Pablo Baragaño Coto
UO251759 – 32894067L

Attribute Grammar

Nodo	Predicados	Reglas Semánticas
programa → *definicion*:definicion*		
cuerpoStruct → *ident*:String *tipo*:tipo		
defVariable:definicion → *ident*:String *tipo*:tipo		
defStruct:definicion → *ident*:String *cuerpostruct*:cuerpoStruct*		
defFuncion:definicion → *ident*:String *parametrosFuncion*:defVariable* *tipo*:tipo *definiciones*:defVariable* *sentencias*:sentencia*	tipo ≠ tipoStruct tipo ≠ tipoArray	
tipoEntero:tipo → λ		
tipoReal:tipo → λ		
tipoChar:tipo → λ		
tipoArray:tipo → *dimension*:String *tipo*:tipo		
tipoStruct:tipo → *ident*:String		
tipoVoid:tipo → λ		
return:sentencia → *expresion*:expresion	si expresion==null: return.funcionEnLaQueEstoy.tipo==TipoVoid si expresion!=null && return.funcionEnLaQueEstoy.tipo != TipoVoid: return. funcionEnLaQueEstoy.tipo == expresion.tipo si no: return.expresion == TipoVoid	
asignacion:sentencia → *izq*:expresion *der*:expresion	izq.tipo ≠tipoStruct izq.tipo ≠tipoArray izq.tipo==der.tipo izq.modificable==true	

print:sentencia → *expresion*:expresion	expresion≠null expresion.tipo≠tipoStruct expresion.tipo≠tipoArray	
println:sentencia → *expresion*:expresion	expresion.tipo≠tipoStruct expresion.tipo≠tipoArray	
printsp:sentencia → *expresion*:expresion	expresion≠null expresion.tipo≠tipoStruct expresion.tipo≠tipoArray	
read:sentencia → *expresion*:expresion	Expresion.modificable == true Expresion.tipo≠tipoArray Expresion.tipo≠tipoStruct	
bucleWhile:sentencia → *condicion*:expresion *cuerpo*:sentencia*	condicion.tipo==tipoEntero	
sentenciaCondicional:sentencia → *condicion*:expresion *cuerpoIf*:sentencia* *cuerpoElse*:sentencia*	condicion.tipo==tipoEntero	
invocacionFuncionSentencia:sentencia→ *id*:String *parametros*:expresion*	|parámetros|== |invocacionFuncionSentencia.definicion.parametros| parametros$_j$.tipo==invocacionFuncionSentencia.definicion.parametros$_i$.tipo	
invocacionFuncionExpresion: expresion → *id*:String *parametros*:expresion*	| parámetros|== |invocacionFuncionSentencia.definicion.parametros| parametros$_j$.tipo==invocacionFuncionSentencia.definicion.parametros$_i$.tipo invocacionFuncionExpresion.definicion.tipo ¡= TipoVoid (o null, si no tienes tipo void)	invocacionFuncionExpresion.tipo=invocacionFuncionExpresion.definicion.tipo invocacionFuncionExpresion.modificable=false
accesoStruct:expresion → *expresion*:expresion *ident*:String	Expresion.tipo==TipoStruct Expresion.tipo.definicion.campos[ident==ident] ≠0 (método en la clase estructura un método buscarCampo que le pasas un string devuelve el campo)	accesoStruct.modificable = true
accesoArray:expresion → *ident*:expresion *posicion*:expresion	Expresion.tipo==TipoArray Posición.tipo==TipoEntero	accesoArray.modificable = true

variable:expresion → *ident*:String		variable.tipo=variable.definition.tipo variable.modificable=true
literalInt:expresion → *value*:String		literalInt.tipo=tipoEntero literalInt.modificable=false
literalReal:expresion → *value*:String		literalReal.tipo=tipoReal literalReal.modificable=false
literalChar:expresion → *value*:String		literalChar.tipo=tipoChar literalChar.modificable=false
cast:expresion → *tipo*:tipo *expresion*:expresion	tipo≠tipoStruct tipo≠tipoArray tipo≠expresion.tipo expresion.tipo≠tipoStruct expresion.tipo≠tipoArray	cast.tipo=tipo cast.modificable=false
negacion:expresion → *expresion*:expresion	expresion.tipo=intType	negacion.tipo=tipoEntero negacion.modificable=false
expresionAritmetica:expresion → *izquierda*:expresion *operador*:String *derecha*:expresion	izquierda.tipo==derecha.tipo izquierda.tipo==tipoEntero izquierda.tipo==tipoReal	expresionAritmetica.tipo=izquierda.tipo expresionAritmetica.modificable=false
expresionBinaria:expresion → *izquierda*:expresion *operador*:String *derecha*:expresion	expresion.izquierda.tipo ≠ tipoStruct expresion.izquierda.tipo ≠ tipoArray expresion.izquierda.tipo == expresion.derecha.tipo	expresionBinaria.tipo=izquierda.tipo expresionBinaria.modificable=false
expresionLogica:expresion → *izquierda*:expresion *operador*:String *derecha*:expresion	expresion.izquierda.tipo==expresion.derecha.tipo expresion.izquierda.tipo == tipoEntero	expresionLogica.tipo=izquierda.tipo expresionLogica.modificable=false

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 expresión (operaciones que admite)
expresion	modificable	boolean	Sintetizado	Indica si la expresión es o no modificable
sentencia	funcionEnLaQueEstoy	DefFuncion	Heredado	Indica en la función en la que está la sentencia estudiada

Pablo Baragaño Coto
UO251759 – 32894067L