“Lexic.txt”

Alphabet:
a. Upper (A-Z) and lower case letters (a-z) of the English Alphabet
b. Decimal digits (0-9)
c. Underscore (_)

Lexic:
operators: + - * / % < > <= = == >= && ||
separators: space < > { } ( ) " " ,
reserved words:
int string char bool array if while for READ PRINT BEGIN END STOP true false

identifiers (< 256 chrs):
- sequence of maximum 256 characters, letters and digits or underscore (_), such that the first character is a letter or underscore(_)
identifier := {"_"}letter{letter | digit}
letter := "a" | "b" | ... | "z" | "A" | "B" | ... | "Z"
digit := "0" | "1" | ... | "9"

constants:
integer:
intconst := [+ | -]no
no := digit{no}

string:
strconst := """string"""
string := char{string}
chr := letter|digit

character:
charconst := "' chr "'

boolean:
boolconst := "true" | "false"

“Syntax.in”

OPERATOR := "+" | "-" | "*" | "/" | "%" | "&&" | "||"
IDENTIFIER := {"_"}letter{letter | digit}
letter := "a" | "b" | ... | "z" | "A" | "B" | ... | "Z"
digit := "0" | "1" | ... | "9"

expression := expression OPERATOR term | term
term := "(" expression ")" | IDENTIFIER

RELATION := "<" | "<=" | "=" | "!=" | ">=" | ">"

condition := expression RELATION expression

ioStmt := ("READ" | "PRINT") expression
assignStmt := IDENTIFIER "=" expression
simplStmt := ioStmt | assignStmt

ifStmt := "if" "(" condition ")" stmt ["else" stmt]
whileStmt := "while" "(" condition ")" stmt

structStmt := ifStmt | whileStmt | cmpdStmt
stmt := structStmt | simplStmt | declaration
stmtlist := stmt {"endline" stmt}
cmpdStmt := "BEGIN" stmtlist "END"

primType := "int" | "string" | "bool" | "char"
arrType := "array" "(" primType ")" "[" no "]"
type := primType | arrType

declaration := type "{" IDENTIFIER {","IDENTIFIER} "}"

program := stmtlist

“tokens.in”

+
-
*
/
%
<
>
<=
=
==
>=
&&
||
<
>
{
}
(
)
"
"
,
int
string

char
bool
array
if
while
for
READ
PRINT
BEGIN
END
STOP
true
false