Lexic.txt:

Alphabet:
	- upper and lower case letters of the English alphabet (a-zA-Z)
	- underline character '_'
	- decimal digits (0-9)

Lexic:
	Special tokens, representing:
		- arithmetic operators: cu(+) fără(-) înmulțât cu(*) împărțât la(/) rest(%)
		- relational operators: capătă(=) îi mai mic decât(<) îi oțâră mai mic decât(<=) îi(==) îi oțâră mai mare decât(=>) îi mai mare decât(>) ? și(&&) ori(||)
		- separators: () { } : space
		- reserved words:
			întreg(int) literă(char) zâcală(string) dăi(for) în cazu-n care(if) dacă nu(else) câtă vreme(while) fă(do) citește(read) zâi(write)

	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"

	Constants:
		- întreg:
			<non-zero digit> ::= 1 | ... | 9
			<digit> ::= 0 | ... | 9
			<sign> ::= + |
			<unsigned integer> ::= <non-zero digit> | <unsigned integer> <digit>
			<signed integer> ::= 0 | <unsigned integer> | <sign> <unsigned integer>
		- literă:
			<character literal> := digit | letter
			<character const> := "'" {character literal} "'"
		- zâcală:
			<character> = <letter> | _ | <digit> | <operator> | <separator>
			<characters> = <character> | <characters> <character>
			<string> := \" {character literal} \"

token.in:

[reserved_words]
întreg(int)
literă(char)
zâcală(string)
dăi(for)
în cazu-n care(if)
dacă nu(else)
câtă vreme(while)
fă(do)
citește(read)
zâi(write)
no

```
[operators]
cu(+)
fără(-)
înmulțât cu(*)
împărțât la(/)
rest(%)
capătă(=)
îi mai mic decât(<)
îi oțâră mai mic decât(<=)
îi(==)
îi oțâră mai mare decât(=>)
îi mai mare decât(>)
?
și(&&)
ori(||)

[separators]
(
)
{
}
,
"
'
```

Syntax.in:

<type> ::= întreg | literă | zâcală

<letter> ::= a | … | z | A | … | Z

<digit> ::= 0 | … | 9

<symbol> ::= _

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

<factor> ::= (<expression>) | <identifier> | <constant>

<term operator> ::= înmulțât cu | împărțât la | rest

<term> ::= <term> <term operator> <factor> | <factor>

<expression operator> ::= cu | fără

<expression> ::= <expression> <expression operator> <term> | <term> | <ternary expression>

<condition> ::= <expression> <relational operator> <expression>

<ternary expression> ::= <condition> ? <expression> : <expression>

<declaration statement> ::= no <type> <identifier> | no <type> <identifier> = <expression>

<assignment statement> ::= no <identifier> = <expression>

<io statement> ::= no citește(<identifier>) | no zâi(<identifier>)

<if statement> ::= no în cazu-n care(<condition>) fă {<statement-list>} | no în cazu-n care (<condition>) fă {<statement list>} dacă nu {<statement-list>}

<while statement> ::= no câtă vreme (<condition>) fă {<statement-list>}

<relational operator> ::= capătă | îi mai mic decât | îi oțâră mai mic decât | îi | îi oțâră mai mare decât | îi mai mare decât

<for statement> ::= no dăi (<statement>, <condition>, <statement>) {<statement-list>}

<statement> ::= <declaration statement> | <assignment statement> | <io statement> | <if statement> | <while statement> | <for statement>

<statement-list> ::= <statement> | <statement-list> <statement>

<program> ::= null | <statement-list>