GRAMATICA

```
%{

%}

%lex

%options case-insensitive

LETRAS [a-zA-ZñÑ]+
ENTERO [0-9]+
DECIMAL {ENTERO}("."{ENTERO})
ID {LETRAS}({LETRAS}|{ENTERO}|"_")*

ESCAPECHAR [\'\"\\nrt]
CESPECIALES \\{ESCAPECHAR}
ACEPTACION [^\"\n]
CADENA \"({CESPECIALES}|{ACEPTACION})*\"

ACEPTACIONC [^\'\n]
CARACTER \'({ACEPTACIONC}|{CESPECIALES})\'

%%


[ \r\t\n\s]+  {}
"//".*      {}
[/][*]([\n]|[^"*/"]|\*|"/")*[*][/] {}

"++"                { return 'inc' }
"--"                { return 'dec' }
"=="                { return 'igualigual' }
"!="                { return 'diferente' }

"("                 { return 'parentesisa' }
")"                 { return 'parentesisc' }
"["                 { return 'corchetea' }
"]"                 { return 'corchetec' }
"{"                 { return 'llavea' }
"}"                 { return 'llavec' }
","                 { return 'coma' }
";"                 { return 'puntoycoma' }
"="                 { return 'igual' }
"?"                 { return 'interrogacion' }
":"                 { return 'dospuntos' }
"+"                 { return 'mas' }
"-"                 { return 'menos' }
"**"                { return 'pow' }
"*"                 { return 'multi' }
```

```
"/"                 { return 'div' }
"%"                 { return 'mod' }
"<="                { return 'menorigual' }
"<"                 { return 'menorque' }
">="                { return 'mayorigual' }
">"                 { return 'mayorque' }


"||"                { return 'or' }
"&&"                { return 'and' }
"!"                 { return 'not' }
"^"                 { return 'xor' }


"int"               { return 'int'}
"double"            { return 'double'}
"boolean"           { return 'boolean'}
"char"              { return 'char'}
"string"            { return 'string'}
//para asignaciones
"const"             { return 'const'}

//resultado booleanos
"true"              { return 'true'}
"false"             { return 'false'}

//palabras para ciclos
"if"                { return 'if'}
"else"              { return 'else'}
"switch"            { return 'switch'}
"case"              { return 'case'}
"default"           { return 'default'}
"while"             { return 'while'}
"do"                { return 'do'}
"for"               { return 'for'}

//palabras para funciones
"break"             { return 'break'}
"continue"          { return 'continue'}
"return"            { return 'return'}
"println"           { return 'println'}
"print"             { return 'print'}
"typeof"            { return 'typeof'}
"void"              { return 'void'}
"length"            { return 'length'}
"call"              { return 'call'}

{ID}                { return 'id'}
```

```
{CADENA}        { return 'cadena'}
{DECIMAL}       { return 'decimal'}
{ENTERO}        { return 'entero'}
{CARACTER}      { return 'caracter'}



//EOF INDICA EL FIN DEL DOCUMENTO LEIDO

<<EOF>> return 'EOF';

.  {//CAPTURA  Y RECUPERACION DE ERRORES LEXICOS }


/lex

//Fin del analisis lexico

//INICIO DEL ANALISIS SINTACTICO
//PRECEDENCIA
%left 'coma','puntoycoma', 'igual'
%left 'or'
%left 'and'
%right 'xor'
%left 'igualigual','diferente','menorque','menorigual','mayorque','mayorigual'
%left 'mas','menos'
%left 'div', 'multi','mod'
%right 'pow'
%left 'inc','dec'
%right 'not'
%right UMINUS


//INICIO DE LA GRAMATICA

%start INICIO
%%
INICIO: INSTRUCCIONES  EOF
        ;
INSTRUCCIONES : INSTRUCCIONES INSTRUCCION
        | INSTRUCCION
        ;
INSTRUCCION: ASIGNACION puntoycoma
        | DECLARACION puntoycoma
        | IF
        | IF_SINLLAVES
        | SWITCH
        | FOR
        | WHILE
        | DO_WHILE
```

```
        | break puntoycoma
        | continue puntoycoma
        | return puntoycoma
        | return EXPRESION puntoycoma
        | N_PRINT puntoycoma
        | N_PRINTLN puntoycoma
        | FUNCIONES
        | METODOS
        | LLAMADA puntoycoma
        | BLOQUE_INST
        | id inc puntoycoma
        | id dec puntoycoma
        | inc id puntoycoma
        | dec id puntoycoma
        | error puntoycoma
        ;
