|  |
| --- |
|  |
| **Nodo** | **Predicados** | **Reglas Semánticas** |
| **program** → *ast*:AST\* |  |  |
|  |  |  |
| **func**:def → *name*:String *parameter*:parameter\*  *retorno*:type *defvar*:defVar\* *sentence*:sentence\* |  |  |
| **defVar**:def → *name*:String *type*:type |  |  |
| **parameter**:def → *name*:String *type*:type |  |  |
| **defStruct**:def → *name*:String *parameter*:parameter\* |  |  |
|  |  |  |
| **intType**:type → λ |  |  |
| **realType**:type → λ |  |  |
| **charType**:type → λ |  |  |
| **arrayType**:type → *index*:int *type*:type |  |  |
| **structType**:type → *name*:String |  |  |
| **voidType**:type → λ |  |  |
|  |  |  |
| **print**:sentence → *string*:String *expr*:expr | EsPrimitivo(expr.type) |  |
| **read**:sentence → *expr*:expr |  |  |
| **assignment**:sentence → *left*:expr *right*:expr | mismoTipo(left,right)  left.lValue |  |
| **ifSentence**:sentence → *condition*:expr *iftrue*:sentence\* | Condition.type == intType |  |
| **ifElseSentence**:sentence → *condition*:expr *iftrue*:sentence\* *else1*:sentence\* | Condition.type == intType |  |
| **whileSentence**:sentence → *condition*:expr *sentence*:sentence\* |  |  |
| **returnNode**:sentence → *expr*:expr |  |  |
| **funcCall**:sentence → *name*:String *args*:expr |  |  |
|  |  |  |
| **exprAritmetica**:expr → *left*:expr *op*:String *right*:expr | esPrimitivo(left.type)  EsPrimitivo(right.type)  If(op = %)  left.tipo = int right.tipo =int  Else  IsNumber(left)  isNumber(right) | exprAritmetica.type = left.type.aritmetica(right.type)  exprAritmetica.lValue = false |
| **exprLogica**:expr → *left*:expr *op*:String *right*:expr | esPrimitivo(left.type)  EsPrimitivo(right.type)  If(op = && || op = ||) left.tipo = int right.tipo =int  Else  IsNumber(left)  isNumber(right) | exprLogica.type = left.type.logica(right.type)  expr.Logica.lValue = false |
| **exprLogicaNe**:expr → *expr*:expr |  | exprLogica.type = expr.type  expr.Logica.lValue = false |
| **acces**:expr → *left*:expr *right*:String | Left.type == struct.type | acces.type = definicion.getTypeOf()  acces.lValue = true |
| **arrayAcces**:expr → *left*:expr *right*:expr | Left.type == arrayType  Right.type == intType | arrayAcces.type = arrayType  arryAcces.lValue = true |
| **cast**:expr → *typeToConvert*:type *expr*:expr | Expr.type != typeToConvert  esPrimitivo(typeToConvert)  esPrimitivo(expr.type) | cast.type = typeToConvert  Cast.lValue = false |
| **litEnt**:expr → *string*:String |  | litEnte.type = IntType  litEnte.lValue = false |
| **litReal**:expr → *string*:String |  | litReal.type = realType  litReal.lValue = false |
| **litChar**:expr → *string*:String |  | litChar.type = charType  litChar.lValue=false |
| **variable**:expr → *string*:String |  | Variable.type = variable.definicion.type  Variable.lValue = true |
| **methodCallExpr**:expr → *name*:String *args*:expr\* |  | methodCallExpr.type= methodCallExpr.definiciom.retorno  methodCallExpr.lValue = false |
|  |  |  |

**Atributos**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Nodo/Categoría Sintáctica** | **Nombre del Atributo** | **Tipo Java** | **Heredado/Sintetizado** | **Descripción** |
| expresion | type | Type | Sintetizado |  |
| expresion | lValue | boolean | Sintetizado |  |