```
grammar Grammar
      ;
import Lexicon
      ;

@parser::header {
    import ast.*;
    import ast.definiciones.*;
    import ast.expresiones.*;
    import ast.sentencias.*;
    import ast.tipos.*;

}

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

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


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

      |           'struct' IDENT '{' campos '}' ';'                                                {$ast =
new DefStruct($IDENT,$campos.list,new TipoStruct($IDENT));}
      |           IDENT '(' parametrosOpc ')' '{' varLocales listaSentencias '}'              {$ast = new
DefFuncion($IDENT,new
TipoFuncion(TipoVoid.getInstance(),$parametrosOpc.list),$varLocales.list,$listaSentencias.list);}
      |           IDENT '(' parametrosOpc ')' ':' tipo '{' varLocales listaSentencias '}' {$ast = new
DefFuncion($IDENT,new TipoFuncion($tipo.ast,$parametrosOpc.list),$varLocales.list,$listaSentencias.list);}
      ;
```

```
parametrosOpc returns[List<DefVariable> list = new ArrayList<DefVariable>()]
    :      parametros  {$list = $parametros.list;}
    |
    ;

parametros returns[List<DefVariable> list = new ArrayList<DefVariable>()]
    :       p1=parametro {$list.add($p1.ast);} (',' p2=parametro {$list.add($p2.ast);})*
    ;

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


varLocales returns[List<DefVariable> list = new ArrayList<DefVariable>()]
      : (varLocal {$list.add($varLocal.ast);})*
      ;

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

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

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

tipoSimple returns[Tipo ast]
      :     'int'       {$ast = TipoInt.getInstance();}
      |     'char'            {$ast = TipoChar.getInstance();}
```

```
      |     'float'          {$ast = TipoFloat.getInstance();}
      ;


tipo returns[Tipo ast]
  :       tipoSimple                          {$ast = $tipoSimple.ast;}
  |         '[' INT_CONSTANT ']' tipo    {$ast = new TipoArray($INT_CONSTANT,$tipo.ast);}
  |         IDENT                             {$ast = new TipoStruct($IDENT);}
  ;



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

sentencia  returns[Sentencia ast]
   :             'print' expr ';'                              {$ast = new Print($expr.ast);}
   |      'printsp' expr ';'                           {$ast = new PrintSp($expr.ast);}
   |        'println' expr ';'                              {$ast = new PrintLn($expr.ast);}
   |        expr '=' expr ';'                           {$ast = new
Asignacion($ctx.expr(0),$ctx.expr(1));}
   |        'if' '(' expr ')'                           '{' listaSentencias '}' 'else' '{'
listaSentencias '}'   {$ast = new
IfElse($expr.ast,$ctx.listaSentencias(0).list,$ctx.listaSentencias(1).list);}
   |        'if' '(' expr ')' '{' listaSentencias '}'            {$ast = new
If($expr.ast,$listaSentencias.list);}
   |        'while' '(' expr ')' '{' listaSentencias '}'   {$ast = new
While($expr.ast,$listaSentencias.list);}
   |        IDENT '(' argumentosOpc ')' ';'                {$ast = new
InvocacionProcedimiento($IDENT,$argumentosOpc.list);}
   |        'return' ';'                                   {$ast = new Return();}
```

```
   |        'return' expr ';'                                  {$ast = new Return($expr.ast);}

   |        'read' expr ';'                                    {$ast = new Read($expr.ast);}

   ;

expr returns[Expresion ast]
  :     IDENT                                                          {$ast = new Variable($IDENT);}
  |         INT_CONSTANT                                              {$ast = new
CTE_Entera($INT_CONSTANT);}
  |         REAL_CONSTANT                                             {$ast = new
CTE_Real($REAL_CONSTANT);}
  |   CHAR_CONSTANT                                                {$ast = new CTE_Char($CHAR_CONSTANT);}
  |     expr '[' expr ']'                                          {$ast = new
AccesoArray($ctx.expr(0),$ctx.expr(1));}
  |         '(' expr ')'                                                {$ast = $expr.ast;}
  |   expr '.' IDENT                                               {$ast = new
AccesoStruct($ctx.expr(0),$IDENT);}
  |         'cast' '<' tipo '>' '(' expr ')'                  {$ast = new ExpresionCast($tipo.ast,
$expr.ast);}
  |   '-' expr                                                          {$ast = new ExprMenosUnario($ctx.expr);}
  |         '!' expr                                                    {$ast = new ExprNot($ctx.expr);}
  |   expr op=('*'|'/') expr                                   {$ast = new
ExprAritmetica($ctx.expr(0),$op.text,$ctx.expr(1));}
  |   expr op=('+'|'-') expr                                   {$ast = new
ExprAritmetica($ctx.expr(0),$op.text,$ctx.expr(1));}
  |         expr op=('=='|'>='|'<='|'!='|'<'|'>') expr     {$ast = new
ExprComparacion($ctx.expr(0),$op.text,$ctx.expr(1));}
  |   expr op=('&&'|'||') expr                                 {$ast = new
ExprLogica($ctx.expr(0),$op.text,$ctx.expr(1));}
  |   IDENT '(' argumentosOpc ')'                              {$ast = new ExpresionInvocacion($IDENT,
$argumentosOpc.list);}
  ;
```

```
argumentosOpc returns[List<Expresion> list = new ArrayList<Expresion>()]
   :               argumentos {$list = $argumentos.list;}
   |
   ;

argumentos returns[List<Expresion> list = new ArrayList<Expresion>()]
   :        p1 = expr {$list.add($p1.ast);} (',' p2=expr {$list.add($p2.ast);})*
   ;
```