Rusnac Ana-Maria, FAF-202

LFPC, Laboratory Work Nr. 3
Grammar Definition

Meta-Notations

<foo>	foo is non-terminal
foo	**foo** is terminal
x*	zero or more occurrences of x
x+	one or more occurrences of x
|	separates alternative
[x]	zero or one occurrence of x note: brackets is quotes '[', ']' are terminals
{x}	zero or more occurrences of x
(x | y)	one of either x or y

V_N = {<function>, <var>, <forLoop>, <ifStatement>, <declName>, <letter>, <digit>, <operators>, <body>, <varDecl>, <controlStructure>, <intLiteral>, <floatLiteral>, <variable>, <array>, <expression>, <constExpression>, <relExpression>, <varName>, <arithOpreator>, <relOperator>, <logicalOperator>, <stepExpression>, <incrOperator>, <functionCall>, <return>}

V_T = {A…Z, a…z, 0, 1…9, (,), [,], ", ', ;, =, +, -, /, *, %, ++, --, <, >, <=, >=, ==, !=, &&, ||, !, @, #, $, ?, {, }, ~, Function, var, EndFunction, For, Next, If, Else, EndIf, return}

P = {<program> → <function>
<function> → **Function** <declName>(<var>*) <body> **EndFunction** <function>*
<declName> → <letter> {, <letter> | <digit>}
<var> → <variable> | <array>
<variable> → **var** <letter> {**,** <letter> | <digit>}
<array> → <variable>**[]**
<body> → {<varDecl> | <controlStructure> | <functionCall> | <assignment> | <return>}
<varDecl> → <variable> [= <literal>]
| <array> [= '**[**' <intLiteral>$^+$ | <floatLiteral>$^+$ '**]**']
<literal> → <intLiteral> | <floatLiteral>
<intLiteral> → <digit> <digit>*
<floatLiteral> → <digit>$^+$. <digit>$^+$
<controlStructure> → <ifStatement> | <forLoop>
<ifStatement> → **If(**<expression>**)** <body> [**else** <body>] **EndIf**
<forLoop> → **For**(<variable>; <expression>; <expression>) <body> **Next**
<expression> → <constExpression> | <relExpression> | <stepExpression>
<constExpression> → (<intLiteral> | varName) <arithOperator> (<intLiteral> | varName)
{<arithOperator> (<intLiteral> | varName)}

| (<floatLiteral> | <varName>) <arithOperator> (<floatLiteral> | <varName>) {<arithOperator> (<floatLiteral> | <varName>)}
<relationalExpression> → (<intLiteral> | varName) <relOperator> (<intLiteral> | varName) {<logicalOperator> <relationalExpression>}
| (<floatLiteral> | <varName>) <relOperator> (<floatLiteral> | <varName>) {<logicalOperator> <relationalExpression>}
<stepExpression> → <varName> <incrOperator> | <incrOperator> <varName>
<varName> → <letter> {, <letter> | <digit>}
<functionCall> → <declName> ([<varName> {, <varName>}])
<assignment> → <varName> = <expression>
<return> → return <literal> | <varName> | <constExpression>
<letter> → **a | b | … | z | A | B | … | Z**
<digit> → **0 |1 | 2 | … | 9**
<arithOperator> → **+ | - | * | / | %**
<relOperator> → **< | > | <= | >= | == | !=**
<logicalOperator> → **&& | ||**
<incrOperator> → **++ | --**
<char> → **a | b | … | z | A | B | … | Z | 0 |1 | 2 | … | 9 | ! | # | $ | % | & | (|) |+ | - | * | / | < | > | = | ? | @ | [|] | { | } | ^ | ~**
}

Sample Code

```
Function main()
var arr[] = [5.7, 30, 45, 43.7, 12];
var n = 5;
largestNum(arr, n);
return 0;
EndFunction

Function largestNum(var array[], var size)
var max = array[0];
  For(var i = 1; i < size; i++)
    If(array[i] > max)
      max = array[i];
    EndIf
  Next
EndFunction
```