Lexic.txt

Alphabet:

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

b. Underline character (_)

c. Decimal digits (0-9)

Special symbols:

- <operators> ::= "+" | "-" | "*" | "/" | "<" | "<=" | "=" | ">="

- <separators> ::= "[" | "]" | "{" | "}" | ":" | ";" | " "

- <reserved_words> ::= "array" | "char" | "const" | "do" | "else" | "check" | "int" | "of" | "program" | "then" | "while" | "for"

Identifiers:

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

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

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

Constants:

integer:

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

<no> ::= <non_zero><digit> | <non_zero>

<non_zero> ::= "1" | "2" | ... | "9"

character:

<character> ::= "'" <letter> "'" | "'" <digit> "'"

string:

<string> ::= """ <charString> """

```
<charString> ::= <char> | <char><charString>
<char> ::= <letter> | <digit>
```

Token.in

```
if
then
for
do
while
.
;
>
<
=
<=
>=
!=
(
)
int
char
string
float
bool
+
-
/
*
```

Syntax.in

```
<program> ::= <decllist> ";" <cmpdstmt>

<decllist> ::= <declaration> | <declaration> ";" <decllist>

<declaration> ::= <type> IDENTIFIER

<type1> ::= "bool" | "char" | "int" | "float" | "string"

<arraydecl> ::= <type1> "[" 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> "then" <stmt> | "if" <condition> "then" <stmt> "else" <stmt>

<whilestmt> ::= "while" <condition> "do" <stmt>

<forstmt> ::= "for" "(" <begin> ";" <end> ";" <step> ")" "do" <stmt>

<begin> ::= <integer>

<end> ::= <integer>

<step> ::= <integer>

<integer> ::= "0" | "1" | "2"| ...

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

<relation> ::= "<" | "<=" | "=" | "!=" | ">=" | ">"
```