GRAMATICA

Entero	[0-9]+
Decimal	[0-9]+("."[0-9]+)
Booleano	True | False
Carácter	' símbolos '
Cadena	“ símbolos+ “

G = {N, T, P, S}

T = [, void ( ) [ ] entero decimal true false carácter cadena + * - / % == > < ++ --<= >= run id { } new toupper tolower tostring tochararray length typeof truncate round ? ; : case if else do while for switch until]

N = { INICIO, PRINCIPAL, CUERPO, DECLARACION, ASIGNACION, FUNCION, METODOS, EJECUTAR, IF, WHILE, DOWHILE, DOUNTIL, SWITCH, L_CASE, CASE, TIPO, EXPRESION, OPCION, OP_CUERPO, ACTUALIZACION, L_VALORES, L_ID, L_2D, CASTEO, VALORCAST, ELSEIF, CONELSEIF, PARAMETROS, L_PARAMETROS, TRANSFERENCIA}

S = {INICIO}

P ={

<INICIO> :: = <PRINCIPAL>

<PRINCIPAL> ::= <PRINCIPAL> <CUERPO>

|<CUERPO>

<CUERPO> :: = <DECLARACION>

|<ASIGNACION>

|<FUNCION>

|<METODOS>

|<EJECUTAR>

<DECLARACION>::= <TIPO> <L_ID> = <EXPRESION> ;

| <TIPO><L_ID> ;

| <TIPO> id [ ] = new <TIPO> [<EXPRESION>] ;

| <TIPO> id [ ] = [<L_VALORES>] ;

| <TIPO> id [ ] = toCharArray (<EXPRESION>);

| <TIPO> id [ ][ ] = [<EXPRESION>] ;

| <TIPO> id [ ][ ] = [<L_2D>] ; <L_ID>::

= <L_ID> , id

| id

<L_2D>::= <L_2D> , [<L_VALORES>]

| [<L_VALORES>]

<L_VALORES>::= <L_VALORES> , <EXPRESION>

| <EXPRESION>

<ASIGNACION>::= id = <EXPRESION> ;

| id ++;

| id --;

| id [<EXPRESION> = <EXPRESION> ;

| id [<EXPRESION>] [<EXPRESION>] = <EXPRESION> ;

<FUNCION>:= id (<L_PARAMETROS>) : <TIPO> { <OP_CUERPO> }

| id ( ) : <TIPO> { <OP_CUERPO> }

<METODO> id (<L_PARAMETROS>) : void { <OP_CUERPO> }
| id ( ) :void{ <OP_CUERPO> }
| id (<L_PARAMETROS>) { <OP_CUERPO> }
| id ( ) { <OP_CUERPO> }

<L_PARAMETROS>:: = <L_PARAMETROS> , <PARAMETROS>
| <PARAMATROS>

<PARAMETROS>:: = <TIPO> id
| <TIPO> id [ ]
| <TIPO> id [ ][ ] <TIPO>

:: = int
| char
| doublé
| boolean
| char
| string

<EJECUTAR> run id (L_VALORES>) ;
| run id ( ) ;

<OP_CUERPO>::= <OP_CUERPO><OPCION>
| <OPCION>

<OPCION>::= <ASIGNACION>

| <DECLARACION>

| <IF>

| <WHILE>

| <FOR>

| <SWITCH>

| <DOWHILE>

| <TRANSFERENCIA>

| <PRINT

| <LLAMADAS> ;

<IF>::= if (<EXPRESION>) {<OP_CUERPO> }

| if (<EXPRESION>) {<OP_CUERPO> } else {<OP_CUERPO> }

| if (<EXPRESION>) {<OP_CUERPO> } <ELSEIF>

| if (<EXPRESION>) {<OP_CUERPO> } <ELSEIF> else {<OP_CUERPO>}

<ELSEIF>:: = <ELSEIF> <CONELSEIF>

| <CONELSEIF>

<CONELSEIF>:: = else if (<EXPRESION>) { <OP_CUERPO>}

<SWITCH>::= switch (<EXPRESION>) {<L_CASE > default : <OP_CUERPO> }

| switch (<EXPRESION>) {<L_CASE >}

| switch (<EXPRESION>) { default : <OP_CUERPO> }

<L_CASE>::= <L_CASE> <CASE>

| <CASE>

<CASE>:: = case <EXPRESION> : <OP_CUERPO>

<WHILE>:: = while (<EXPRESION>) {<OP_CUERPO>}

<DOWHILE>::= do {<OP_CUERPO> } while (<EXPRESION>) ;

<DOUNTIL>::= do {<OP_CUERPO> } until (<EXPRESION>) ;

<FOR>::= for(<ASIGNACION> <EXPRESION> ; <ACTUALIZACION>){
<OP_CUERPO> }

| for (<DECLARACION> <EXPRESION> ; <ACTUALIZACION>) {
<OP_CUERPO>}

<ACTUALIZACION>::= id ++

| id –

| id = <EXPRESION>

<TRANSFERENCIA> break ;

| continue ;

| return <EXPRESION> ;

| return ;

<LLAMADA>::= id ( ) ;

| id (<L_VALORES>) ;

<PRINT>::= print (<EXPRESION>) ;

<PRINLN>:: = println (<EXPRESION>);

<EXPRESION>:: = <EXPRESION> + <EXPRESION>
| <EXPRESION> - <EXPRESION>
| <EXPRESION> * <EXPRESION>
| <EXPRESION> / <EXPRESION>
| <EXPRESION> % <EXPRESION>
| <EXPRESION> ^ <EXPRESION>
| <EXPRESION> || <EXPRESION>
| <EXPRESION> && <EXPRESION>
| <EXPRESION> == <EXPRESION>
| <EXPRESION> ¡= <EXPRESION>
| <EXPRESION> <= <EXPRESION>
| <EXPRESION> < <EXPRESION>
| <EXPRESION> >= <EXPRESION>
| <EXPRESION> > <EXPRESION>
| - <EXPRESION>
| ¡ <EXPRESION>
| (<EXPRESION>)
| id
| id++
| id--
| id [<EXPRESION>]
| id [<EXPRESION>] [<EXPRESION>]
| cadena

| decimal

| entero

| carácter

| true

| false

| tolower(<EXPRESION>)

| toupper(<EXPRESION>)

| length(<EXPRESION>)

| typeof(<EXPRESION>)

| round(<EXPRESION>)

| truncate(<EXPRESION>)

| tostring(<EXPRESION>)

| <CASTEO>

| <LLAMADAS>

<CASTEO>::= (<TIPO>)<VALORCAST>

<VALORCAST>:: = cadena

| decimal

| id

| entero

| false

| true

| caracter

<TIPO>::= int

| string

| boolean

| double

| char

Repositorio:

https://github.com/Pdante1897/OLC1-P2-201700945.git