ESQUEMA DE TRADUCCIÓN

INTEGRANTES:

- *Aguilar Castro Carlos Alfonso*
- *Bustamante Piza Karla Mireli*
- *Ugalde Vivo José Francisco*

Facultad de Ingeniería, UNAM

Grupo: 01

Compiladores

Producción	Reglas semánticas
PROGRAMA -> DECLARACIONES FUNCIONES	PROGRAMA -> {TS_pila.push(nuevaTS()); TT_pila .push(newTT()); dir = 0} DECLARACIONES FUNCIONES {PROGRAMA.code = FUNCIONES.code}
DECLARACIONES -> TIPO LISTA_VAR SCOLON DECLARACIONES	DECLARACIONES ->{TIPO_GLOBAL= tipo.tipo} TIPO LISTA_VAR SCOLON DECLARACIONES
DECLARACIONES -> TIPO_REGISTRO LISTA_VAR SCOLON DECLARACIONES	DECLARACIONES -> { TIPO_GLOBAL =TIPO_REGISTRO.tipo} TIPO_REGISTRO LISTA VAR SCOLON DECLARACIONES
TIPO_REGISTRO -> STRUCT START DECLARACIONES END	TIPO_REGISTRO→{TS_pila.push(nuevaTS()); TT_pila.push(nuevaTT()); SDir.push(dir) ;dir = 0; } STRUCT START DECLARACIONES END { SimbT = TS_pila.pop(); SimbT.tipoTab = TT_pila.pop(); tam = getTam(SimbT) ; dir = SDir.pop(); TIPO_REGISTRO.TIPO = TT_pila.getCima().insert(”estructura”, tam, SimbT)}
TIPO -> BASE TIPO_ARREGLO	TIPO ->BASE {BASE_GLOBAL = BASE.BASE} TIPO_ARREGLO {TIPO.tipo = TIPO_ARREGLO.tipo}
BASE -> INT	BASE-> INT {BASE.BASE =TT_pila.getCima().getTipo(’int’)}
BASE -> FLOAT	BASE -> FLOAT {BASE.BASE =TT_pila.getCima().getTipo(’float’)}
BASE -> DOUBLE	BASE -> DOUBLE {BASE.BASE =TT_pila.getCima().getTipo(’double’)}
BASE -> CHAR	BASE -> CHAR {BASE.BASE =TT_pila.getCima().getTipo(’car’)}
BASE -> SIN	BASE -> SIN {BASE.BASE =TT_pila.getCima().getTipo(’sin’)}
TIPO_ARREGLO -> LPAR NUM RPAR TIPO_ARREGLO	TIPO_ARREGLO -> NUM TIPO_ARREGLO1{ Si NUM.tipo = INT Entonces; Si NUM.dir > 0 Entonces TIPO_ARREGLO.tipo = TT_pila.getCima().insert(’arreglo’,NUM TIPO_ARREGLO1.TIPO); Sino Error(“No es válido el tamaño”) Fin Si Sino Error(“El tamaño del arreglo no es un número entero”) Fin Si}
TIPO_ARREGLO -> Epsilon	TIPO_ARREGLO -> Epsilon {TIPO_ARREGLO.tipo = BASE_GLOBLAL}
LISTA_VAR -> ID LISTA_VAR_P COMMA ID	LISTA_VAR -> LISTA_VAR_P , ID { Si no TS_pila.getCima().existe(ID) Entonces

	TS_pila.getCima().insert(ID, TIPO_GLOBAL, dir, 'var', NULL, NULL) dir <- dir + TT_pila.getCima().getTam(TIPO_GLOBAL) Sino Error("La variable ya existe") Fin Si}
LISTA_VAR_P -> ID	LISTA_VAR -> ID {Si no TS_pila.getCima().existe(ID) Entonces TS_pila.getCima().insert(ID, TIPO_GLOBAL, dir, 'var', NULO, NULO) dir <- dir + TT_pila.getCima().getTam(TIPO_GLOBAL) Sino Error("Ya se declaró la variable") Fin Si}

FUNCIONES -> DEF TIPO ID LPAR ARGUMENTOS RPAR START DECLARACIONES SENTENCIAS END FUNCIONES	FUNCIONES ->DEF TIPO ID {
	Si no TS_pila.getGlobal().existe(ID) Entonces
	TS_pila.push(nuevaTS())
	TT_pila.push(nuevaTT()) SDir.push(dir) dir = 0
	lista_ret = nueva_lista()
	Sino
	Error("La variable ya existe")
	Fin Si
	} (ARGUMENTOS) START DECLARACIONES SENTENCIAS END {
	Si cRet(lista_ret, tipo.tipo) Entonces
	L = nuevaEtiqueta()
	backpatch(SENTENCIAS.nextlist, L)
	genCode(etiqueta L)
	TS_pila.pop()
	TT_pila.pop()
	Sino
	Error("No coincide el tipo de retorno")
	Fin Si
	}

