

L

Любая программа на языке L - набор statement, которые выполняются последовательно. Statement делятся на несколько типов:

- (1) объявление функций и переменных
- (2) присваивание переменных
- (3) выражение в скобках или вызов функции
- (4) управляющие конструкции
- (5) input-output

После всех statement (кроме управляющих конструкций) обязательно должны идти точка с запятой (;)

Объявление функций:

```
1 fun <FUNCTION_NAME> (<PARAM_NAME1>, <PARAM_NAME1>, ...) {  
2     <STATEMENT_1>  
3     <STATEMENT_2>  
4     <STATEMENT_3>  
5     ...  
6 }
```

При этом множество параметров и statement может быть пустым, а название функции должно содержать только цифры, латинские буквы и нижнее подчеркивание, и при этом не должно начинаться с цифры.

Объявление переменных:

```
1 var <VAR_NAME>
```

или

```
1 var <VAR_NAME> = <EXPRESSION>
```

При этом на имена переменных действуют те же ограничения, что и на имена функций.

Присваивание переменных:

```
1 <VAR_NAME> = <EXPRESSION>
```

Управляющие конструкции:

Циклы:

```
1 while (<EXPRESSION>) {  
2     <STATEMENTS>  
3 }
```

Условные операторы:

```
1 if (<EXPRESSION>) {  
2   <STATEMENTS>  
3 }  
4 if (<EXPRESSION>) {  
5   <STATEMENTS>  
6 } else {  
7   <OTHER_STATEMENTS>  
8 }
```

Выражения (expression):

(Вставка кода из antlr с комментариями, т.к. кажется, что это наиболее удобный для чтения формат):

Общее описание выражения

```
1 expression: binaryExpression | '(' expression ')' | atomicExpression;
```

```
1 functionCall: Identifier '(' (arguments)? ')';  
2 arguments: expression (',' expression)*;
```

Примитивное выражение, имеет наивысший приоритет
Здесь Identifier - название переменной, а Literal - число

```
1 atomicExpression: functionCall | Identifier | Literal;
```

```
1 binaryExpression: eqExpression | boolExpression | mdExpression | pmExpression;
```

Каждое из последующих видов бинарных выражений является праворекурсивным и, следовательно, однозначным

Так же для каждого типа выражения строится Helper, который содержит все возможные выражения с более высоким приоритетом

```
1 boolExpression: boolExpressionHelper BoolOp (boolExpressionHelper | boolExpression);  
2 boolExpressionHelper: eqExpressionHelper | eqExpression;  
3  
4 eqExpression: eqExpressionHelper EqOp (eqExpressionHelper | eqExpression);  
5 eqExpressionHelper: pmExpressionHelper | pmExpression;  
6  
7 pmExpression: pmExpressionHelper PmOp (pmExpressionHelper | pmExpression);  
8 pmExpressionHelper: mdExpressionHelper | mdExpression;  
9  
10 mdExpression: mdExpressionHelper MdOp (mdExpressionHelper | mdExpression);  
11 mdExpressionHelper: atomicExpression | '(' expression ')';
```

Input-Output:

```
1 read <VAR_NAME>
```

```
1 write <EXPRESSION>
```