NGG

Aleksandr Dremov

Syntax

Main

```
  Start            ::= {FuncDecl | VarDefStmt}*
```

Identifier

```
Identifier       ::= Identifier[Character {Character | Number}*]
Number           ::= Number[Digit+]
Digit            ::= "regexp:[0-9]"
Character        ::= EngCharacter | '_'
EngCharacter     ::= "regexp:[a-zA-Z]"
```

Function essentials

```
FuncDecl         ::= FDecl['never gonna'] Identifier
                     LPA['('] ArgumentsList? RPA[')'] BlockStmt
ArgumentsList    ::= Identifier {Comma[','] Identifier}*
CallList         ::= rValue {Comma[','] rValue}*
```

Values

```
rValue           ::= AddSubExpr {(Eq['=='] | Leq['<='] | Geq['>=']
                    | Neq['!='] | Gr['>']| Le['<']) AddSubExpr}?
AddSubExpr       ::= MulDivExpr
                     {(Plus['+'] | Minus['-']) MulDivExpr}*
MulDivExpr       ::= UnaryExpr {(Mul['*'] | Div['/']) UnaryExpr}*
UnaryExpr        ::= ((Plus['+'] | Minus['-']) PrimaryExpr)
                     | PrimaryExpr
PrimaryExpr      ::= LPA['('] rValue RPA[')'] | Number | FuncCall |
                     Identifier
FuncCall         ::= Identifier LPA['('] CallList? RPA[')']
```

Blocks

```
BlockStmt        ::= BStart['strangers'] Statement* BEnd['to love']
Statement        ::= (VarDef | Print | PrintLine | AssignExpr |
                      FuncCall)  StEnd['bdum']
                      | ReturnStmt | IfStmt | WhileStmt | BlockStmt

AssignExpr       ::= Identifier (Assg['='] | AdAssg['+='] |
                      MiAssg['-=']
                     | MuAssg['*='] | DiAssg['/=']) rValue
VarDef           ::= VDecl['make you'] (Identifier Assg['=']
                     rValue |  Identifier )
VarDefStmt       ::= VarDef StEnd['bdum']
Print            ::= Print['goodbye']     rValue
PrintLine        ::= PrintL['desert you'] rValue
```

Control Sequences

```
IfStmt           ::= If['you know the rules']
                     LPA['('] rValue RPA[')']
                     BlockStmt {Else['and so do I'] BlockStmt}?
WhileStmt        ::= While['run around']
                     LPA['('] rValue RPA[')'] BlockStmt
ReturnStmt       ::= Return['known each other for so long'] rValue
                     StEnd['bdum']
```

Memory management

```
… | global variables | stack memory
```

Use	Register
Bp register	rex
Tmp calculations	rax
Tmp calculations2	rbx