

LFPC Lab №3

Syntax definition:

```
1 // var decs
2 int i = 10;
3 string s = "abc";
4 double d = 1.2;
5
6 // function dec
7 fun add(int i1, int i2) ret int {
8     return i1 + i2;
9 }
10
11 // recursive functio
12 fun fib(int n) ret int {
13     if (n == 0) return 0;
14     if (n == 1) return 1;
15     else fib(n - 1) + fib(n - 2);
16 }
17
18 int i = add(2, 1);
19
20 // arrays
21 array<string> a = {"a", "b", "c"};
22 string s = a[0];
```

Code example:

```
1 // code example
2 fun addValues(array<int> a) ret int {
3     int sum = 0;
4     for (int i = 0 to length(a); i++) {
5         sum += a[i];
6     }
7 }
8
9 fun main() ret void {
10    array<int> a = {1, 2, 3, 5};
11
12    int sum = addValues(a);
13 }
```

Grammar of the language:

Table 1: EBNF meta notation

$\langle x \rangle$	means x is non-terminal
x or $'x'$	means x is a terminal
$[x]$	means x is optional (0 or 1 occurrences of x)
x^*	means 0 or more occurrences of x
x^+	means 1 or more occurrences of x
$ $	separates alternatives
$\{ \}$ and $\{ \}$	are used for grouping alternatives

The grammar of the DSL is $G = (V_N, V_T, S, P)$ where:

$V_N = \{ \langle \text{program} \rangle, \langle \text{statements} \rangle, \langle \text{statement} \rangle, \langle \text{nosemicolon_statement} \rangle, \langle \text{semicolon_statement} \rangle, \langle \text{ctrlflow_statement} \rangle, \langle \text{block} \rangle, \langle \text{comment} \rangle, \langle \text{return_statement} \rangle, \langle \text{expression} \rangle, \langle \text{assignment} \rangle, \langle \text{for_statement} \rangle, \langle \text{if_statement} \rangle, \langle \text{declaration} \rangle, \langle \text{function_dec} \rangle, \langle \text{parameter} \rangle, \langle \text{variable_dec} \rangle, \langle \text{variable_init} \rangle, \langle \text{type} \rangle, \langle \text{scalar_type} \rangle, \langle \text{multid.type} \rangle, \langle \text{function_call} \rangle, \langle \text{prefix_expression} \rangle, \langle \text{infix_expression} \rangle, \langle \text{postfix_expression} \rangle, \langle \text{bracket_expression} \rangle, \langle \text{paranthesis_expression} \rangle, \langle \text{identifier} \rangle, \langle \text{number} \rangle, \langle \text{integer} \rangle, \langle \text{double} \rangle, \langle \text{character} \rangle, \langle \text{string} \rangle, \langle \text{digit} \rangle, \langle \text{nonzero_digit} \rangle, \langle \text{operators} \rangle, \langle \text{infix_op} \rangle, \langle \text{postfix_op} \rangle, \langle \text{prefix_op} \rangle \}$,

$V_T = \{ ';', //, [,], \{, \}, (,), ', +, -, ++, --, +=, -=, !, ==, <=, >=, >, <, \%, *, /, \text{array}, \text{int}, \text{double}, \text{break}, \text{for}, \text{if}, \text{else}, \text{void}, \text{ret}, \text{fun}, \text{'_'}, \text{a}, \text{b}, \text{c}, \dots \text{z}, \text{A}, \text{B}, \dots \text{Z}, 0, 1, \dots 9 \}$,

$S = \langle \text{program} \rangle,$

$P = \{$

STATEMENTS:

$\langle \text{program} \rangle ::= \langle \text{statements} \rangle^*$

$\langle \text{statements} \rangle ::= \langle \text{statement} \rangle^*$

$\langle \text{statement} \rangle ::= \langle \text{nosemicolon_statement} \rangle \mid \{ \langle \text{semicolon_statement} \rangle \text{' ;' } \}$

$\langle \text{nosemicolon_statement} \rangle ::= \langle \text{ctrlflow_statement} \rangle \mid \langle \text{block} \rangle \mid \langle \text{comment} \rangle$

$\langle \text{semicolon_statement} \rangle ::= \langle \text{declaration} \rangle \mid \langle \text{return_statement} \rangle \mid \langle \text{expression} \rangle \mid \langle \text{assignment} \rangle \mid$

break

$\langle \text{return_statement} \rangle ::= \text{return } \langle \text{expression} \rangle$

$\langle \text{ctrlflow_statement} \rangle ::= \langle \text{for_statement} \rangle \mid \langle \text{if_statement} \rangle$

