Lab 1 + 2

Github Link

https://github.com/rusuraluca/lftc/tree/main/lftc_lab1%2B2

Lexic.txt

```
Alphabet:
- upper and lower case letters of the English alphabet <letter>
- underline character _
- decimal digits (0-9) <digit>
- operators <operator>
- separators <separator>

Identifiers:
- any combination of letters, or digits that starts with an underscore

Constants:
- integer:
  <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>

- character
  <character literal> := digit | letter
  <character const> := "'" {character literal} "'"

- string
  <character> = <letter> | _ | <digit> | <operator> | <separator>
  <characters> = <character> | <characters> <character>
  <string> := \" {character literal} \"

Special symbols, representing:
- arithmetic operators: + - * / %
- relational operators: = := < <= == => > ?
- separators: () [ ] : ; space ?
- reserved words:
    int char string collection for if while for do elsdo r w
```

token.in

```
[reserved_words]
int
char
string
collection
if
else
for
while
do
r
w
close
[operators]
+
-
*
/
%
=
!
!=
==
+=
-=
/=
*=
<
>
<=
>=
||
&&
?
[separators]
[
]
(
)
{
}
,
;
"
'
```

Syntax.in

<type> ::= int | char | string | collection

<letter> = a | … | z | A | … | Z

<digit> = 0 | … | 9

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

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

<term operator> ::= * | / | %

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

<expression operator> ::= + | -

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

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

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

<declaration statement> ::= <type> <identifier>; | <type> <identifier> = <expression>;

<assignment statement> ::= <identifier> = <expression>;

<io statement> ::= r(<identifier>); | w(<identifier>);

<if statement> ::= if (<condition>) do (<statement-list>) | if (<condition>) do (<statement-list>) elsdo (<statement-list>)

<while statement> ::= while (<condition>) do (<statement-list>)

<relational operator> ::= = | < | <= | == | := | => | >

<for statement> ::= for (<statement>, <condition>, <statement>) do (<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>

p1

```
int _a;
int _b;
```

```
r(_a);
r(_b);

while (_a != _b) do {
    if (_a > _b) do {
        _a = _a - _b;
    } else do {
        _b = _b - _a;
    }
}

w(_a);
```

p2

```
string _unu="a bc";
string _doi = "ab";
char _trei = 'a';
w(_unu);
w(_doi);
w(_trei);
```

p3

```
int _n;
r(_n);

int _sum = 0;

while ( _n != 0) do {
    int _x;
    r(_x);
    _sum = _sum + _x;
    _n = _n - 1;
}

w(_sum);
```

p4

```
int _a = 8;
int _b = 3;
int _c = 10;

int _max = 0;

if (_a > _b) do {
    _max = _a;
} else do {
    _max = _b;
}

if (_c > _max) do {
    _max = _c;
}

w(_max);
```

p1err

```
number =.+ number;
write(number);
```