

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 | underline | (letter | underline){letter}{underline}{digit}

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

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

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

<identifier> ::= <letter> | \_ | <identifier> <alphabet\_item>

c.constants

1.integer - rule:

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

<naturalnumber> := <nonzerodigit> | <number> <digit>

$\langle \text{integer} \rangle ::= \langle \text{naturalnumber} \rangle | +\langle \text{naturalnumber} \rangle | -\langle \text{naturalnumber} \rangle | 0$

2.character

$\langle \text{character} \rangle ::= \langle \text{letter} \rangle | \langle \text{digit} \rangle | \_ | " | ' | ! | @ | \$ | \% | \&$

3.string

$\langle \text{string} \rangle ::= \langle \text{character} \rangle | \langle \text{character} \rangle \langle \text{string} \rangle$

Syntax.in:

$\langle \text{program} \rangle ::= \langle \text{cmpdstmt} \rangle | \langle \text{program} \rangle \langle \text{cmpdstmt} \rangle$

$\langle \text{constant} \rangle ::= \langle \text{integer} \rangle | \langle \text{character} \rangle | \langle \text{string} \rangle$

$\langle \text{declaration} \rangle ::= \text{define } \langle \text{type} \rangle \langle \text{identifier} \rangle [= \langle \text{constant} \rangle]$

$\langle \text{simpletype} \rangle ::= \text{alpha} | \text{fibre} | \text{int}$

$\langle \text{arraydecl} \rangle ::= \text{arr of } (\langle \text{integer} \rangle | \langle \text{identifier} \rangle) \langle \text{identifier} \rangle$

$\langle \text{arrayaccess} \rangle ::= \langle \text{identifier} \rangle ["(\langle \text{identifier} \rangle |)"]$

$\langle \text{type} \rangle ::= \langle \text{simpletype} \rangle | \langle \text{arraydecl} \rangle$

$\langle \text{cmpdstmt} \rangle ::= \{ \langle \text{stmtlist} \rangle \}$

$\langle \text{stmtlist} \rangle ::= \langle \text{stmt} \rangle | \langle \text{stmt} \rangle ; \langle \text{stmtlist} \rangle$

$\langle \text{stmt} \rangle ::= \langle \text{simplstmt} \rangle ; | \langle \text{structstmt} \rangle | \langle \text{declaration} \rangle ;$

$\langle \text{simplstmt} \rangle ::= \langle \text{assignstmt} \rangle | \langle \text{iostmt} \rangle$

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

<expression> ::= ["("] <expression> (+ | - | \* | /) <expression> [")"] | <term>

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

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

<readstmt> ::= read "(" (<identifier> | <arrayaccess>), <channel> ")"

<showstmt> ::= show "(" (<identifier> | <arrayaccess> | <constant>), <channel> ")"

<channel> ::= stdin | stdout

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

<ifstmt> ::= if <condition> {<stmt>} [fi {<stmt>}]

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

<forstmt> ::= for <identifier> starts from (<identifier> | <arrayaccess> | <integer>)

transforms into <assignstmt>

stops at <condition>

{<stmt>}

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

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

<comment> ::= #<string>#

token.in:

int

alpha

arr

fibre

make

if

fi

of

prgr

read

show

define

now

persistent

for

while

and

or

not

starts

from

transforms

stops

at

stdin

stdout