lexic.txt

Alphabet:

a. Upper (A-Z) and lower case letters (a-z) of the English alphabet

b. Underline character '_';

c. Decimal digits (0-9);

a.Special symbols, representing:

- operators + - * / % := < <= == >= !=

- separators [ ] { } : ; space

- reserved words:

array, char, write_to_console, else, if, int, read, while, do, read, const

b.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” | …s | “z”.

digit = "0" | "1" | ... | “9”.

c.constants

1. integer - rule:

noconst = [“+” | “-“] no

no = digit{no}

2. character

character = ‘letter’ | ‘digit’

3. string

constchar = "string"

string = char{string}

char = ‘letter’ | ‘digit’

syntax.in

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

program = (decllist stmtlist) “;”.

decllist = declaration [decllist].

declaration = simpledecl | arraydecl | structdecl.

simpledecl = type identifier”;”.

type = “char” | “int”.

arraydecl = type identifier "[" number "]" “;”.

structdecl = "struct" identifier “[“ decllist “]” “;”.

stmtlist = stmt [stmtlist].

stmt = simplstmt | structstmt.

simplstmt = assignstmt | iostmt.

assignstmt = identifier “:=“ expression “;”.

var = identifier | noconst | char | constchar

expression = var | expression operator expression | “(“ expression operator expression “)”

operator = "+" | "-" | "/" | "*" | "%".

iostmt = “read “ identifier “;” | “write_to_console(“ (identifier | noconst | char | constchar) “);”.

structstmt = ifstmt | whilestmt | forstmt.

ifstmt = "if(" condition ") [“ stmtlist “]“ [“else [“ stmtlist “]”] “;”

whilestmt = "while(" condition ") do [“ stmtlist “];”.

forstmt = "for(" assignstmt “,” condition “,” assignstmt ") [“ stmtlist “]”.

condition = expression relation expression.

relation = "<" | "<=" | "=" | "!=" | ">=" | ">".

token.in

```
+
-
*
/
%
<
<=
>
>=
!=
==
:=
;
[
]
(
)
\n
space
int
char
while
do
read
write_to_console
for
if
```