L1b

Tokens.in

```
+
-
*
/
%
=
==
!=
<
>
<=
>=
&&
||
{
}
[
]
(
)
,
;
.
let
if
else
while
```

Lexic.in

```
operator = "+" | "-" | "*" | "/" | "%" | "=" | "==" | "!=" | "<" | ">" | "<=" | ">=" |
"&&" | "||"
separator = "{" | "}" | "[" | "]" | "(" | ")" | "," | ";" | "."
reserved = "let" | "if" | "else" | "while"

zero = "0"
nonzero = "1" | "2" | ... | "9"
digit = zero | nonzero
numLit = zero | (["+"|"-"] nonzero {digit})

letter = "A" | "B" | ... | "Z"
char = letter | digit
charLit = "'" char "'"
stringLit = "\"" char {char} "\""

id = ("_" | letter) {"_" | letter | digit}
```

Syntax.in

```
primType = "Int" | "Char" | "Bool"
array = "[" type "]"
type = primType | array

factor = id | numLit | charLit | stringLit | ("(" expr ")")
term = factor [("*" | "/" | "%") factor]
cmpTerm = term [("+" | "-") term]
logicTerm = cmpTerm [("==" | "!=" | "<" | ">" | "<=" | ">=") cmpTerm]
arrayTerm = logicTerm [("&&" | "||") logicTerm]
expr = arrayTerm | ("[" [arrayTerm {"," arrayTerm} [","]] "]")

readStmt = "read" "(" expr ")"
printStmt = "print" "(" expr ")"
declStmt = "let" id [":" type] "=" expr
assignStmt = id "=" expr
ifStmt = "if" expr compStmt ["else" compStmt]
whileStmt = "while" expr compStmt
stmt = readStmt | printStmt | declStmt | assignStmt | ifStmt | whileStmt | compStmt
stmtList = stmt ";" {stmt ";"}
comptStmt = "{" stmtList "}"

program = stmtList
```