## Attribute Grammar

|  |  |  |
| --- | --- | --- |
| **Nodo** | **Predicados** | **Reglas Semánticas** |
| definicion\_funcion:definicion → nombre:String parametros:definicion\_variable\* retorno:tipo locales:definicion\_variable\* sentencias:sentencia\* | tipoSimple(retorno.tipo)  tipoSimple(parametrosi) | sentenciasi.funcionActual = definicion\_funcion |
|  |  |  |
| sentencia\_asignacion:sentencia → izquierda:expr derecha:expr | mismoTipo(izquierda.tipo, derecha.tipo)  tipoSimple(izquierda.tipo)  izquierda.modificable == true |  |
| sentencia\_print:sentencia → expresiones:expr fincadena:String | 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.definicion.parametrosi|  sentencia\_llamada\_funcion.parametrosi.tipo == sentencia\_llamada\_funcion.definicion.parametrosi.tipo |  |
| sentencia\_return:sentencia → expresion:expr | si expresion == null  sentencia\_return.funcionActual.tipo == tipoVoid  sino  sentencia\_return.funcionActual.tipo == expresion.tipo |  |

|  |  |  |
| --- | --- | --- |
| 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\_uso\_variable:expr → string:String |  | expr\_uso\_variable.tipo=expr\_ident.definicion.tipo expr\_uso\_variable.modificable=true |
| expr\_operacion:expr → izquierda:expr operador:operador derecha:expr | si(operador es aritmético)  tipoSimple(izquierda.tipo)  si(operador es booleano)  tipoSimple(izquierda.tipo)  si(operador es lógico)  izquierda.tipo==tipoInt  mismoTipo(izquierda, derecha) | expr\_operacion.tipo=izquierda.tipo  expr\_operacion.modificable=false |
| expr\_negada:expr → operador:operador derecha:expr | derecha.tipo==tipoInt | expr\_negada.tipo=izquierda.tipo  expr\_negada.modificable=false |
| expr\_cast:expr → tipo\_convertido:tipo expr:expr | tipoSimple(expr\_cast.tipo\_convertido)  tipoSimple(expr\_cast.expr.tipo)  !mismoTipo(expr.tipo\_convertido, expr.tipo) | expr\_cast.tipo = tipo.tipo  expr\_cast.modificable=false |
| expr\_acceso\_vector:expr → fuera:expr dentro:expr | fuera.tipo==tipoArray  dentro.tipo==tipoInt | expr\_acceso\_vector.tipo = tipoArray  expr\_acceso\_vector.modificable=true |
| expr\_acceso\_struct:expr → struct:expr campo:String | struct.tipo.def.campos[nombre == campo] | expr\_acceso\_struct.tipo = tipoStruct  expr\_acceso\_struct.modificable=true |
| expr\_llamada\_funcion:expr → nombre:String parametros:expr\* | |expr\_llamada\_funcion.parametrosi| == |expr\_llamada\_funcion.definicion.parametrosi|  expr\_llamada\_funcion.parametrosi.tipo == expr\_llamada\_funcion.definicion.parametrosi.tipo  expr\_llamada\_funcion.definicion.retorno ≠ tipoVoid | expr\_llamada\_funcion.tipo = expr.tipo  expr\_llamada\_funcion.modificable=false |

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

### Atributos

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Nodo/Categoría Sintáctica** | **Nombre del Atributo** | **Tipo Java** | **Heredado/Sintetizado** | **Descripción** |
| expr | tipo | Tipo | sintetizado | Indica el tipo de los elementos de la expresión para cuando un predicado pida solo de cierto tipo |
| expr | modificable | boolean | sintetizado | Indica si el contenido de la expresión se puede asignar en una asignación o no |
| sentencia | funcionActual | Definicion\_funcion | heredado | Indica la funcion en la que estas, sirve para saber el tipo de retorno que tiene la funcion |

### Metodos auxiliares

|  |
| --- |
| tipoSimple(tipo)  tipo == TipoInt || tipo == TipoFloat || tipo == Tipochar |
| mismoTipo(tipo1, tipo2)  tipo1 == tipo2 |