```
grammar Grammar
	;

import Lexicon
	;

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

start returns[Program ast]

	: {List<AST> prog = new ArrayList<AST>();}
	(definicion{prog.add($definicion.ast);}|funcion{prog.add($funcion.ast);}|estructura{prog.add($estructura.ast);})* EOF { $ast = new Program(prog); }
	;

estructura returns[DefStruct ast]
	: 'struct' IDENT '{' (pa+=parametro ';')*'}' ';' { $ast = new DefStruct($IDENT,$pa);}
	;

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

sentencia returns[Sentence ast]
	: te=('print'|'printsp'|'println') expr ';' {$ast = new Print($te.text,$expr.ast);}
	| 'read' expr ';'  {$ast = new Read($expr.ast);}
	| left=expr '=' right=expr ';' {$ast = new Assignment($left.ast, $right.ast);}
	| 'if' '(' expr ')' '{' tru+=sentencia* '}' 'else' '{' fals+=sentencia* '}' {$ast = new IfElseSentence($expr.ast,$tru,$fals);}
	| 'if' '(' expr ')' '{' tru+=sentencia* '}' {$ast = new IfSentence($expr.ast,$tru);}
```

```
        | 'while' '(' expr ')' '{' se+=sentencia* '}' {$ast = new WhileSentence($expr.ast,$se);}
        | 'return' expr ';'  {$ast = new ReturnNode($expr.ast);}
        | 'return' ';'       {$ast = new ReturnNode(null);}
        | IDENT '('')' ';' {$ast = new FuncCall($IDENT,new ArrayList<Expr>());}
        | IDENT'('ex+=expr ( ','ex+=expr )* ')' ';' {$ast = new FuncCall($IDENT,$ex);}
        ;

expr returns[Expr ast]
        : LITENT {$ast = new LitEnt($LITENT);}
        | LITREAL {$ast = new LitReal($LITREAL);}
        | LITCHAR {$ast = new LitChar($LITCHAR);}
        | IDENT {$ast = new Variable($IDENT);}
        | IDENT '('')' {$ast =new MethodCallExpr($IDENT,new ArrayList<Expr>());}
        | IDENT'('ex+=expr ( ','ex+=expr )* ')' {$ast =new MethodCallExpr($IDENT,$ex);}
        | left=expr '[' right=expr ']'{$ast = new ArrayAcces($left.ast, $right.ast);}
        | left=expr '.' IDENT {$ast = new Acces($left.ast, $IDENT);}
        | '(' expr ')' {$ast = $expr.ast ;}
        |  '<' tipo '>' '(' expr ')' {$ast = new Cast($tipo.ast, $expr.ast);}
        | left=expr op=('*' | '/') right=expr {$ast = new ExprAritmetica($left.ast, $op.text,
$right.ast);}
        | left=expr op=('+' | '-') right=expr {$ast = new ExprAritmetica($left.ast, $op.text,
$right.ast);}
        | left=expr op=('>' | '<' | '>=' | '<=') right=expr {$ast = new ExprLogica($left.ast,
$op.text, $right.ast);}
        | left=expr op=('==' | '!=') right=expr {$ast = new ExprLogica($left.ast, $op.text,
$right.ast);}
        | left=expr op='&&' right=expr {$ast = new ExprLogica($left.ast, $op.text, $right.ast);}
        | left=expr op='||' right=expr{$ast = new ExprLogica($left.ast, $op.text, $right.ast);}
        | '!' expr {$ast = new ExprLogicaNe( $expr.ast);}
        ;

funcion returns [Func ast]
```

```
	:

	IDENT '('')' '{' '}' {$ast = new Func($IDENT,null,null,null,null);}

	|{List<Parameter> para = new ArrayList<Parameter>();List<Sentence> sent = new
ArrayList<Sentence>(); List<DefVar> def = new ArrayList<DefVar>();}

	IDENT '('p=parametro{para.add($p.ast);} (','pa=parametro {para.add($pa.ast);})*')' ':'
tipo '{'(definicion{def.add($definicion.ast);})* (sentencia{sent.add($sentencia.ast);})+ '}'

	{$ast = new Func($IDENT,para,$tipo.ast,def,sent);}

	|{List<Parameter> para = new ArrayList<Parameter>();List<Sentence> sent = new
ArrayList<Sentence>(); List<DefVar> def = new ArrayList<DefVar>();}

	IDENT '('p=parametro{para.add($p.ast);} (','pa=parametro {para.add($pa.ast);})*')'
'{'(definicion{def.add($definicion.ast);})* (sentencia{sent.add($sentencia.ast);})+ '}'

	{$ast = new Func($IDENT,para,null,def,sent);}

	|{List<Sentence> sent = new ArrayList<Sentence>(); List<DefVar> def = new
ArrayList<DefVar>();}

	IDENT '('')' ':' tipo '{'(definicion{def.add($definicion.ast);})*
(sentencia{sent.add($sentencia.ast);})+ '}'

	{$ast = new Func($IDENT,null,$tipo.ast,def,sent);}

	|{List<Sentence> sent = new ArrayList<Sentence>(); List<DefVar> def = new
ArrayList<DefVar>();}

	IDENT '('')' '{'(definicion{def.add($definicion.ast);})*
(sentencia{sent.add($sentencia.ast);})+ '}'

	{$ast = new Func($IDENT,null,null,def,sent);}

	|IDENT '('')' ':' '{' '}' {$ast = new Func($IDENT,null,null,null,null);}

	;


parametro returns[Parameter ast]

	:	IDENT ':' tipo {$ast = new Parameter($IDENT,$tipo.ast);}

	;


tipo returns[Type ast]

	: 'int' {$ast = new IntType();}

	| 'float' {$ast = new RealType();}

	| 'char' {$ast = new CharType();}

	| '['LITENT']' tipo {$ast = new ArrayType($LITENT, $tipo.ast);}
```

```
|       IDENT {$ast = new StructType($IDENT.text);}
;
```