```
grammar Grammar;
import Lexicon;

@parser::header {
      import ast.*;
}

start returns[Program ast]
      : definiciones EOF { $ast = new Program($definiciones.lista); }
      ;

definiciones returns[List<Definicion> lista = new
ArrayList<Definicion>()]
      : (definicion { $lista.add($definicion.ast);})*;

definicion returns[Definicion ast]
      : defVariable { $defVariable.ast.setAmbito("global"); $ast =
$defVariable.ast; }
      | 'struct' IDENT '{' campos '}' ';' { $ast = new DefStruct($IDENT,
$campos.lista); }
      | funcion { $ast = $funcion.ast; }
      ;

variables returns[List<DefVariable> lista = new ArrayList<DefVariable>()]
      : (defVariable { $defVariable.ast.setAmbito("local");
$lista.add($defVariable.ast); })*
      ;

defVariable returns[DefVariable ast]
      : 'var' IDENT ':' tipo ';' { $ast = new DefVariable($tipo.ast,
$IDENT, ""); }
      ;

campos returns[List<VariableStruct> lista = new
ArrayList<VariableStruct>()]
      : (campo {$lista.add($campo.ast);})*
      ;

campo returns[VariableStruct ast]
      : IDENT ':' tipo ';' { $ast = new VariableStruct($IDENT,
$tipo.ast); }
      ;

funcion returns[DefFuncion ast]
      : IDENT '(' parametros ')' '{' variables sentencias '}' { $ast =
new DefFuncion($IDENT, $parametros.lista, null, $variables.lista,
$sentencias.lista); }
      | IDENT '(' parametros ')' ':' tipo '{' variables sentencias '}' {
$ast = new DefFuncion($IDENT, $parametros.lista, $tipo.ast,
$variables.lista, $sentencias.lista); }
      ;

sentencias returns[List<Sentencia> lista = new ArrayList<Sentencia>()]
      : (sentencia { $lista.add($sentencia.ast); })*
```

```
      ;

sentencia returns[Sentencia ast]
      : expresion '=' expresion ';' { $ast = new
Asignacion($ctx.expresion(0), $ctx.expresion(1)); }
      | 'printsp' expresion ';' { $ast = new Print($expresion.ast, "sp");
}
      | 'print' expresion ';' { $ast = new Print($expresion.ast, ""); }
      | 'println' expresion ';' { $ast = new Print($expresion.ast, "ln");
}
      | 'read' expresion ';' { $ast = new Read($expresion.ast); }
    | IDENT '(' expresiones ')' ';' { $ast = new FuncionLlamada($IDENT,
$expresiones.lista); }
      | 'if' '(' expresion ')' '{' sentencias '}' { $ast = new
If($expresion.ast, $sentencias.lista, null); }
      | 'if' '(' expresion ')' '{' sentencias '}' 'else' '{' sentencias
'}' { $ast = new If($expresion.ast, $ctx.sentencias(0).lista,
$ctx.sentencias(1).lista); }
      | 'while' '(' expresion ')' '{' sentencias '}' { $ast = new
While($expresion.ast, $sentencias.lista); }
      | 'return' expresion ';' { $ast = new Return($expresion.ast); }
      | 'return' ';' { $ast = new Return(null); }
      ;

expresiones returns[List<Expresion> lista = new ArrayList<Expresion>()]
      : (expresion { $lista.add($expresion.ast); } (',' expresion {
$lista.add($expresion.ast); })*)*
      ;

parametros returns[List<Parametros> lista = new ArrayList<Parametros>()]
      : (parametro { $lista.add($parametro.ast); } (',' parametro {
$lista.add($parametro.ast); })*)*
      ;

parametro returns[Parametros ast]
      : IDENT ':' tipo { $ast = new Parametros($IDENT, $tipo.ast); }
      ;

expresion returns[Expresion ast]
      : IDENT { $ast = new Ident($IDENT); }
      | LITENT { $ast = new LitEnt($LITENT); }
      | LITREAL  { $ast = new LitReal($LITREAL); }
      | LITCHAR  { $ast = new LitChar($LITCHAR); }
      | IDENT '(' expresiones ')' { $ast = new ExpresionLlamada($IDENT,
$expresiones.lista); }
      | '(' expresion ')' { $ast = $expresion.ast; }
      | '<' tipo '>' '(' expresion ')' { $ast = new Cast($tipo.ast,
$expresion.ast); }
      | expresion '[' expresion ']' { $ast = new Array($ctx.expresion(0),
$ctx.expresion(1)); }
      | expresion '.' IDENT { $ast = new Struct($ctx.expresion(0),
$IDENT); }
      | expresion op=('*'|'/'|'%') expresion { $ast = new
ExpresionAritmetica($ctx.expresion(0), $op, $ctx.expresion(1)); }
```

```
      | expresion op=('+'|'-') expresion { $ast = new
ExpresionAritmetica($ctx.expresion(0), $op, $ctx.expresion(1)); }
      | expresion op=('<'|'>'|'<='|'>=') expresion { $ast = new
ExpresionLogica($ctx.expresion(0), $op, $ctx.expresion(1)); }
      | expresion op=('!='|'==') expresion { $ast = new
ExpresionLogica($ctx.expresion(0), $op, $ctx.expresion(1)); }
      | expresion '&&' expresion { $ast = new
ExpresionLogicaAndOr($ctx.expresion(0), "&&", $ctx.expresion(1));}
      | expresion '||' expresion { $ast = new
ExpresionLogicaAndOr($ctx.expresion(0), "||", $ctx.expresion(1));}
      | '!' expresion { $ast = new ExpresionDistinto($expresion.ast); }
      ;


tipo returns[Tipo ast]
      : 'int' { $ast = new IntTipo(); }
      | 'float' { $ast = new RealTipo(); }
      | 'char' { $ast = new CharTipo(); }
      | '[' LITENT ']' tipo { $ast = new ArrayTipo($LITENT, $tipo.ast); }
      | IDENT { $ast = new StructTipo($IDENT); }
      ;
```