmt
20. assign-stmt ::= **identifier** **":="** simple_expr
21. if-stmt ::= **if** condition **then** stmt-list if-stmt’
22. if-stmt’ ::= **end**
23. if-stmt’ ::= **else** stmt-list **end**
24. condition ::= expression
25. do-stmt ::= **do** stmt-list stmt-suffix
26. stmt-suffix ::= **while** condition
27. read-stmt ::= **in** **"("** identifier **")"**
28. write-stmt ::= **out** **"("** writable **")"**
29. writable ::= simple-expr
30. expression ::= simple-expr expression’
31. expression’ ::= λ
32. expression’ ::= relop simple-expr
33. simple-expr ::= term simple-expr’
34. simple-expr’ ::= λ
35. simple-expr’ ::= addop simple-expr
36. term ::= factor-a term’
37. term’ ::= λ
38. term’ ::= mulop term
39. factor-a ::= factor
40. factor-a ::= **not** factor
41. factor-a ::= **"-"** factor
42. factor ::= **identifier**
43. factor ::= constant
44. factor ::= **"("** expression **")"**
45. relop ::= **"="**
46. relop ::= **">"**
47. relop ::= **">="**
48. relop ::= **"<"**
49. relop ::= **"<="**
50. relop ::= **"<>"**

51. addop ::= **"+"**
52. addop ::= **"-"**
53. addop ::= **or**
54. mulop ::= **"*"**
55. mulop ::= **"/"**
56. mulop ::= **and**
57. constant ::= **integer_const**
58. constant ::= **literal**

Tabela First - Follow

	FIRST	FOLLOW
program	var, begin	$
decl-list	identifier	begin
decl-list’	λ, identifier	begin
decl	identifier	“;”
ident-list	identifier	is
ident-list’	λ, “,”	is
type	int, string	“;”
stmt-list	identifier, if, do, in, out	end, else, while
stmt-list’	λ, 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, integer_const, “(”, not, “-”	then, “;”
do-stmt	do	“;”
stmt-suffix	while	“;”
read-stmt	in	“;”
write-stmt	out	“;”

writable	identifier, literal, integer_const, “(”, not, “-”	“)”
expression	identifier, literal, integer_const, “(”, not, “-”	“)”, then, “;”
expression’	λ, "=" , ">" , ">=" , "<" , "<=", "<>"	“)”, then, “;”
simple-expr	identifier, literal, integer_const, “(”, not, “-”	"=", ">", ">=", "<", "<=", "<>", “)”, then, “;”
simple-expr’	λ, “+”, “-”, or	"=", ">", ">=", "<", "<=", "<>", “)”, then, “;”
term	identifier, literal, integer_const, “(”, not, “-”	"=", ">", ">=", "<", "<=", "<>", “+”, “-”, or, “)”, then, “;”
term’	“*”, “/”, and	"=", ">", ">=", "<", "<=", "<>", “+”, “-”, or, “)”, then, “;”
factor-a	identifier, literal, integer_const, “(”, not, “-”	“*”, “/”, and, "=", ">", ">=", "<", "<=", "<>", “+”, “-”, or, “)”, then, “;”
factor	identifier, integer_const, literal, “(”	“*”, “/”, and, "=", ">", ">=", "<", "<=", "<>", “+”, “-”, or, “)”, then, “;”
relop	"=" , ">" , ">=" , "<" , "<=", "<>"	identifier, integer_const, “(”, not, “-”
addop	“+”, “-”, or	identifier, integer_const, “(”, not, “-”
mulop	“*”, “/”, and	identifier, integer_const, “(”, not, “-”
constant	integer_const, literal	“*”, “/”, and, "=", ">", ">=", "<", "<=", "<>", “+”, “-”, or, “)”, then, “;”

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