1. Конкретный синтаксис.

```
digit_nonzero = 1 | .. | 9
digit = 0 | digit_nonzero
alpha = a | ... | z | A | ... Z

nondigit = _ | alpha

alphanum = digit | alpha

prime = digit_nonzero | digit_nonzero digit | digit_nonzero digit prime

ident := nondigit | nondigit alphanum | nondigit alphanum ident
bop = + | .- | ^ | * | / | == | /= | <= | < | > | >= | || | &&

unop = ! | -

aexpr := aexpr bop aexpr
      | uop aexpr
      | var
      | prime
      | (aexpr)


arg := ident

marg := arg
      | (arg)

args := marg args
      | arg

expr' := aexpr
       | let

expr := expr' | (expr')

bind := ident args = expr

let := 'let' bind 'in' expr | (let)

terminate := ';'

program := main = expr
```

Точка входа: функция main без аргументов.

2. Абстрактный синтаксис

Представлен в виде АСД.

```
data Ast = Bind

data Bind = Bind Name Args Expr

data Args = [Var String]

data Expr = AExpr | Let Bind Expr

data AExpr = BinOp BinOperator Expr Expr
           | UnOP  UnOperator Expr
           | Primary Int
           | Var String
```