Lexic.txt

Lab 1b - Lexic
Alphabet:
a. Upper (A-Z) and lower case letters (a-z) of the English alphabet
b. Underline character '_'
c. Decimal digits (0-9)

Lexic:
Lexic:
a. Special symbols, representing:
1. Operators
- Arithmetic + - * / %
- Relational == < <= >= > <>
- Logical && || !
- Assignment :=
2. Separators
 : ; space [] {}
3. Reserved words
 if else int char bool while print read struct

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” | “b” | … | “z”
DIGIT = “0” | “1” | ... | “9”
```

c. Constants
1. Integer
```
NONZERODIGIT = "1" | "2" | ... | "9"
NR = "0" | ["+" | "-"] NONZERODIGIT { DIGIT }
```
2. Character
```
CHARACTER = 'CHAR'
```
3. String
```
CONSTCHAR = "STRING"
STRING = CHAR{STRING}
CHAR = LETTER | DIGIT
```

syntax.in

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

```
PROGRAM = "{" STMTLIST "}"
STMT = SIMPLESTMT | STRUCTSTMT
STMTLIST = STMT [";" STMTLIST]
DECLARATIONSTMT = (TYPE IDENTIFIER) | STRUCTDECL
STRUCTDECL = "struct " IDENTIFIER "{" DECLARATION {";" DECLARATION} "}"
TYPE = SIMPLETYPE | ARRAYDECL
SIMPLETYPE = "bool" | "char" | "int"
ARRAYDECL = SIMPLETYPE "[" NR "]"
ASSIGNSTMT = IDENTIFIER ":=" EXPRESSION
IOSTMT = IDENTIFIER ":= read()" | "print(" IDENTIFIER ")"
EXPRESSION = (TERM | EXPRESSION OPERATION EXPRESSION | “(” EXPRESSION
OPERATION EXPRESSION “)”
TERM = IDENTIFIER | NR
OPERATION = “+” | “-” | “*” | “/”
STRUCTSTMT = IFSTMT | WHILESTMT
IFSTMT = "if(" CONDITION ") {" STMTLIST "}" ["else {" STMTLIST "}"]
WHILESTMT = "while(" CONDITION ") {" STMTLIST "}"
```

```
CONDITION = EXPRESSION RELATION EXPRESSION
RELATION = "<" | "<=" | "=" | "<>" | ">=" | ">" | "||" | "&&"
SIMPLESTMT = ASSIGNSTMT | IOSTMT | DECLARATIONSTMT
```

tokens.in

```
:=
+
-
/
*
<
<=
>=
<
>
<>
==
||
&&
(
)
{
}
[
]
\n
\t
space
int
char
bool
string
struct
;
read
print
if
else
while
```