GRAMATICA TYPESTY

ANALIZADOR LEXICO

La gramática establecida maneja el formato de case insensitive dentro del apartado léxico del archivo jison.

\s+ = Se utiliza esta declaración léxica para el manejo de los espacios en blanco dentro del archivo analizado, ignora los espacios en blanco.

"//".* = Esta expresión Ignora los comentarios unilineal establecidos con dos barras.

[/][*][^*]*[*]+([^/*][^*]*[*])*[/] = Esta expresión ignora los comentarios multilínea.

Terminales Del Lenguaje

1) "int" = retorna el tipo de declaración de valor entero.
2) "char" = retorna el tipo de declaración de valor carácter.
3) "double" = retorna el tipo de declaración de valor decimal.
4) "string" = retorna el tipo de declaración de valor cadena.
5) "boolean" = retorna el tipo de declaración de valor booleano.
6) "true" = retorna el valor booleano, verdadero.
7) "false" = retorna el valor booleano, falso.
8) "void" = indica el tipo de retorno vacio en un método.
9) ";" = retorna el token punto y coma.
10) "," = retorna el token coma.
11) "{" = retorna el token llave de abrir.
12) "}" = retorna el token llave de cerrar
13) "(" = retorna el token paréntesis de abrir.
14) ")" = retorna el token paréntesis de cerrar.
15) "?" = retorna el token signo de interrogación.
16) ":" = retorna el token dos puntos.
17) "++" = retorna el token incremento.
18) "--" = retorna el token decremento.
19) "-" = retorna el token resta.
20) "+" = retorna el token suma.
21) "*" = retorna el token multiplicación.
22) "/" = retorna el token división.
23) "^" = retorna el token potencia.
24) "%" = retorna el token modulo.
25) "<=" = retorna el token menor igual.
26) ">=" = retorna el token mayor igual.
27) "==" = retorna el token igual igual.

28) “!=” = retorna el token diferente.
29) “<” = retorna el token menor.
30) “>” = retorna el token mayor
31) “=” = retorna el token igual
32) “||” = retorna el token or.
33) “&&” = retorna el token and.
34) “!” = retorna el token not.
35) “if” = retorna el token si.
36) “else” = retorna el token sino.
37) “switch” = retorna el token switch.
38) “case” = retorna el token case.
39) “default” = retorna el token default.
40) “while” = retorna el token mientras.
41) “do” = retorna el token do.
42) “for” = retorna el token for.
43) “break” = retorna el token break.
44) “continue” = retorna el token continuee.
45) “return” = retorna el token return.
46) “print” = retorna el token imprimir.
47) “tolower” = retorna el token tolower.
48) “toupper” = retorna el token toupper.
49) “length” = retorna el token length.
50) “truncate” = retorna el token truncate.
51) “round” = retorna el token round.
52) “typeof” = retorna el token typeof.
53) “tostring” = retorna el token tostring.
54) “tochararray” = retorna el token tochararray.
55) “exec” = retorna el token exec.
56) ([0-9])+([“.”])([0-9])+ = expresión utilizada para retornar los valores decimales.
57) [0-9]+ = expresión utilizada para retornar los valores enteros.
58) ([a-zA-Z])[a-zA-Z0-9_]* = expresión utilizada para retornar los valores identificadores.
59) [“] = identifica comilla doble para inicio de cadena, retorna string vacio en la cadena.

<str>[^”\\]+ = modulo str concatena cualquier sentencia dentro de la cadena analizada.

<str>”\\n” = modulo str concatena el salto de línea dentro de la cadena.

<str>”\\r” = modulo str concatena el \r dentro de la cadena.

<str>”\\t” = modulo str concatena el \t dentro de la cadena.

<str>"\\\\" = modulo str concatena el valor de \\ dentro de la cadena.

<str>"\\'" = modulo str concatena el valor de comilla simple dentro de la cadena

<str>[“] = modulo str identifica fin de string analizado y retorna valor cadena.

60) [\'](“\\n”|“\\r” |“\\t” |“\\'” |“\\\”” |“\\\\” |[^\'])[\'] = expresión utilizada para retornar valores caracteres.
61) <<EOF>> = retorna el final del archivo.
62) . = imprime en consola el error léxico detectado dentro del archivo analizado.
63) /lex = define el fin de las sentencias léxicas del archivo jison.

