Attribute Grammar

Nodo	Predicados	Reglas Semánticas
programa → *definiciones*:Definicion*		
DefVariable:Definicion → *nombre*:String *tipo*:Tipo		
DefFuncion:Definicion → *nombre*:String *tipo*:Tipo *defVariableLocal*:Definicion* *sentencias*:Sentencia*	tipo ∈ { TipoInt, TipoFloat, TipoChar } OR tipo == null	DefFuncion.sentencias$_i$.funcion=DefFuncion
DefStruct:Definicion → *nombre*:String *campos*:Definicion* *tipo*:Tipo		
DefCampoStruct:Definicion → *nombre*:String *tipo*:Tipo		
Print:Sentencia → *expresion*:Expresion	expresión.tipo ∈ { TipoInt, TipoFloat, TipoChar }	
PrintLn:Sentencia → *expresion*:Expresion	expresión.tipo ∈ { TipoInt, TipoFloat, TipoChar }	
PrintSp:Sentencia → *expresion*:Expresion	expresión.tipo ∈ { TipoInt, TipoFloat, TipoChar }	
Asignacion:Sentencia → *expIzq*:Expresion *expDrcha*:Expresion	expIzq.tipo ∈ { TipoInt, TipoFloat, TipoChar } expIzq.tipo == expDrcha.tipo expIzq.modificable==true	
If:Sentencia → *condicion*:Expresion *sentenciasIf*:Sentencia*	condicion.tipo==TipoInt	sentencias$_i$.funcion=If.funcion

IfElse:Sentencia → *condicion*:Expresion *sentenciasIf*:Sentencia* *sentenciasElse*:Sentencia*	condicion.tipo==TipoInt	sentencias$_i$.funcion=IfElse.funcion
While:Sentencia → *condicion*:Expresion *sentencias*:Sentencia*	condicion.tipo==TipoInt	sentencias$_i$.funcion=While.funcion
InvocacionProcedimiento:Sentencia → *nombre*:String *argumentos*:Expresion*	|InvocacionProcedimiento.argumentos|==|InvocacionProcedimiento.definicion.tipo.defParametros| argumentos$_i$.tipo==InvocacionProcedimiento.definicion.tipo.defParametros$_i$.tipo	
Return:Sentencia → *expresion*:Expresion	expresión.tipo ∈ { TipoInt, TipoFloat, TipoChar} SI Return.funcion.tipo==TipoVoid expresion==null SINO expresión.tipo ∈ { TipoInt, TipoFloat, TipoChar} Return.funcion.tipo==expresion.tipo	
Read:Sentencia → *expresion*:Expresion	expresión.tipo ∈ { TipoInt, TipoFloat, TipoChar} expresion.modificable==true	
ExprAritmetica:Expresion → *expIzq*:Expresion *operator*:String *expDrcha*:Expresion	expIzq.tipo == expDrcha.tipo expIzq.tipo ∈ { TipoInt, TipoFloat, TipoChar}	ExprAritmetica.tipo=exprIzq.tipo ExprAritmetica.modificable=false

ExprComparacion:Expresion → *expIzq*:Expresion *operator*:String *expDrcha*:Expresion	expIzq.tipo == expDrcha.tipo expIzq.tipo ∈ {TipoInt, TipoFloat}	ExprComparacion.tipo=TipoInt ExprAritmetica.modificable=false
ExprLogica:Expresion → *expIzq*:Expresion *operator*:String *expDrcha*:Expresion	expIzq.tipo==TipoInt expIzq.tipo == expDrcha.tipo	ExprLogica.tipo=TipoInt ExprAritmetica.modificable=false
ExprMenosUnario:Expresion → *expr*:Expresion	expr.tipo ∈ { TipoInt, TipoFloat, TipoChar}	SI (expr.tipo==TipoChar) ExprMenosUnario.tipo=TipoInt SINO ExprMenosUnario.tipo=expr.tipo ExprAritmetica.modificable=false
ExprNot:Expresion → *expr*:Expresion	Expr.tipo==TipoInt	ExprNot.tipo=TipoInt ExprAritmetica.modificable=false
AccesoArray:Expresion → *identificador*:Expresion *posicion*:Expresion	identificador.tipo==TipoArray posicion.tipo==TipoInt	AccesoArray.tipo=identificador.tipo.tipoDe AccesoArray.modificable=true
ExpresionCast:Expresion → *tipoCast*:Tipo *expresion*:Expresion	expresion.tipo ∈ { TipoInt, TipoFloat, TipoChar} tipoCast ∈ { TipoInt, TipoFloat, TipoChar} expresion.tipo≠tipoCast	ExpresionCast.tipo=tipoCast ExpresionModificable=false

AccesoStruct:Expresion → *expresion*:Expresion *identificador*:String	expresion.tipo==TipoStruct expresión.tipo.definicion.campos[nombre==identificador] ≠ ∅	AccesoStruct.tipo=expresión.tipo.definicion. campos[nombre==identificador] AccesoStruct.modificable=true
ExpresionInvocacion:Expresion → *nombre*:String *argumentos*:Expresion*	|ExpresionInvocacion.argumentos|==|ExpresionInvocacion.definicion.tipo.DefParametros| ExpresionInvocacion.argumentos$_i$.tipo==ExpresionInvocacion.definicion.tipo.defParametros$_i$.tipo ExpresionInvocacion.definicion.tipo == TipoFuncion	ExpresionInvocacion.tipo= ExpresionInvocacion. definicion.tipo.tipoRetorno
CTE_Real:Expresion → *valor*:String		CTE_Real.tipo=TipoFloat CTE_Real.modificable=false
CTE_Entera:Expresion → *valor*:String		CTE_Entera.tipo=TipoInt CTE_Entera.modificable=false
CTE_Char:Expresion → *lexema*:String		CTE_Char.tipo=TipoChar CTE_Char.modificable=false
Variable:Expresion → *nombre*:String		Variable.tipo=definicion.tipo Variable.modificable=true
TipoInt:Tipo → λ		
TipoStruct:Tipo → *identificador*:String		
TipoFloat:Tipo → λ		

TipoChar:Tipo → λ		
TipoArray:Tipo → *size*:Integer *tipoDe*:Tipo		
TipoVoid:Tipo → λ		
TipoFuncion:Tipo → *tipoRetorno*:Tipo *defParametros*:Definicion*	defParametros.tipo ∈ { TipoInt, TipoFloat, TipoChar}	

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

Atributos

Categoría Sintáctica	Nombre del atributo	Tipo Java	Heredado/Sintetizado	Descripción
Expresion	tipo	Tipo	Sintetizado	Tipo de la expresion
Expresion	modificable	boolean	Sintetizado	Para saber si la expresion puede aparecer a la izquierda de una asignacion
Sentencia	Funcion	DefFuncion	Heredado	Indica en que funcion estamos