Nodo

program → definitions: definition*

defVariable: definition → name: String type: type

defStruct: definition → name: String param: structParam*

structParam: definition → name: String type: type

defFunc: definition → name: String args: defVariable* returnType: type
definitions: defVariable* sentences: sentence*

intType: type → λ
realType: type → λ
charType: type → λ
varType: type → λ
arrayType: type → size: intConstant type: type
errorType: type → λ
voidType: type → λ

- assignment: sentence → left: expression right: expression

ifElse: sentence → expression: expression if_sent: sentence*
else_sent: sentence*

while: sentence → param: expression sentence: sentence*

return: sentence → expression: expression

read: sentence → expression: expression

print: sentence → expression: expression

println: sentence → expression: expression

printsp: sentence → expression: expression

funcSentence: sentence → name: String args: expression*

- arrayCall: expression → expr: expression index: expression
- fieldAccess: expression → expression: expression name: String
- arithmethicExpr: expression → left: expression operator: String
right: expression
- comparationExpr: expression → left: expression operator: String
right: expression
- logicExpr: expression → left: expression operator: String right: expression
- negationExpr: expression → operator: String expression: expression
- castExpr: expression → type: type expression: expression
- funcExpr: expression → name: String args: expression*

	Predicados	Reglas semánticas
arithmethicExpr	mismoTipo (left.tipo, right.tipo) left.type ∈ tiposSimples	arithmethicExpr.type = left type arithmethicExpr.modificable = false
comparableExpr	mismotipo (left.tipo, right.tipo) left.type ∈ tiposSimples	comparableExpr.type = left.type comparableExpr.modificable = false
logicExpr	mismoTipo (left tipo, right.tipo) left.type == intType	logicExpr.type = int Type logicExpr.modificable = false
negationExpr	sameType (expression.type, intType)	unaryExpression.type = expression.type unaryExpression.modificable = false
castExpr	expression.type ∈ Tipos simple !mismoTipo (type, expression.type) type ∈ tiposSimple	castExpr.type = type castExpr.modificable = false.
fieldAccess	mismoTipo (varType, fieldAccess.type) expresion.type.field ≠ ∅	fieldAccess.type = expression.type.field fieldAccess.modificable = true
funcExpr	funcExpr.args.size == args.size checkParams (funcExpr.args, args)	funcExpr.type = funcExp.definition.returnType funcExpr.modificable = false
arrayCall	index.type = intType index ≠ ∅ expr.type = arrayType	arrayCall.type = expr.type.type arrayCall.modificable = true

	Predicados	Reglas semánticas
assignment	mismoTipo(left.type, right.type) left.type ∈ tiposSimple left.modificable	
ifElse	mismoTipo(expression.type, intType)	Si no ifElse.definition.hasReturn ifElse.definition.hasReturn = checkReturns(if.sent, else.sent)
dowhile	mismoTipo(param.type, intType)	
return	expression.type ∈ tiposReturn mismoTipo(return.defFunc.returnType, expression.type)	return.def.hasReturn = true.
read	expression.type e tiposSimple expression.modificable	
funcSentence	args.size = funcSentence.args.size	