Comprobación de Tipos

Nodo	Predicados	Reglas Semánticas
funcion:bloque → *nombre*:String *parametros*:parametro* *retorno*:tipo *locales*:definicion_variable_local* *sentencias*:sentencia*	tipoSimple(retorno.tipo) tipoSimple(parametro$_i$)	sentencias$_i$.funcionActual = funcion
sentencia_asignacion:sentencia → *izquierda*:expr *derecha*:expr	mismoTipo(izquierda.tipo, derecha.tipo) tipoSimple(izquierda.tipo) izquierda.modificable == true	
sentencia_print:sentencia → *expresiones*:expr	tipoSimple(expresiones.tipo)	
sentencia_read:sentencia → *expresiones*:expr	tipoSimple(expresiones.tipo) expresiones.modificable==true	
sentencia_if:sentencia → *condicion*:expr *sentencias*:sentencia* *sino*:sentencia*	condicion.tipo==tipoInt	sentencias$_i$.funcionActual = sentencia_if.funcionActual
sentencia_while:sentencia → *condicion*:expr *sentencias*:sentencia*	condicion.tipo==tipoInt	sentencias$_i$.funcionActual = sentencia_while.funcionActual
sentencia_llamada_funcion:sentencia → *nombre*:String *parametros*:expr*	| sentencia_llamada_funcion.parametros$_i$| == | sentencia_llamada_funcion.definicion.parametros$_i$| sentencia_llamada_funcion.parametros$_i$.tipo == sentencia_llamada_funcion.definicion.parametros$_i$.tipo	

sentencia_return:sentencia → *expresion*:expr	si expresion == null
	  sentencia_return.funcionActual.tipo == tipoVoid
	sino
	  sentencia_return.funcionActual.tipo == expresion.tipo	

expr_int:expr → *string*:String		expr_int.tipo = tipoInt expr_int.modificable = false
expr_real:expr → *string*:String		expr_real.tipo = tipoFloat expr_real.modificable=false
expr_char:expr → *string*:String		expr_char.tipo=tipoChar expr_chat.modificable=false
expr_ident:expr → *string*:String		expr_ident.tipo=expr_ident.definicion.tipo expr_ident.modificable=true

expr_binaria:expr → *izquierda*:expr *operador*:operador *derecha*:expr	si(operador es aritmético)
	  tipoSimple(izquierda.tipo)
	si(operador es booleano)
	  tipoSimple(izquierda.tipo)
	si(operador es lógico)
	  izquierda.tipo==tipoInt
	mismoTipo(izquierda, derecha)
		expr_binaria.tipo=izquierda.tipo
		expr_binaria.modificable=false

expr_negada:expr → *operador*:operador *derecha*:expr	derecha.tipo==tipoInt	expr_binaria.tipo=izquierda.tipo expr_binaria.modificable=false
expr_vector:expr → *fuera*:expr *dentro*:expr	fuera.tipo==tipoArray dentro.tipo==tipoInt	expr_vector.tipo = tipoArray expr_vector.modificable=true
expr_punto:expr → *izquierda*:expr *derecha*:expr	izquierda.tipo == tipoStruct derecha ∈ izquierda.tipo.definicion.Definicion_campo_struct	expr_punto.tipo = derecha.tipo expr_punto.modificable=true
expr_parentesis:expr → *expr*:expr		expr_parentesis.tipo = expr.tipo expr_parentesis.modificable=expr.modificable
expr_cast:expr → *tipo_convertido*:tipo *expr*:expr	tipoSimple(expr_cast.tipo_convertido) tipoSimple(expr_cast.expr.tipo) !mismoTipo(expr.tipo_convertido, expr.tipo)	expr_cast.tipo = tipo.tipo expr_cast.modificable=false
expr_llamada_funcion:expr → *nombre*:String *parametros*:expr*	|expr_llamada_funcion.parametros$_i$| == |expr_llamada_funcion.definicion.parametros$_i$| expr_llamada_funcion.parametros$_i$.tipo == expr_llamada_funcion.definicion.parametros$_i$.tipo expr_llamada_funcion.definicion.retorno ≠ tipoVoid	expr_llamada_funcion.tipo = expr.tipo expr_llamada_funcion.modificable=false

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

Atributos

Categoría Sintáctica	Nombre del atributo	Tipo Java	Heredado/Sintetizado	Descripción
expr	modificable	boolean	sintetizado	Indica si el contenido de la expresión se puede asignar en una asignación o no
expr	tipo	Tipo	sintetizado	Indica el tipo de los elementos de la expresión para cuando un predicado pida solo de cierto tipo
sentencia	funcionActual	Funcion	heredado	Indica la funcion en la que estas, sirve para saber el tipo de retorno que tiene la funcion

Metodos auxiliares

tipoSimple(tipo)

```
tipo == TipoInt || tipo == TipoFloat || tipo == Tipochar
```

mismoTipo(tipo1, tipo2)

```
tipo1 == tipo2
```