Mini Language Specification

Token.in

declare
identifier
constant
int
string
boolean
char
{
}
[
]
(
)
,
;
>
<=
>=
<
>
=
==
.
+
-
*
/
%
div
mod
print
read
if
elif
else
True
False
for
by
increase
while
break
“”
‘’
start
end

Lexic.txt

a.Special symbols, representing:

- operators: + - * / % = < > <= >=
- separators [ ] { } : ; “” ‘’ ( ) ,
- reserved words: declare int string boolean char div mod print read if elif else True False for while increase by start end break

b.Identifiers

-a sequence of letters and digits, such that the first character is a letter; the rule is:

identifier = letter | letter{letter}{digit}
letter = "A" | "B" | . ..| "Z"
digit = "0" | "1" |...| "9"

c.Constants

1. int : no = “0” | sign non_zero digit
   non_zero = “1” | “2”| “3”| …| “9”
   sign = [‘+’| ‘-‘]
2. char
   char = “ ‘ ” [letter|digit] “ ‘ ”
3. string
   string= “ “ ” { char } “ “ ”
4. boolean
   bool = “True”|”False”

Syntax.in

Syntactic rules:

program = "START" “:” LISTdecl “END” "."
LISTdecl = stmt | stmt ";" LISTdecl
type1 = "BOOLEAN" | "CHAR" | "INT" | “STRING”
ARRAYdecl = identifier "[" integer "]" “;”
type = type1 ARRAYdecl | type1 identifier “;”
LISTstmt = stmt | stmt ";" LISTstmt
stmt = declaration | SIMPLEstmt | STRUCTstmt | BREAKstmt
declaration = “DECLARE” type
SIMPLEstmt = ASSIGNstmt | IOstmt
STRUCTstmt = IFstmt | WHILEstmt | FORstmt
BREAKstmt = “break” “;”
ASSIGNstmt = identifier "=" expression “;”
IOstmt = ["READ" | "PRINT"] "(" identifier ")" “;” | “PRINT” “(” constant ”)” “;”
IFstmt = "IF” “(" condition ")” “{" stmt “}” “;” ["ELSE{" stmt “}”| “ELIF” “(" condition ")” ”{" stmt “}” “;”]
WHILEstmt = "WHILE” “(" condition ")” “{" stmt “}” “;”
FORstmt = “FOR(” “increase by” integer “;” identifier “=” integer “,” identifier “){” stmt “}”
expression = term | expression operation expression
term = identifier | constant
condition = expression RELATION expression
RELATION = "<" | "<=" | "=" | "<>" | ">=" | ">"
OPERATION = “+” | “-” | “*” | “/” | “%” | “mod” | “div”