Attribute Grammar

Nodo	Predicados	Reglas Semánticas
program → *definitions*:definition*		
varDefinition:definition → *name*:String *type*:type		
structDefinition:definition → *name*:varType *definitions*:structField*		
funDefinition:definition → *name*:String *params*:definition* *return_t*:type *definitions*:varDefinition* *sentences*:sentence*	params.type ∈ tiposSimples return_t ∈ tiposSimples	
structField:definition → *name*:String *type*:type		
intType:type → λ		
realType:type → λ		
charType:type → λ		
varType:type → *type*:String		
voidType:type → λ		
arrayType:type → *size*:intConstant *type*:type	size > 0	
structType:type → *fields*:structField*		
errorType:type → λ		
print:sentence → *expression*:expression	expression.type ∈ tiposSimples	
printsp:sentence → *expression*:expression	expression.type ∈ tiposSimples	
println:sentence → *expression*:expression	expression.type ∈ tiposSimples	
read:sentence → *expression*:expression	expression.type ∈ tiposSimples expression.modificable	
assignment:sentence → *left*:expression *right*:expression	mismoTipo(left.type, right.type) left.modificable left.type ∈ tiposSimples	
return:sentence → *expression*:expression	expression.type ∈ tiposSimples mismoTipo(expression.type, return.definition.return_t)	return.definition.hasReturn = true
ifElse:sentence → *expression*:expression *if_s*:sentence* *else_s*:sentence*	expression.type == intType	
while:sentence → *expression*:expression *sentence*:sentence*	expression.type == intType	
funcInvocation:sentence → *name*:String *params*:expression*	params.type ∈ tiposSimples params.size == funcInvocation.definition.params.size	

	para cada param param.type == funcInvocation.definition.param.type	
variable:expression → *name*:String		variable.type = variable.definicion.type variable.modificable = true
intConstant:expression → *value*:String		intConstant.type = intType intConstant.modificable = false
realConstant:expression → *value*:String		realConstant.type = realType realConstant.modificable = false
charConstant:expression → *value*:String		charConstant.type = charType charConstant.modificable = false
voidConstant:expression → λ		voidConstatn.type = voidType voidConstant.modificable = false
funcInvocationExpression:expression → *name*:String *params*:expression*	params.type ∈ tiposSimples params.size == funcInvocation.definition.params.size para cada param param.type == funcInvocation.definition.param.type	funcInvocationExpression.type = funcInvocationExpression.definicion.r
arithmeticExpression:expression → *left*:expression *operator*:String *right*:expression	mismoTipo(left.type, rigth.type) left.type ∈ {intType, realType}	arithmeticExpression.type = left.type arithmeticExpression.modificable = fal
logicalExpression:expression → *left*:expression *operator*:String *right*:expression	mismoTipo(left.type, rigth.type) left.type == intType	logicalExpression.type = intType logicalExpression.modificable = false
unaryExpression:expression → *operator*:String *expr*:expression	expr.type == intType	unaryExpression.type = intType unaryExpression.modificable = false
comparableExpression:expression → *left*:expression *operator*:String *right*:expression	mismoTipo(left.type, rigth.type) left.type ∈ {intType, realType}	comparableExpression.type = intType comparableExpression.modificable = f
castExpression:expression → *type*:type *expr*:expression	expr.type ¡= type expr.type ∈ tiposSimples type ∈ tiposSimples	castExpression.type = type castExpression.modificable = false
fieldAccessExpression:expression → *expr*:expression *name*:String	expr.type == structType	fieldAccessExpression.type = (expression.type).field(name) fieldAccessExpression.modificable = t
indexExpression:expression → *expr*:expression *index*:expression	expr.type == arrayType index.type == intType	indexExpression.type = expr.type.type indexExpression.modificable = true
unarySumExpression:expression -> *operator*:String *expr*:expression	expr.type == intType expr.modificable	unaryExpression.type = intType unaryExpression.modificable = false

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

Atributos

Nodo/Categoría Sintáctica	Nombre del Atributo	Tipo Java	Heredado/Sintetizado	Descripción

funcInvocation	definition	FuncDefinition	Sintetizado	
funcInvocationExpression	definition	FuncDefinition	Sintetizado	
variable	definition	VarDefinition	Sintetizado	
expression	type	Type	Sintetizado	
expression	modificable	boolean	Sintetizado	