# Attribute Grammar

|  |  |  |
| --- | --- | --- |
| Nodo | Predicados | Reglas Semánticas |
| program → *bloque*:bloque\* |  |  |
|  |  |  |
| definicion\_variable\_struct → *nombre*:String *tipo*:tipo | campos\_struct[nombre]== ∅ | campos\_struct [nombre] = definicion\_variable\_struct |
| tamanio\_vector → *numero*:int |  |  |
| parametro → *nombre*:String *tipo*:tipo | parametros[nombre]== ∅ | parametros[nombre]= parametro |
|  |  |  |
| definicion\_variable:bloque → *nombre*:String *tipo*:tipo | variables[nombre]== ∅ | variables[nombre]=definicion\_vairable |
| struct:bloque → *nombre*:String *definicion\_variable\_struct*:definicion\_variable\_struct\* | estructuras[nombre]== ∅ | estructuras[nombre]=struct  {  campos\_struct.set()  visit(definicion\_variable\_struct i)  campos\_struct.reset()  } |
| funcion:bloque → *nombre*:String *parametros*:parametro\* *retorno*:tipo\* *sentencia*:sentencia\* | funcion[nombre] == ∅  tipoSimple(retorno.tipo)  tipoSimple(parametroi) | funcion [nombre] = funcion  {  parametros.set()  variables\_locales.set()  visit(parametro i)  variables\_locales.reset()  parametros.reset()  }  Sentenciasi.funcionEnLaQueEstoy = defFuncion(en java es this) |
|  |  |  |
| definicion\_variable\_funcion:sentencia → *nombre*:String *tipo*:tipo | variables\_funcion[nombre]== ∅  parametros[nombre]== ∅ | variables\_funcion[nombre]=definicion\_variable\_funcion |
| sentencia\_asignacion:sentencia → *izquierda*:expr *derecha*:expr | mismoTipo(izquierda.tipo, derecha.tipo)  tipoSimple(izquierda.tipo)  izquierda.modificable == true |  |
| sentencia\_print:sentencia → *expresiones*:expr | expresiones.tipo ≠ null  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.funcionEnLaQueEsto = sentencia\_if.funcionEnLaQueEsto |
| sentencia\_while:sentencia → *condicion*:expr *sentencias*:sentencia\* | condicion.tipo==tipoInt | Sentenciasi.funcionEnLaQueEsto = sentencia\_while.funcionEnLaQueEsto |
| sentencia\_return:sentencia → *expresion*:expr | expresion.tipo == funcion.retorno.tipo | Expr =/= null  return.funcionEnLaQueEstoy… |
| sentencia\_expresion:sentencia → *expresion*:expr |  |  |
|  |  |  |
| tipoInt:tipo → λ |  |  |
| tipoFloat:tipo → λ |  |  |
| tipoChar:tipo → λ |  |  |
| tipoVar:tipo → *string*:String | estructuras[nombre] ≠ ∅ | tipoVar.definicion = estructuras[nombre] |
| tipoArray:tipo → *tamanio*:String *tipo*: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\_ident:expr → *string*:String | variables[nombre]= ∅  variables\_funcion[nombre]≠ ∅  parametros[nombre]≠ ∅ | expr\_ident.definicion = variables[nombre]  expr\_ident.definicion = variables\_funcion [nombre]  expr\_ident.definicion = parametros [nombre]  expr\_ident.tipo=expr\_ident.definicion.tipo expr\_ident.modificable=true |
| expr\_binaria:expr → *izquierda*:expr *operador*:operador *derecha*:expr | mismoTipo(izquierda.tipo, derecha.tipo)  si(operador es aritmético)  tipoSimple(izquierda.tipo)  si(operador es lógico)  tipoSimple(izquierda.tipo)  si(operador es booleano)  izquierda.tipo==tipoInt | 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*:tipo *expr*:expr | tipoSimple(tipo.tipo)  tipoSimple(expr.tipo)  mismoTipo(tipo.tipo, expr.tipo) | expr\_cast.tipo = tipo.tipo  expr\_cast.modificable=false |
| expr\_llamada\_funcion:expr → *nombre*:String *parametros*:expr\* | funciones [nombre] ≠ ∅  expr\_llamada\_funcion.definicion.parametros.tipo == parametros.tipo | funciones.definition = funciones[nombre]  expr\_llamada\_funcion.tipo = expr.tipo  expr\_llamada\_funcion.modificable=false |
|  |  |  |
| operador\_aritmetico:operador → *operador\_aritmetico*:operador |  |  |
| operador\_logico:operador → *operador\_logico*:operador |  |  |
| operador\_booleano:operador → *operador\_booleano*:operador |  |  |

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

# Atributos

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

# Conjuntos Auxiliares

|  |  |
| --- | --- |
| funciones | Map<String, funcion> |
| estructuras | HashMap<String, Struct> |
| campos\_struct | ContextMap<String, |