$\langle \text{for_statement} \rangle ::= \text{for } (\langle \text{variable_dec} \rangle \text{ to } \langle \text{expression} \rangle ; \langle \text{expression} \rangle) \langle \text{statement} \rangle$

$\langle \text{if_statement} \rangle ::= \text{if } (\langle \text{expression} \rangle) \langle \text{statement} \rangle [\langle \text{else_statement} \rangle]$

$\langle \text{else_statement} \rangle ::= \mathbf{else} \{ \langle \text{if_statement} \rangle \mid \langle \text{statement} \rangle \}$

$\langle \text{comment} \rangle ::= \text{'//'} \langle \text{character} \rangle^*$

$\langle \text{block} \rangle ::= \{ \langle \text{statements} \rangle^* \}$

$\langle \text{return_type} \rangle ::= \langle \text{type} \rangle \mid \mathbf{void}$

$\langle \text{assignment} \rangle ::= \langle \text{identifier} \rangle = \langle \text{expression} \rangle$

DECLARATIONS:

$\langle \text{declaration} \rangle ::= \langle \text{function_dec} \rangle \mid \langle \text{variable_dec} \rangle$

$\langle \text{function_dec} \rangle ::= \mathbf{fun} \langle \text{identifier} \rangle (\langle \text{parameter} \rangle^*) \mathbf{ret} \langle \text{return_type} \rangle \langle \text{block} \rangle$

$\langle \text{parameter} \rangle ::= \langle \text{type} \rangle \langle \text{identifier} \rangle [\text{'}, \text{'}]$

$\langle \text{variable_dec} \rangle ::= \langle \text{type} \rangle \langle \text{identifier} \rangle [\langle \text{variable_init} \rangle]$

$\langle \text{variable_init} \rangle ::= \text{'='} \langle \text{expression} \rangle$

DATA TYPES:

$\langle \text{type} \rangle ::= \langle \text{scalar_type} \rangle \mid \langle \text{multidim_type} \rangle$

$\langle \text{scalar_type} \rangle ::= \mathbf{int} \mid \mathbf{double} \mid \mathbf{string}$

$\langle \text{multidim_type} \rangle ::= \mathbf{array} \langle \langle \text{scalar_type} \rangle \rangle \langle \text{identifier} \rangle$

EXPRESSIONS:

$\langle \text{expression} \rangle ::= \langle \text{identifier} \rangle \mid \langle \text{number} \rangle \mid \langle \text{prefix_expression} \rangle \mid \langle \text{infix_expression} \rangle \mid \langle \text{postfix_expression} \rangle \mid \langle \text{bracket_expression} \rangle \mid \langle \text{parenthesis_expression} \rangle \mid \langle \text{cbracket_expression} \rangle \mid \langle \text{function_call} \rangle$

$\langle \text{function_call} \rangle ::= \langle \text{identifier} \rangle (\{ \langle \text{expression} \rangle \text{'}, \text{'}}^*)$

$\langle \text{prefix_expression} \rangle ::= \langle \text{prefix_op} \rangle \langle \text{expression} \rangle$

$\langle \text{postfix_expression} \rangle ::= \langle \text{expression} \rangle \langle \text{postfix_op} \rangle$

$\langle \text{infix_expression} \rangle ::= \langle \text{expression} \rangle \langle \text{infix_op} \rangle \langle \text{expression} \rangle$

$\langle \text{parenthesis_expression} \rangle ::= (\langle \text{expression} \rangle)$

$\langle \text{cbracket_expression} \rangle ::= \{ \langle \text{expression} \rangle \}$

$\langle \text{bracket_expression} \rangle ::= [\langle \text{expression} \rangle] \text{'['} \langle \text{expression} \rangle \text{'}'}$

IDENTIFIERS, NUMBERS, OPERATORS:

```

<identifier> ::= <character> {<character> | <digit>}*
<number> ::= <integer> | <double>
<integer> ::= <nonzero_digit><digit>
<double> ::= <integer> '.' <digit>*
<character> ::= a | b | c ... z | A | B ... Z | _
<digit> ::= 0 | 1 | ... | 9
<nonzero_digit> ::= 1 | ... | 9
<operators> ::= <infix_op> | <prefix_op> | <postfix_op>
<infix_op> ::= + | - | && | || | % | == | / | * | < | <= | > | >= | , | ;
<prefix_op> ::= ++ | -- | !
<postfix_op> ::= ++ | - | , | ;
}

```

Note: <string> is any combination of UTF-8 characters surrounded by double quotes.