Attribute Grammar

Nodo	Predicados	Reglas Semánticas
programa → *definiciones*:definicion*		
definicionFuncion:definicion → *nombre*:String *params*:definicionVariable* *tipo*:tipo *variablesLocales*:definicionVariable* *sentencias*:sentencia*	Si tipo ≠ TipoVoid: esTipoPrimitivo(tipo)	
definicionVariable:definicion → *nombre*:String *tipo*:tipo	Si definicionVariable.ambito == Parametro: esTipoPrimitivo(definicionVariable.tipo)	
definicionStruct:definicion → *nombre*:String *campos*:campo*		
campo → *nombre*:String *tipo*:tipo		
tipoEntero:tipo → λ		
tipoReal:tipo → λ		
tipoChar:tipo → λ		
tipoArray:tipo → *longitud*:int *tipo*:tipo		
tipoStruct:tipo → *nombre*:String campos:campo*		
tipoVoid:tipo → λ		
print:sentencia → *expresiones*:expresion* *tipo_print*:String	Si expresiones.size() > 0 esTipoPrimitivo(expresiones.get(0).tipo)	
read:sentencia → *expresion*:expresion	esTipoPrimitivo(expresion.tipo) expresion.modificable == true	
asignacion:sentencia → *izquierda*:expresion *derecha*:expresion	esTipoPrimitivo(izquierda.tipo) esTipoPrimitivo(derecha.tipo) izquierda.modificable == true mismoTipo(izquierda.tipo, derecha.tipo)	
if:sentencia → *condicion*:expresion *verdadero*:sentencia* *falso*:sentencia*	condicion.tipo == TipoEntero	
while:sentencia → *condicion*:expresion *sentencia*:sentencia*	condicion.tipo == TipoEntero	
invocacion:sentencia → *nombre*:String *params*:expresion*	|params| == |definicion.params| $params_i$.tipo == definicion.$params_i$.tipo	
return:sentencia → *expresion*:expresion*	Si expresión.size() == 0: definicionFuncion.tipo == TipoVoid else: esTipoPrimitivo(expresión.get(0).tipo) definicionFuncion.tipo ==	

	expresión.get(0).tipo	
constanteEntero:expresion → *valor*:int		tipo = tipoEntero modificable = false
constanteReal:expresion → *valor*:double		tipo = tipoReal modificable = false
constanteChar:expresion → *valor*:String		tipo = tipoChar modificable = false
variable:expresion → *nombre*:String	Si es el nombre de un campo de Struct: TipoStruct.getCampo(nombre) ≠ null	Si es el nombre de un campo de Struct: Tipo = TipoStruct.getCampo(nombre).tipo Else: tipo = definicion.tipo modificable = true
expresionAritmetica:expresion → *izq*:expresion *operador*:String *der*:expresion	esTipoPrimitivo(izq.tipo) esTipoPrimitivo(der.tipo) izq.tipo ≠ tipoChar der.tipo ≠ tipoChar mismoTipo(izq.tipo, der.tipo)	tipo = izq.tipo Modificable = false
expresionLogica:expresion → *izq*:expresion *operador*:String *der*:expresion	izq.tipo == tipoEntero der.tipo == tipoEntero	tipo = izq.tipo modificable = false
comparacion:expresion → *izq*:expresion *operador*:String *der*:expresion	esTipoPrimitivo(izq.tipo) esTipoPrimitivo(der.tipo) izq.tipo ≠ tipoChar der.tipo ≠ tipoChar mismoTipo(izq.tipo, der.tipo)	tipo = TipoEntero modificable = false
expresionUnaria:expresion → *expresion*:expresion *operador*:String	expresion.tipo == tipoEntero	tipo = expresion.tipo modificable = false
conversion:expresion → *nuevoTipo*:tipo *expresion*:expresion	esTipoPrimitivo(nuevoTipo) esTipoPrimitivo(expresion.tipo) ~ mismoTipo(nuevoTipo, expresion.tipo)	tipo = nuevoTipo modificable = false
invocacionExpresion:expresion → *nombre*:String *params*:expresion*	|params| == |definicion.params| $params_i$.tipo == definicion.$params_i$.tipo definicion.tipo ≠ TipoVoid	tipo = definicion.tipo modificable = false
accesoArray:expresion → *array*:expresion *indice*:expresion	array.tipo == tipoArray indice.tipo == tipoEntero	tipo = array.tipo.tipo modificable = true

accesoCampo:expresion → *struct*:expresion *campo*:expresion	struct.tipo == TipoStruct	tipo = campo.tipo modificable = true

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	Indica si la expresión puede aparecer en el lado izquierdo de una asignación

Funciones auxiliares

- boolean esTipoPrimitivo(Tipo t): devuelve *true* si el tipo especificado es primitivo (char, entero o real).
- boolean mismoTipo(Tipo t1, Tipo t2): devuelve *true* si los dos tipos son el mismo.