SER 502 Project Team 12, Spring 2022

Authors: Achuth Reddy Rajula, Varshik Sonti, Rajiv Kashyap Jalakam, Rahul Vuppla, RajaLaxman Thakkalapelli
Language Name: Venom
Language extension: .venom

Lexical Analysis:

- The source code goes through lexical analysis as the first step. The code is split into tokens which we are going to implement using Python.

Parsing:

- This step checks whether the code has the correct syntax whose rules we defined in the grammar
- We are going to be using a Top-down parser here

Interpreter:

- Prolog is the language that we will be using for our interpreter
- A parse tree is given as an input here, which is evaluated
- The data structure used here is list

Language Design - Venom

The language that we will be designing will handle all of the following features:

1. **Boolean:**
   As per the requirement, our language will support the following boolean operators
   - AND
   - OR

- NOT

2. **Data types:**
   We are going to support the following three data types
   - Integer
   - Float
   - String

3. **Loops:**
   We are going to handle the following loops. The loops will run as long as the condition that they operate in is true and exit the loop once it is false
   - For
   - While
   - For-Range

4. **Conditional Statements**
   As mentioned in the requirements, our language will handle the basic conditionals as mentioned below
   - If-Then-Else
   - Ternary operator - ‘#’ and ‘>>’

5. **Display**
   Will implement the print logic using the following command
   - flash (for displaying all specified identifier values)

6. **Operators**
   Our language will be able to handle the following basic arithmetic and comparison operators like addition, subtraction, multiplication, division, equals
   - ‘+”, ‘ -’, ‘*’, ‘/’ ,’^’, ‘%’ (Arithmetic operations)
   - ‘>’, ‘<’, ‘==’, ‘~=’, ‘<=’, ‘>=’ (comparison operators)
   - ‘=’ (Assignment operator)

7. **Comments**

- Our language uses “$” for writing comments

Grammar for Venom Programming Language:

Integer → 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 0
String → ‘[a-zA-Z0- 9]$’
Float →
Char →
Boolean → ‘True’ | ‘False’ |
LogicalOperator → ‘AND’ | ‘OR’ | ‘NOT’
Identifiers → '^[A-Za-z_][A-Za-z0-9_]*'
DataType → int | float | string | boolean
Addition → ‘+’
Subtract → ‘-’
Multiplication → ‘*’
Division → ‘/’
Assignment → ‘=’
Modulus → ‘%’
Square → “^”
Comparison → ‘==’ | ‘<’ | ‘<=’ | ‘>’ | ‘>=’ | ‘~=’
LogicalOperator → and | or | not
If → ‘if’
Else → ‘else’
ElseIf → ‘elseIf’
For → ‘for’
While → ‘while’
In → ‘in’
Range → ‘range’
Comment → ‘$’
StartQuote → ‘ “ ‘
EndQuote → ‘ “ ’
OpenParenthesis → ‘ (‘
CloseParenthesis → ‘) ’
StartBlock → ‘ { ‘

EndBlock → ‘ } ’
Colon → ‘ : ‘
Comma → ‘ , ’
SemiColon → ‘ ; ‘
Begin → ‘start’
End → ‘end’
Print → ‘flash’

<program> → <Begin> <statement> <End>

<statement> → <declaration> <statement>
| <assign> <statement>
| <if> <statement>
| <if_else> <statement>
| <while> <statement>
| <for> <statement>
| <for_range> <statement>
| <statement>

<declaration> → <Datatype> <Space> <Identifier> <Space> <Assignment> <Space>
<declaration_helper>
| <Datatype> <Space> <Identifier>

<declaration_helper> → <expression> | <value>

<assign> → <Identifier> <Assignment> <expression>

<print> → <Print> <Space> <StartQuote> <String> <EndQuote>
| <Print> <Space> <Identifier>

<if> → <If> <OpenParenthesis> <condition> <CloseParenthesis> <StartBlock> <statement>
<EndBlock>

<else_if> → <ElseIf> <OpenParenthesis> <condition> <CloseParenthesis> <StartBlock>
<statement> <EndBlock>

<else> → <Else> <StartBlock> <statement <EndBlock>

<if_else> → <if> <if_else_helper>

<if_else_helper> → <else_if> | <else> | <else_if> <else>

<while> → <While> <OpenParenthesis> <condition> <CloseParenthesis> <StartBlock> <statement> <EndBlock>

<for> → <For> <OpenParenthesis> <assign> <SemiColon> <Identifier> <Comparison> <expression> <SemiColon> <CloseParenthesis> <StartBlock> <statement> <Endblock>
    | <For> <OpenParenthesis> <assign> <SemiColon> <Identifier> <Comparison> <expression> <SemiColon> <expression> <CloseParenthesis> <StartBlock> <statement> <Endblock>

<for_in_range> → <For> <Identifier> <In> <Range> <OpenParenthesis> <expression> <Comma> <expression> <CloseParenthesis> <StartBlock> <statement> <Endblock>

<condition> → <Identifier> <Space> <Comparison> <Space> <expression>
    | <Identifier> <Space> <Comparison> <Space> <expression> <Space> <LogicalOperator> <Space> <condition> | <Boolean>

<comment> → <Comment> <String>

<expression> → <expression> <operator> <expression>
    | <expression_helper> <expression>

<expression_helper> → <OpenParenthesis> <expression> <CloseParenthesis>
    | <Identifier> | <value>

<value> → <Integer> | <Float> | <Char> | <String> | <Boolean>