LEXICAL AND SYNTAX ANALYZER - SUBHASH PATHURI

Rule

Code starts with `begin` and ends with `end`. Each lexeme is seperated by a space.

Statements

* Assignment
* Declaration
* Condition
* Loop

TOKENS

Mathematical Operations

|  |  |  |
| --- | --- | --- |
| TOKEN CODE | Operation | Regex |
| ADD | + | + |
| SUB | - | - |
| MUL | \\* | \\* |
| DIV | / | / |
| OPEN | ( | ( |
| CLOSE | ) | ) |

Comparing

|  |  |  |
| --- | --- | --- |
| Token Code | Operation | Regex |
| Less | < | < |
| Great | > | > |
| Lessequal | <= | <= |
| Greatequal | >= | >= |
| Equal | == | == |
| Notequal | != | != |
|  |  |  |

Integer Types

|  |  |
| --- | --- |
| Token code | Size |
| I | 1 byte |
| II | 2 bytes |
| III | 4 bytes |
| IIII | 8 bytes |

|  |  |
| --- | --- |
| Token code | Regex |
| VARIABLE | [a-zA-Z]\_{6,8} |
| TASK | task |
| REPEAT | repeat |
| BEGIN | begin |
| END | end |

Keywords

Miscellaneous

|  |  |  |
| --- | --- | --- |
| Token Code | Operation | Regex |
| ASSIGNMENT | = | = |
| CODESTARTS | { | { |
| CODEENDS | } | } |
|  |  |  |

Priority order

* ()
* -
* +
* /
* \*
* %

Production Rules

```txt

<Program> --> begin <stmt\_list> end

<stmt\_list> --> {<stmt> `;`}

<stmt> --> <if\_stmt> | <while\_stmt> | <as\_s> | <declaration>

<if\_stmt> --> task <bool> `{` { <stmt> ';'} `}`

<while\_stmt> --> repeat `{` <bool> { <stmt> ';' } `}`

<as\_s> --> <var> `=` <expression> `;`

<declaration> --> <datatype> <var> `;`

<datatype> --> (I|II|III|IIII)

<var> --> [a-zA-Z\_]{6,8} // Variable Naming restriction

<expression> --> <term> { (`\*`|`\` ) <term> }

<term> --> <term> { (`+`|`-`) <term> }

<factor> --> [0-9]+ | <var> | `(` <expression> `)`

<bool> --> <expression> (`<=`|`>=` | `<` | `>`) <expression>

E -> E + T Expression + Term

E -> E - T Expression - Term

E -> T Some expression can be a term

T -> T \* F Term \* Factor

T -> T / F Term / Factor

T -> F Some Terms can be Factors

F -> -F Unary Minus

F -> +F Unary Plus

F ->( E ) Factor can be an Expression in parentheses

F -> c Factor can be a constant

```

TEST CASES

begin

II varOne;

varOne = 1;

IIII varTwo;

varTwo = 1 + 1 + (2 + (100 \* 2)) \* varOne;

task (varOne == varTwo) {

task (varOne <= varTwo) {

varOne = varTwo \* 2;

}

}

I varThree;

varThree = 0;

repeat (varThree <= 2) {

varThree = varThree + 4;

task (varThree == 3) {

varThree = varThree \* 2;

}

}

End

Start

IIII varOne;

varOne = 2 \* 4 - 1;

I varTwo;

varTwo = 4;

L var\_a;

task (varTwo < varOne){

var\_a = 1;

}

End

Put the relative path in the index.py

I put the test cases in a Case tests file

C) It is following top – Down automata

D) I think the LR table doesn’t show any red in the action block. SO there isn’t any ambiguity

LR parse tables

Graphical user interface, application

Description automatically generated

FAIL 1

Chart

Description automatically generated

PASS – 1

Table

Description automatically generated