Lexic.txt:  
Alphabet:

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

b. Underline character '\_';

c. Decimal digits (0-9);

d. Special characters !@$%&

Lexic:

a.Special symbols, representing:

- operators

+ - \* / = != < <== ==> > ===

separators: () [] {} ; , space

-reserved words

int alpha arrr fibre make if fi of prgr read show define

now persistent for while and or not starts from transforms stops at

stdin stdout

b.identifiers

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

<identifier> ::= <letter>|\_|<letter><digit><identifier>|\_<digit><identifier>|

<letter><identifier>|\_<identifier>|<identifier><digit>

<digit> ::= 0|1|...|9

<letter> ::= A|B|...|Z|a|b|...|z

c.constants

1.integer - rule:

<nonzerodigit> ::= 1|2|...|9

<naturalnumber> := <nonzerodigit>|<nonzerodigit><digit\_sequence>

<digit\_sequence>::=<digit>|<digit><digit\_sequence>

<integer> ::= <naturalnumber>|+<naturalnumber>|-<naturalnumber>|0

2.character

<character> ::= '<letter>'|'<digit>'|'\_'|' '|'!'|'@'|'$'|'%'|'&'

3.string

<string> ::= "<character\_aux>"|"<character\_aux><character\_seq>"

<character\_seq> ::= <character\_aux>|<character\_aux><character\_seq>

<character\_aux>::=<letter>|<digit>|\_| |!|@|$|%|&

Syntax.in:  
<program> ::= <cmpdstmt>|<program><cmpdstmt>

<constant> ::= <integer>|<character>|<string>

<declaration> ::= "define" <type> <identifier> | <declaration\_and\_assignment>

<declaration\_and\_assignment> ::= "define" <type> <identifier>=<constant>

<simpletype> ::= "alpha"|"fibre"|"int"

<arraydecl> ::= "arrr" "of" <integer\_constant\_or\_identifier> <identifier>

<integer\_constant\_or\_identifier> ::= <integer>|<identifier>

<arrayaccess> ::= <identifier>"["<integer\_constant\_or\_identifier>"]"

<type> ::= <simpletype>|<arraydecl>

<cmpdstmt> ::= {<stmtlist>}

<stmtlist> ::= <stmt> ";" | <stmt> ";" <stmtlist>

<stmt> ::= <simplstmt> | <structstmt> | <declaration>

<simplstmt> ::= <assignstmt> | <iostmt>

<assignstmt> ::= <identifier>=<expression>|<arrayaccess>=<expression>

<expression> ::= <term><operator><expression><term>| <term>

<operator> := +|-|\*|/

<term> ::= <identifier>|<arrayaccess>|<constant>R

<iostmt> ::= <readstmt>|<show>

<readstmt> ::= "read""("<identifier\_or\_arrayaccess>","<channel>")"

<showstmt> ::= "show""("<variable\_or\_constant>","<channel>")"

<identifier\_or\_arrayaccess> = <identifier>|<arrayaccess>

<variable\_or\_constant> = <identifier\_or\_arrayaccess>|<constant>

<channel> ::= "stdin"|"stdout"

<structstmt> ::= <cmpdstmt> | <ifstmt> | <whilestmt> | <forstmt>

<ifstmt> ::= if <condition> {<stmt>} | if <condition> fi {<stmt>} | if <condition> fi <ifstmt>

<whilestmt> ::= <while> <condition> {<stmt>}

<forstmt> ::= for <identifier> starts from <variable\_or\_constant>

transforms into <assignstmt>

stops at <condition>

{<stmt>}

<condition> ::= <expression><relation><expression>

<relation> ::= <|<==|==>|>|===|and|or|not

<increment\_or\_decrement\_variable> ::= <identifier\_or\_arrayaccess><increment\_or\_decrement><integer\_constant\_or\_identifier>

<increment\_or\_decrement> ::= "+="|"-="

<comment> ::= #<string>#

token.in:  
+

-

\*

/

=

<

<==

==>

>

===

()

[]

{}

;

,

+=

-=

int

alpha

arrr

fibre

make

if

fi

of

prgr

read

show

define

now

persistent

for

while

and

or

not

starts

from

transforms

stops

at

stdin

stdout