Студент: Никита Сазанович
Группа: АУ 302

Домашнее задание #9

***Задача* 1.** *Предложить удобный конкретный синтаксис языка L для приведенного абстрактного.*

Доказательство.

Ничего не описывает синтаксис лучше, чем сам формат для генерации *Antlr* – (оставлены только правила для парсера – токены опущены):

```
file
    : (function)* block EOF
    ;

block
    : (statement)*
    ;

blockWithBraces
    : L_CURLY_BRACE block R_CURLY_BRACE
    ;

statement
    : assignment SEMICOLON
    | readCall SEMICOLON
    | writeCall SEMICOLON
    | returnStatement SEMICOLON
    | functionCall SEMICOLON
    | whileBlock
    | ifStatement
    ;

function
    : FUNCTION IDENTIFIER L_BRACE parameterNames R_BRACE blockWithBraces
    ;

parameterNames
    : (IDENTIFIER (COMMA IDENTIFIER)*)?
    ;

assignment
    : IDENTIFIER ASSIGN expression
    ;

readCall
    : READ L_BRACE IDENTIFIER R_BRACE
    ;

writeCall
    : WRITE L_BRACE expression R_BRACE
    ;

returnStatement
    : RETURN expression
    ;

functionCall
    : IDENTIFIER L_BRACE arguments R_BRACE
    ;
```

```
50
51 arguments
52     : (expression (COMMA expression)*)?
53     ;
54
55 whileBlock
56     : WHILE L_BRACE expression R_BRACE blockWithBraces
57     ;
58
59 ifStatement
60     : IF L_BRACE expression R_BRACE THEN blockWithBraces (ELSE blockWithBraces)?
61     ;
62
63 expression
64     : lorExpression
65     ;
66
67 lorExpression
68     : landExpression (LOR landExpression)*
69     ;
70
71 landExpression
72     : equivalenceExpression (LAND equivalenceExpression)*
73     ;
74
75 equivalenceExpression
76     : relationalExpression ((EQ | NQ) relationalExpression)*
77     ;
78
79 relationalExpression
80     : additiveExpression ((GT | LT | GTE | LTE) additiveExpression)*
81     ;
82
83 additiveExpression
84     : multiplicativeExpression ((PLUS | MINUS) multiplicativeExpression)*
85     ;
86
87 multiplicativeExpression
88     : atomicExpression ((MULTIPLY | DIVIDE | MODULUS) atomicExpression)*
89     ;
90
91 bracedExpression
92     : L_BRACE expression R_BRACE
93     ;
94
95 atomicExpression
96     : bracedExpression
97     | functionCall
98     | IDENTIFIER
99     | NUMBER
100    ;
```

.

Собственно, это синтаксис предполагает, что язык будет игнорировать *whitespace*. Для ограничения блоков, используются фигурные скобки.

Также, в этом синтаксисе разделение выражений происходит с помощью точек с запятыми. Если statement содержит блок кода (if statement и while statement), то точка с запятой не нужна.

Также, были оставлены скобки и запятые для аргументов и параметров процедур.

Теперь, когда вызовы функций – это выражения, мы немного изменили синтаксис. Вместо ключевого слова *proc* стали исользовать fun. Чтобы указать, какое значение следует вернуть, было введено ключевое слово *return* и соответсвтующий ему *returnStatement*.

Чтобы ознакомиться с полным набором правил для лексера и парсера, обратитесь к L.g4.

□