Lexic.txt

Alphabet:

a. Upper (A-Z) and lower case letters (a-z)
b. Underline character '_'
c. Decimal digits (0-9)

Lexic:

a. Special symbols:
   - operators: + - * / \ < > = <= >= == != % !
   - separators: ( ) [ ] { } : ; , ' " space newline
   - reserved words: be number integer string char bool const check else readFromConsole showInConsole stopWhen for function
b. Identifiers:
   - it is a sequence and chars and digits, the first letter being a letter:
     - letter = "a" | "b" | ... | "z" | "A" | "B" | ... | "Z"
     - digit = "0" | "1" | "2" | ... | "9"
     - identifier = letter | letter {letter} {digit}
c. Constants:
   - number:

     number = ["-"]digit {digit}
   - char:

     char = 'letter' | 'digit'
   - string:

     char = letter | digit

     string = char {string}

token.in

(

)

*

-

+

/

\

:

;

[

]

{

}

%

=

<

>

==

<=

>=

!=

&

|

‘

“

!

’

be

number

integer

bool

string

char

const

check

else

readFromConsole

showInConsole

stopWhen

function

for

space

newline

Syntax.in

program ::=

decllist ::= declaration | declaration decllist

declaration ::= "be" IDENTIFIER type

type1 ::= "bool" | "char" | "integer" | "number" | "string"

arraydecl ::= type1 "[" nr "]"

type ::= type1 | arraydecl

cmpdstmt ::= stmtlist

stmtlist ::= stmt | stmt stmtlist

stmt ::= simplstmt | structstmt

simplstmt ::= assignstmt | iostmt

assignstmt ::= IDENTIFIER "=" expression

expression ::= expression ("+" | "-") term | term

term ::= term ("*" | "/" | "%") factor | factor

factor ::= "(" expression ")" | CONST | IDENTIFIER

iostmt ::= "readFromConsole" "(" IDENTIFIER ")" | "showInConsole" "(" IDENTIFIER ")"

structstmt ::= cmpdstmt | ifstmt | whilestmt

ifstmt ::= "check" "(" condition ")" "{" cmpdstmt "}" "else" "{" cmpdstmt "}"

whilestmt ::= "stopwhen" "(" condition ")" "{" cmpdstmt "}"

condition ::= expression relation expression ("| |" | "&&" condition)

relation::= "<" | "<=" | "==" | "!=" | ">=" | ">"