%{

Esta estructura permite importar módulos y declaración de variables y entornos a utilizar dentro del archivo jison, para el proyecto se cuenta con las siguientes importaciones:

1) Const TIPO_OPERACION = require(“../árbol/instrucciones”).TIPO_OPERACION
2) Const TIPO_VALOR = require(“../árbol/instrucciones”).TIPO_ VALOR
3) Const INSTRUCCIONES = require(“../árbol/instrucciones”). INSTRUCCIONES
4) Const TIPO_DATO = require(“../árbol/tablasimbolos”).TIPO_ DATO
5) Var cadena = “”;

}%

Precendencia de Operadores

1) %left ‘or’
2) %left ‘and’
3) %rigth ‘not’
4) %left ‘menor’ ‘menorigual’ ‘mayor’ ‘mayorigual’ ‘igualigual’ ‘noigual’
5) %left ‘incremento’ ‘decremento’ ‘mas’ ‘menos’
6) %left ‘por’ ‘dividido’ ‘modulo’
7) %left ‘potencia’
8) %left UMENOS

%start INICIO = esta instrucción le indica al archivo con que no terminal empezar la producción de análisis

GRAMATICA

No Terminales

1) INICIO = Se utiliza para indicar el inicio de las producciones de la gramática.
2) CUERPO = utilizada para manejar el entorno global de la aplicación typesty
3) CUERPO2 = utilizada para manejar el entorno local ya sea un método o función dentro de la aplicación typesty.
4) MAIN = utilizada para manejar la declaración del método o función principal de la aplicación.
5) METODO = utilizada para manejar la creación de métodos dentro del archivo analizado.
6) PARAMETROS = utilizada para manejar la declaración de parámetros para una función o método dentro del archivo analizado.
7) LLAMADA = utilizada para manejar la declaración de llamadas de métodos o funciones dentro del archivo analizado.
8) VALORESLLAMADA = utilizada para manejar la declaración de valores para los parámetros de un método o función.
9) DECLARACION = utilizada para manejar la declaración de variables.
10) ASIGNACION = utilizada para manejar la asignación de variables.
11) IMPRIMIR = utilizada para manejar la función imprimir.
12) SI = utilizada para manejar la estructura de la sentencia de control if.
13) SWITCHH = utilizada para manejar la estructura de la sentencia de control switch.
14) CASES = utilizada para manejar la estructura de los casos para la sentencia de control switch.
15) WHILEE = utilizada para manejar la estructura de la sentencia cíclica while.
16) DOWHILEE = utilizada para manejar la estructura de la sentencia cíclica do while.
17) FOR = utilizada para manejar la estructura de la sentencia cíclica do while.
18) BREAKK = utilizada para manejar las expresiones break.
19) CONTINUEE = utilizada para manejar las expresiones continue.
20) RETURNN = utilizada para manejar las expresiones return.
21) CASTEOR = utilizada para manejar la estructura de casteos.
22) TERNARIO = utilizada para manejar la estructura de la funcion ternaria.
23) TIPO = utilizada para manejar la declaración de tipo de dato de una variable o función.
24) EXP = utilizada para manejar las expresión dentro de cualquier estructura, ya sean valores enteros, decimales, cadenas, caracteres, booleanos e identificadores, operaciones aritméticas, relacionales y lógicas; signos de agrupación, incremento o decremento de variables, funciones to lower, to upper y nativas.

Producciones

1) INICIO: CUERPO EOF;
2) CUERPO

: CUERPO MAIN

| CUERPO METODO

| CUERPO DECLARACION

| CUERPO ASIGNACION

| MAIN

| METODO

| DECLARACION

| ASIGNACION;

3) CUERPO2

: CUERPO2 DECLARACION

| CUERPO2 ASIGNACION

| CUERPO2 LLAMADA

| CUERPO2 IMPRIMIR

| CUERPO2 SI

| CUERPO2 SWITCHH

| CUERPO2 WHILEE

| CUERPO2 DOWHILEE

| CUERPO2 FOR

| CUERPO2 BREAKK

| CUERPO2 CONTINUEE

| CUERPO2 RETURNN

| DECLARACION

| ASIGNACION

| LLAMADA

| IMPRIMIR

| SI

| SWITCHH

| WHILEE

| DOWHILEE

| FOR

| BREAKK

| CONTINUEE

| RETURNN;

4) MAIN

: exec identificador parentecisa parentesisc pcoma

| exec identificador parentecisa VALORESLLAMADA parentesisc pcoma;

5) METODO

: vacio identificador parentecisa parentesisc llavea CUERPO2 llavec

| vacio identificador parentecisa PARAMETROS parentesisc llavea CUERPO2 llavec;

6) PARAMETROS

: PARAMETROS coma TIPO identificador

| TIPO identificador;

7) LLAMADA

: identificador parentecisa VALORESLLAMADA parentesisc pcoma

| identificador parentecisa parentesisc pcoma;

