# Attribute Grammar

|  |  |  |
| --- | --- | --- |
| Nodo | Predicados | Reglas Semánticas |
| program → *bloque*:bloque\* |  |  |
|  |  |  |
| definicion\_campo\_struct → *nombre*:String *tipo*:tipo |  |  |
| definicion\_variable\_local → *nombre*:String *tipo*:tipo |  |  |
| parametro → *nombre*:String *tipo*:tipo |  |  |
|  |  |  |
| definicion\_variable\_global:bloque → *nombre*:String *tipo*:tipo |  |  |
| struct:bloque → *nombre*:String *definicion\_campo\_struct*:definicion\_campo\_struct\* |  |  |
| funcion:bloque → *nombre*:String *parametros*:parametro\* *retorno*:tipo *locales*:definicion\_variable\_local\* *sentencias*:sentencia\* | tipoSimple(retorno.tipo)  tipoSimple(parametroi) | sentenciasi.funcionActual = funcion |
|  |  |  |
| sentencia\_asignacion:sentencia → *izquierda*:expr *derecha*:expr | mismoTipo(izquierda.tipo, derecha.tipo)  tipoSimple(izquierda.tipo)  izquierda.modificable == true |  |
| sentencia\_print:sentencia → *expresiones*:expr | tipoSimple(expresiones.tipo) |  |
| sentencia\_read:sentencia → *expresiones*:expr | tipoSimple(expresiones.tipo) expresiones.modificable==true |  |
| sentencia\_if:sentencia → *condicion*:expr *sentencias*:sentencia\* *sino*:sentencia\* | condicion.tipo==tipoInt | sentenciasi.funcionActual = sentencia\_if.funcionActual |
| sentencia\_while:sentencia → *condicion*:expr *sentencias*:sentencia\* | condicion.tipo==tipoInt | sentenciasi.funcionActual = sentencia\_while.funcionActual |
| sentencia\_llamada\_funcion:sentencia → *nombre*:String *parametros*:expr\* | | sentencia\_llamada\_funcion.parametrosi| == | sentencia\_llamada\_funcion.funcionActual.parametrosi|  sentencia\_llamada\_funcion.parametrosi.tipo == sentencia\_llamada\_funcion.funcionActual.parametrosi.tipo |  |
| sentencia\_return:sentencia → *expresion*:expr | si expresion == null  sentencia\_return.funcionActual.tipo == tipoVoid  sino  sentencia\_return.funcionActual.tipo == expresion.tipo |  |
|  |  |  |
| tipoInt:tipo → λ |  |  |
| tipoFloat:tipo → λ |  |  |
| tipoChar:tipo → λ |  |  |
| tipoVar:tipo → *string*:String |  |  |
| tipoArray:tipo → *tamanio*:String *tipo*:tipo |  |  |
| tipoStruct:tipo → λ |  |  |
| - |  |  |
| expr\_int:expr → *string*:String |  | expr\_int.tipo = tipoInt  expr\_int.modificable = false |
| expr\_real:expr → *string*:String |  | expr\_real.tipo = tipoFloat  expr\_real.modificable=false |
| expr\_char:expr → *string*:String |  | expr\_char.tipo=tipoChar  expr\_chat.modificable=false |
| expr\_ident:expr → *string*:String |  | expr\_ident.tipo=expr\_ident.definicion.tipo expr\_ident.modificable=true |
| expr\_binaria:expr → *izquierda*:expr *operador*:operador *derecha*:expr | si(operador es aritmético)  tipoSimple(izquierda.tipo)  si(operador es lógico)  tipoSimple(izquierda.tipo)  si(operador es booleano)  izquierda.tipo==tipoInt  mismoTipo(izquierda, derecha) | expr\_binaria.tipo=izquierda.tipo  expr\_binaria.modificable=false |
| expr\_vector:expr → *fuera*:expr *dentro*:expr | fuera.tipo==tipoArray  dentro.tipo==tipoInt | expr\_vector.tipo = tipoArray  expr\_vector.modificable=false |
| expr\_punto:expr → *izquierda*:expr *derecha*:expr |  | expr\_punto = tipoStruct  expr\_punto.modificable=false |
| expr\_parentesis:expr → *expr*:expr |  | expr\_parentesis.tipo = expr.tipo  expr\_parentesis.modificable=expr.modificable |
| expr\_cast:expr → *tipo\_convertido*:tipo *expr*:expr |  | expr\_cast.tipo = tipo.tipo  expr\_cast.modificable=false |
| expr\_llamada\_funcion:expr → *nombre*:String *parametros*:expr\* | |expr\_llamada\_funcion.parametrosi| == |expr\_llamada\_funcion.funcionActual.parametrosi|  expr\_llamada\_funcion.parametrosi.tipo == expr\_llamada\_funcion.funcionActual.parametrosi.tipo  expr\_llamada\_funcion.funcionActual.retorno ≠ tipoVoid | expr\_llamada\_funcion.tipo = expr.tipo  expr\_llamada\_funcion.modificable=false |
|  |  |  |
| operador\_aritmetico:operador → *string*:String |  |  |
| operador\_logico:operador → *string*:String |  |  |
| operador\_booleano:operador → *string*:String |  |  |
|  |  |  |

Recordatorio de los operadores (para cortar y pegar): ⇒ ⇔ ≠ ∅ ∈ ∉ ∪ ∩ ⊂ ⊄ ∑ ∃ ∀

# Atributos

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Categoría Sintáctica** | **Nombre del atributo** | **Tipo Java** | **Heredado/Sintetizado** | **Descripción** |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |