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);

1. Lexic:

a. Special symbols, representing:

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

- separators () <{}> . space ~

- reserved words: MORE char const perform otherwise when then choose nr scan write while var

b. Identifiers

identifier ::= letter | letter{letter}{digit}{symbol}

letter ::= "A" | "B" | . ..| "Z"

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

symbols ::= "@" | "#" | "&" | "~"

c. Constants

1. Integer - rule:

noconst:="+"no|"-"no|no

n:=digit{no}

2. Character

character:= 'letter'|'digit'

3. String

constchar:="string"

string:=char{string}

char:=letter|digit

Syntax.txt

```
program ::= "COD" decllist "~" cmpdstmt "."

decllist ::= declaration | declaration "~" decllist

declaration ::= IDENTIFIER ":" type

type1 ::= "ALBA_NEAGRA" | "VORBE" | "DINAIA" | "DITOATE"

arraydecl ::= "MORE" "[" nr "]" "OF" type1

type  ::= type1|arraydecl

cmpdstmt ::= "ONWARD" stmtlist "AT_EASE"

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 ::= "SCAN" | "WRITE" "(" IDENTIFIER ")"

structstmt ::= cmpdstmt | ifstmt | whilestmt | choosestmt

ifstmt ::= "WHEN" condition "THEN" stmt ["OTHERWISE" stmt]

whilestmt ::= "WHILE" condition "PERFORM" stmt

choosestmt ::= "CHOOSE" "(" expression ")" "{" caselist "}"

caselist ::= case caselist | defaultstmt

case ::= "CASE" constant ":" stmtlist

defaultstmt ::= "DEFAULT" ":" stmtlist

condition ::= expression RELATION expression

RELATION ::= "<" | "<=" | "=" | "<>" | ">=" | ">"
```

Token.txt