8) VALORESLLAMADA

: VALORESLLAMADA coma EXP

| VALORESLLAMADA coma CASTEO

| VALORESLLAMADA coma TERNARIO

| EXP

| CASTEO

| TERNARIO;

9) DECLARACION

: TIPO identificador igual EXP pcoma

| TIPO identificador pcoma

| TIPO identificador igual CASTEO pcoma

| TIPO identificador igual TERNARIO pcoma;

10) ASIGNACION

: identificador igual EXP pcoma

| identificador igual CASTEO pcoma

| identificador igual TERNARIO pcoma

| identificador incremento pcoma

| identificador decremento pcoma;

11) IMPRIMIR

: imprimir parentecisa EXP parentesisc pcoma

| imprimir parentecisa CASTEO parentesisc pcoma

| imprimir parentecisa TERNARIO parentesisc pcoma

| imprimir parentecisa parentesisc pcoma;

12) SI

: si parentecisa EXP parentesisc llavea CUERPO2 llavec sino llavea CUERPO2 llavec

| si parentecisa EXP parentesisc llavea CUERPO2 llavec sino SI

| si parentecisa EXP parentesisc llavea CUERPO2 llavec

| si parentecisa CASTEO parentesisc llavea CUERPO2 llavec sino llavea CUERPO2 llavec

| si parentecisa CASTEO parentesisc llavea CUERPO2 llavec sino SI

| si parentecisa CASTEO parentesisc llavea CUERPO2 llavec

| si parentecisa TERNARIO parentesisc llavea CUERPO2 llavec sino llavea CUERPO2 llavec

| si parentecisa TERNARIO parentesisc llavea CUERPO2 llavec sino SI

| si parentecisa TERNARIO parentesisc llavea CUERPO2 llavec;

13) SWITCHH

: switch parentecisa EXP parentesisc llavea CASES llavec

| switch parentecisa CASTEO parentesisc llavea CASES llavec

| switch parentecisa TERNARIO parentesisc llavea CASES llavec;

14) CASES

: case EXP dospuntos CUERPO2 CASES

| case EXP dospuntos CUERPO2

| case CASTEO dospuntos CUERPO2 CASES

| case CASTEO dospuntos CUERPO2

| case TERNARIO dospuntos CUERPO2 CASES

| case TERNARIO dospuntos CUERPO2

| default dospuntos CUERPO2;

15) WHILEE

: mientras parentecisa EXP parentesisc llavea CUERPO2 llavec

| mientras parentecisa CASTEO parentesisc llavea CUERPO2 llavec

| mientras parentecisa TERNARIO parentesisc llavea CUERPO2 llavec;

16) DOWHILEE

: do llavea CUERPO2 llavec mientras parentecisa EXP parentesisc pcoma

| do llavea CUERPO2 llavec mientras parentecisa CASTEO parentesisc pcoma

| do llavea CUERPO2 llavec mientras parentecisa TERNARIO parentesisc pcoma;

17) FOR

: for parentecisa DECLARACION EXP pcoma EXP parentesisc llavea CUERPO2 llavec

| for parentecisa ASIGNACION EXP pcoma EXP parentesisc llavea CUERPO2 llavec;

18) BREAKK : break pcoma;
19) CONTINUEE : continuee pcoma;
20) RETURNN : returnn pcoma;
21) CASTEO : parentesisa TIPO parentesisc EXP;
22) TERNARIO

: EXP signointerrogacion EXP dospuntos EXP

| EXP signointerrogacion CASTEO dospuntos CASTEO

| EXP signointerrogacion CASTEO dospuntos EXP

| EXP signointerrogacion EXP dospuntos CASTEO;

23) TIPO

: entero

| decimal

| caracter

| cadena

| bandera;

24) EXP

: EXP mas EXP

| EXP menos EXP

| identificador incremento

| identificador decremento

| EXP por EXP

| EXP dividido EXP

| EXP potencia EXP

| EXP modulo EXP

| menos EXP %prec UMENOS

| EXP menor EXP

| EXP mayor EXP

| EXP menorigual EXP

| EXP mayorigual EXP

| EXP igualigual EXP

| EXP noigual EXP

| EXP or EXP

| EXP and EXP

| not EXP %prec UMENOS

| parentesisa EXP parentesisc

| tolower parentesisa EXP parentecisc

| toupper parentesisa EXP parentecisc

| length parentesisa EXP parentecisc

| truncate parentesisa EXP parentecisc

| round parentesisa EXP parentecisc

| typeof parentesisa EXP parentecisc

| tostring parentesisa EXP parentecisc

| tochararray parentesisa EXP parentecisc

| enteroo

| decimall

| caracterr

| cadenaa

| truee

| falsee

| identificador;