Alphabet:

a. Upper (A-Z) and lower case letters (a-z) of the English alphabet
b. Decimal digits (0-9);

Special symbols :
- operators + - * / < <= = >= !=
- separators [ ] { } : ; space ( )

Reserved Words:

array, int, char, if, else, read, write, for, while

Identifiers:

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

identifier = letter{(letter|digit)}

letter = "A" | "B" | ... | "Z" | "a" | "b" | ... | "z" |

digit = "0" | "1" |...| "9" |

Constants:

integer:
noconst =+non_zero_number|-non_zero_number|non_zero_number|"0"
non_zero_digit="1"|"2"|....|"9"
non_zero_number=non_zero_digit{digit}

bool:
boolean = "false" | "true"

character:
character='letter'|'digit'

string:
constchar="string"
string=character{string}
character=letter|digit

Language Specification:

Syntax:
The words - predefined tokens are specified between " and ":

Rules:

program = {(decllist | cmpdstmt)}

decllist = declaration | declaration ";" decllist

declaration = IDENTIFIER ":" type

IDENTIFIER = id | id "," LIDENTIFIER;

type1 = "boolean" | "char" | "int"

arraydecl = type1 "ARRAY" "[" 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 ")" | IDENTIFIER

iostmt = "read" "(" IDENTIFIER ")" | "write" "(" IDENTIFIER ")"

structstmt = cmpdstmt | ifstmt | whilestmt

ifstmt = "if" condition stmt |"if" condition stmt "else" stmt

whilestmt = "while" condition stmt

condition = expression RELATION expression

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

Tip atom
identificator
constanta
program
array
of
var
int
real
boolean
read
write
for
if
then
else
and
or
not
!
:
;
,
.
+
*
(
)

```
------------------
[
------------------
]
------------------
-
------------------
<
------------------
>
------------------
=
------------------
```