Comprobación de Tipos

Nodo	Predicados	Reglas Semánticas
programa → *definiciones*:definicion*		
parametro → *tipo*:tipo *nombre*:String	tipo ∈ tiposSimples	
tipoInt:tipo → λ		
tipoFloat:tipo → λ		
tipoChar:tipo → λ		
tipoVoid:tipo → λ		
tipoArray:tipo → *tipo*:tipo *dimensiones*:String		
tipoStruct:tipo → *nombre*:String		
defVariable:definicion → *tipo*:tipo *nombre*:String		
defAtributo:definicion → *tipo*:tipo *nombre*:String		
defStruct:definicion → *nombre*:String *atributos*:defAtributo*		
defFuncion:definicion → *nombre*:String *parametros*:parametro* *retorno*:tipo *declaraciones*:defVariable* *sentencias*:sentencia*	retorno ∉ { tipoArray, tipoStruct }	$sentencia_i$.funcion = defFuncion
llamadaFuncionExpresion:expresion → *nombre*:String *args*:expresion*	definicion.tipo != tipoVoid AND args.size() == definicion.parametros.size() AND ∀ (argumento ∈ args) ⇒ $argumento_i$.tipo == $definicion.parametros_i$.tipo	llamadaFuncionExpresion.tipo = llamadaFuncionExpresion.definicion.tipo llamadaFuncionExpresion.modificable = false
expresionAritmetica:expresion → *izquierda*:expresion *operador*:String *derecha*:expresion	izquierda.tipo ∈ { tipoInt, tipoFloat } AND derecha.tipo == izquierda.tipo	expresionAritmetica.tipo = izquierda.tipo expresionAritmetica.modificable = false
expresionBooleana:expresion → *izquierda*:expresion *operador*:String *derecha*:expresion	izquierda.tipo == tipoInt AND derecha.tipo == izquierda.tipo	expresionBooleana.tipo = tipoInt expresionBooleana.modificable = false
expresionComparativa:expresion → *izquierda*:expresion *operador*:String *derecha*:expresion	izquierda.tipo ∈ { tipoInt, tipoFloat } AND derecha.tipo == izquierda.tipo	expresionComparativa.tipo = tipoInt expresionComparativa.modificable = false
negacionBooleana:expresion → *expresion*:expresion	expresion.tipo == tipoInt	negacionBooleana.tipo = tipoInt negacionBooleana.modificable = false
accesoArray:expresion → *izquierda*:expresion *derecha*:expresion	izquierda.tipo == tipoArray AND derecha.tipo == tipoInt	accesoArray.tipo = izquierda.tipo.tipo accesoArray.modificable = true
accesoStruct:expresion → *expresion*:expresion *campo*:String	expresion.tipo == tipoStruct AND ∃ (atributo ∈ definicion.atributos) | atributo.nombre == campo	accesoStruct.tipo = expresion.tipo.definicion.campo.tipo accesoStruct.modificable = true
variable:expresion → *name*:String		variable.tipo = variable.definicion.tipo variable.modificable = true
cast:expresion → *tipo*:tipo *expresion*:expresion	tipo ∈ tiposSimples AND expresion.tipo ∈ tiposSimples AND tipo != expresion.tipo	cast.tipo = tipo cast.modificable = false
constanteInt:expresion → *valor*:String		constanteInt.tipo = tipoInt constanteInt.modificable = false

constanteFloat:expresion → *valor*:String		constanteFloat.tipo = tipoFloat constanteFloat.modificable = false
constanteChar:expresion → *valor*:String		constanteChar.tipo = tipoChar constanteChar.modificable = false
asignacion:sentencia → *variable*:expresion *valor*:expresion	variable.modificable AND variable.tipo == valor.tipo AND variable ∈ tiposSimples	
read:sentencia → *variable*:expresion	variable.tipo ∈ tiposSimples AND variable.modificable	
if:sentencia → *condicion*:expresion *cierto*:sentencia* *falso*:sentencia*	condicion.tipo == tipoInt	∀ (sentencia ∈ cierto) ⇒ $sentencia_i$.funcion = if.funcion ∀ (sentencia ∈ falso) ⇒ $sentencia_i$.funcion = if.funcion
while:sentencia → *condicion*:expresion *sentencias*:sentencia*	condicion.tipo == tipoInt	$sentencia_i$.funcion = while.funcion
print:sentencia → *expresion*:expresion	expresion.tipo ∈ tiposSimples	
printsp:sentencia → *expresion*:expresion	expresion.tipo ∈ tiposSimples	
println:sentencia → *expresion*:expresion	expresion.tipo ∈ tiposSimples	
llamadaFuncionSentencia:sentencia → *nombre*:String *args*:expresion*	args.size() == definicion.parametros.size() AND ∀ (argumento ∈ args) ⇒ $argumento_i$.tipo == $definicion.parametros_i$.tipo	
return:sentencia → *expresion*:expresion	funcion.retorno ∈ tiposSimples AND funcion.retorno == expresion.tipo	
returnVacio:sentencia → λ	funcion.tipo == tipoVoid	

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

Atributos

Nodo/Categoría Sintáctica	Nombre del Atributo	Tipo Java	Heredado/Sintetizado	Descripción
expresion	tipo	Tipo	Sintetizado	Tipo de la expresión
expresion	modificable	boolean	Sintetizado	Determina si puede aparecer a la izquierda de una asignación
sentencia	funcion	DefFuncion	Heredado	Función en la que aparece la sentencia

Conjuntos

tiposSimples = { tipoInt, tipoFloat, tipoChar }