CS315-Project 2

Group Members

Ayşegül Gökçe - 21602330 - Section 1
Aziz Utku Kağıtçı - 21602869 - Section 2
Nogay Evirgen - 21601881 - Section 3

Course Name: CS315 - Programming Languages
Group Number: 17
Language Name: nua
Date: 7.11.2019

BNF Grammer

<nua> -> <statements>

<statements> -> <statement> | <statement><statements>

<statement> -> <while statement> | <for statement> | <if statement> | <assignment> | <define function> | <connect url> | <call function> | <comment>

<prim function> -> <identifier>()

<connect url> -> connectUrl("<url>") | connectUrl(<string name>)

<url> -> <extended alphanumerics>

<send integer> -> <connect url>.sendInteger(<identifier>) | <connect url>.sendInteger(<name>) | <connect url>.sendInteger(<digits>)

<receive integer> -> <identifier> = <connect url>.receiveInteger()

<assignment> -> <name> = <expr>

<expr> -> <expr> <addition> <term> | <expr> <subtraction> <term> | <term>

<addition> -> +

<subtraction> -> -

<multiplication> -> *

<division> -> /

<term> -> <term> <multiplication> <factor> | <term> <division> <factor> | <factor>

<factor> -> (<expr>) | <name> | <digits> | <call function> | <prim function>

<if statement> → if (<logical expressions>) {<stmt>} | if (<logical expressions>) {<stmt>} else {<stmt>}

<and or> -> && | ||

<comparison> -> == | != | > | < | >= | <=

<logical expression> -> <switch> | <identifier> <comparison> <identifier> | <identifier> <comparison> <digits>

<logical expressions> -> <logical expression> | (<logical expression>) <and or> <logical expressions>

<while statement> -> while(<logical expressions>){<statements>}

<define function> -> func <alphanumerics>(<params>){<statements> return <identifier>; } | func <alphanumerics>(<params>){<statements>}

<params> -> <param> | <param>,<params> | <space>

<param> -> <identifier> | <switch> | <call function>| <string name> | <integer name>

<double name> -> d<alphanumerics>

<call function> -> <name>(<params>)

<string name> -> c<alphanumerics>

<integer name> -> n<alphanumerics>

<double name> -> d<alphanumerics>

<method name> -> m<alphanumerics>

<alphanumerics> -> <alphanumerics> <digits> | <alphanumerics><letters>
| <letters> | <digits>

<extended alphanumerics> -> <extended alphanumerics><alphanumerics>
| <extended alphanumerics><sign> | <alphanumerics> | <sign>

<spaces> -> <space> | <space> <spaces>

<letters> -> <letter> | <letter> <letters>

<digits> -> <digit> | <digit> <digits>

<signs> -> <sign> | <sign> <signs>

<comment> -> /-- <commentLines> --/

<escapeChars> -> <escapeChar> | <escapeChar> <escapeChars>

<commentLines> -> <commentLine> | <commentLine> <commentLines>

<commentLine> -> <letters> | <digits> | <signs> | <escapeChars> | <spaces>

<identifier> -> temperature | humidity | airCondition| airQuality| light | soundLevel | <switch> | timestamp

<switch> -> switch1 | switch2 | switch3 | switch4 | switch5 | switch6 | switch7 | switch8 | switch9 | switch10

<escapeChar> -> \n | \r | \t | \b | \f | \v | \0

<letter> -> A | B | C | D | E | F | G | H | I | J | K | L | M | N | O | P | Q | R | S | T | U | V | W | X | a | b | c | d | e | f | g | h | i | j | k | l | m | n | o | p | q | r | s | t | u | v | w | x | y | z

<sign> -> ! | \" | # | $ | % | & | \' | (|) | * | + | , | - | . | / | : | ; | < | = | > | ? | @ | [| \\ |] | ^ | _ | ` | { | \| | } | ~

<digit> -> 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9

<space> ->

<reservedWords> ->if | else | while | func | connectUrl | receiveInteger | sendInteger | <identifier>

1. **Primitive functions: read data from each sensor, timestamp from timer.**

   User receives the sensor and switch values by calling prim functions of those switches and sensors such as temperature() and switch1().

2. **Mechanism to define a connection to a given URL.**

   When the user wants to connect a url, the function connectUrl(), predefined by operating system, should be called with a string or a string variable.

3. **Mechanisms to send/receive an integer at a time, from/to a connection.**

   When the user wants to send or receive an integer from or to a connection, whether they are already connected or not, they first need to call connectUrl() function with the url that they want to connect and then follow it by calling sentInteger() or receiveInteger() functions. connectUrl() and receiveInteger()/sentInteger() functions should be seperated with a “.” .The integer to be send might be an integer value, an identifier or variable.

4. **Variable names**

   Variable names can only include uppercase characters, lower case characters and digits and it has to start with a specific letter which specifies the type of the variable. Integer variables start with letter “n” , real variables start with letter “d”, string variables start with letter “c”.

5. **Assignment operator, arithmetic operators**

The language only supports the assignment operator(=) and the four basic arithmetic operations: addition (+), subtraction(-), multiplication(*) and division(/). We minimal feature multiplicity. There is no ++ or -- operators. Hence it increases readability of our language

6. **Arithmetic expressions**

The parse tree is created according to the mathematical order of operations including parentheses. In that way we increase reliability of our language.

7. **Conditional statements; e.g., if-then, if-then-else**

There are two ways to use if statements. First way is a if-then statement, in this one user writes “if” keyword followed by a left-right brackets which includes a logical expression. If the logical expression is true, statements in the body will be executed. If not, nothing happens. Second way is a if-then-else-then statement, in this one user writes “if” keyword followed by a left-right brackets which includes a logical expression. If the logical expression is true, statements in the body will be executed. If not, statements in the body of else will be executed. The BNF grammar for if-then, if-then-else causes different parse trees however we will deal with that in the yacc. So, It’s not a problem.

8. **Loops; e.g., for, while**

User can use loops with while statement or for statement. While statement takes a logical expression to loop the execution of body of the while statement until logical statement becomes false. For statement takes three parts, parts are divided with a semicolon. In first part there is a control variable, in second part there is a logical expression and in the third part control variable gets updated. For loop executes the body of the for statement as long as logical expression is true.

9. **Mechanisms for defining and calling functions**

In order to define a function, user must type func followed by a non-reversed alphanumeric keyword which starts with the letter “m”. Then, user can choose to add one parameter, multiple parameters or no parameter at all. User should add parameters in a left and a right bracket which comes after functions name. If there is only one parameter, user should just put that parameter. If there are multiple parameters, user should separate these parameters with semicolons. If there are no parameters, user shouldn’t write anything in brackets. To call a function, user must type name of the function then follow it by it’s parameters if that functions has any parameter.

10. Comments

Users can add comments to their program by writing the things they want to between “/--” and “--/”. Whether it’s a single-line comment or a multi-line comment, this is the format we use in our language. Since there’s only one way to add a comment, it is writable and readable.