DECLARACION: TIPOVAR CONJID igual EXPRESION
        | const TIPOVAR CONJID igual EXPRESION
        | TIPOVAR CONJID
        ;

ASIGNACION: CONJID igual EXPRESION
        ;

CONJID: CONJID coma id
        | id
        ;

TIPOVAR: int
        | double
        | boolean
        | char
        | string
  ;
TIPODATO: cadena
        | caracter
        | decimal
        | id
        | entero
        | true
        | false
        ;


IF: if parentesisa EXPRESION parentesisc BLOQUE_INST
        | if parentesisa EXPRESION parentesisc BLOQUE_INST else BLOQUE_INST
        | if parentesisa EXPRESION parentesisc BLOQUE_INST else IF
        ;
```

```
IF_SINLLAVES: if parentesisa EXPRESION parentesisc IFS_INSTRUCCION
        | if parentesisa EXPRESION parentesisc IFS_INSTRUCCION  else IFS_INSTRUCCION
        | if parentesisa EXPRESION parentesisc IFS_INSTRUCCION  else IF_SINLLAVES
        ;

IFS_INSTRUCCION: ASIGNACION puntoycoma
        | DECLARACION puntoycoma
        | N_PRINT puntoycoma
        | N_PRINTLN puntoycoma
        | id inc puntoycoma
        | id dec puntoycoma
        | inc id puntoycoma
        | dec id puntoycoma
        ;


//SWITCH
SWITCH: switch parentesisa EXPRESION parentesisc llavea CASES_LIST llavec
        | switch parentesisa EXPRESION parentesisc llavea CASES_LIST DEFAULT llavec
        | switch parentesisa EXPRESION parentesisc llavea DEFAULT llavec
        | switch parentesisa EXPRESION parentesisc llavea DEFAULT CASES_LIST  llavec
        | switch parentesisa EXPRESION parentesisc llavea CASES_LIST DEFAULT CASES_LIST  llavec
        ;

CASES_LIST: CASES_LIST CASE
        | CASE
        ;
CASE: case EXPRESION dospuntos INSTRUCCIONES
        | case EXPRESION dospuntos
        ;
DEFAULT: default dospuntos INSTRUCCIONES
        | default dospuntos
        ;



//FOR
FOR: for parentesisa DECLARACION puntoycoma EXPRESION puntoycoma EXPRESION parentesisc
BLOQUE_INST
   | for parentesisa ASIGNACION puntoycoma EXPRESION puntoycoma EXPRESION parentesisc
BLOQUE_INST
   ;


//WHILE
WHILE: while parentesisa EXPRESION parentesisc BLOQUE_INST
   ;
```

```
//DO WHILE
DO_WHILE: do BLOQUE_INST while parentesisa EXPRESION parentesisc puntoycoma
        ;


//METODOS
METODOS: void id parentesisa PARAMETROS parentesisc  BLOQUE_INST
        | void id parentesisa parentesisc  BLOQUE_INST
        ;

//FUNCIONES
FUNCIONES: TIPOVAR id parentesisa PARAMETROS parentesisc BLOQUE_INST
        | TIPOVAR id parentesisa parentesisc BLOQUE_INST
        ;

//Parametros
PARAMETROS:PARAMETROS coma TIPOVAR id
        | TIPOVAR id
        ;

//LLAMADA
LLAMADA: call  id parentesisa PARAMETROSLLAMADA parentesisc
        | call  id parentesisa parentesisc
        | id parentesisa PARAMETROSLLAMADA parentesisc
        | id parentesisa parentesisc
        ;

PARAMETROSLLAMADA: PARAMETROSLLAMADA coma EXPRESION
        | EXPRESION
        ;

N_PRINTLN: println parentesisa EXPRESION parentesisc
        | println parentesisa parentesisc
        ;
N_PRINT: print parentesisa EXPRESION parentesisc
        | print parentesisa parentesisc
        ;
N_TYPEOF: typeof parentesisa EXPRESION parentesisc
        ;

//Bloque de Instrucciones
BLOQUE_INST: llavea INSTRUCCIONES llavec
        | llavea llavec
        ;
```

```
EXPRESION:
        menos EXPRESION %prec UMINUS
        | EXPRESION mas EXPRESION
        | EXPRESION menos EXPRESION
        | EXPRESION div EXPRESION
        | EXPRESION multi EXPRESION
        | EXPRESION mod EXPRESION
        | EXPRESION pow EXPRESION

        | EXPRESION igualigual EXPRESION
        | EXPRESION diferente EXPRESION
        | EXPRESION menorque EXPRESION
        | EXPRESION menorigual EXPRESION
        | EXPRESION mayorque EXPRESION
        | EXPRESION mayorigual EXPRESION

        | EXPRESION or EXPRESION
        | EXPRESION and EXPRESION
        | EXPRESION xor EXPRESION
        | not EXPRESION

        | id inc
        | inc id
        | id dec
        | dec id

        | parentesisa EXPRESION parentesisc
        | TIPODATO
        | N_TYPEOF

        | id igual EXPRESION
        | LLAMADA
        ;
```