FUNCIONES -> Epsilon	FUNCIONES -> Epsilon
ARGUMENTOS -> LISTA_ARG	ARGUMENTOS -> LISTA_ARG {ARGUMENTOS.lista = LISTA_ARG.lista; ARGUMENTOS.num = LISTA_ARG.num}
ARGUMENTOS -> SIN	ARGUMENTOS -> SIN{ARGUMENTOS.lista = NULL; ARGUMENTOS.num = 0}
LISTA_ARG -> LISTA_ARG_P COMMA ARG LISTA_ARG	LISTA_ARG -> LISTA_ARG_P , ARG {LISTA_ARG.lista = LISTA_ARG_P.lista; LISTA_ARG.lista.añadir(ARG.tipo); LISTA_ARG.num = LISTA_ARG_P.num + 1}
LISTA_ARG -> ARG	LISTA_ARG → ARG {LISTA_ARG.lista = nuevaLista(); LISTA_ARG.lista.añadir(ARG.tipo); LISTA ARG.num = 1}
ARG -> TIPO_ARG ID	ARG → TIPO_ARG ID { Si no TS_pila.getCima().existe(ID) Entonces TS_pila.getCima().añadir(ID, tipo.tipo, dir, 'ARG' , NULL, NULL)

	dir←dir + TT_pila.getCima().getTam(tipo.tipo) ARG.tipo = tipo.tipo Sino Error("La variable ya existe") Fin Si}
TIPO_ARG -> BASE PARAM_ARR	TIPO_ARG -> BASE {BASE_GLOBAL = BASE.BASE;} param_arr {tipo.tipo = PARAM ARR.tipo}
PARAM_ARR -> LPAR RPAR PARAM_ARR1	PARAM_ARR.tipo -> PARAM_ARR{PARAM_ARR.tipo = ST_pila(getCima().insertar('arreglo'), 0, PARAM_ARR_P.tipo)}
PARAM_ARR -> Epsilon	PARAM_ARR -> Epsilon{PARAM:ARR.tipo = BASE_GLOBAL
SENTENCIAS -> SENTENCIA SENTENCIAS_P	SENTENCIAS -> SENTENCIAS_P {L = nuevaEtiqueta() backpatch(SENTENCIAS_P.nextlist, L) genCode(etiqueta L)} SENTENCIA
SENTENCIA -> IF E_BOOL THEN SENTENCIA END	SENTENCIA -> si E_BOOL entonces SENTENCIA_P fin {L = nuevaEtiqueta() backpatch(E_BOOL.trueList, L) SENTENCIA.nextlist = combinar(E_BOOL.falseList, SENTENCIA_P.nextlist) genCode(etiqueta L)}
SENTENCIA -> IF E_BOOL THEN SENTENCIA1 ELSE SENTENCIA2 END	SENTENCIA si E_BOOL entonces SENTENCIA_P sino SENTENCIA_S fin {L1 = nuevaEtiqueta() L2 = nuevaEtiqueta() backpatch(E_BOOL.trueList, L1) backpatch(E_BOOL.falseList, L2) SENTENCIA.nextlist = combinar(SENTENCIA_P.nextlist, SENTENCIA_S.nextlist) genCode(etiqueta L1) genCode('goto' SENTENCIA_P.nextlist[0]) genCode(etiqueta L2)}
SENTENCIA -> WHILE E_BOOL DO SENTENCIA END	SENTENCIA- > mientras E_BOOL hacer SENTENCIA_P fin {L1 = nuevaEtiqueta() L2 = nuevaEtiqueta() backpatch(SENTENCIA_P.nextlist,L1) backpatch(E_BOOL.trueList, L2) SENTENCIA.nextList = E_BOOL.falseList genCode(etiqueta L1) b genCode(etiqueta L2) genCode('goto' SENTENCIA_P.nextlist[0])}
SENTENCIA -> DO SENTENCIA _P WHILE E_BOOL SCOLON	SENTENCIA -> hacer {L = nuevaEtiqueta() genCode("etiqueta" L)} SENTENCIA1 mientras E_BOOL; {backpatch(SENTENCIA1.nextlist, L)}
SENTENCIA -> SWITCH LPAR VARIABLE RPAR DO CASES PREDETERMINADO END	SENTENCIA -> segun (VARIABLE) hacer predeterminado fin {L1 = nuevaEtiqueta()

	prue= combinar(CASE.prue, predeterminado.prue) backpatch(CASES.nextlist, L2) sustituir(”??”, VARIABLE.dir, prueba)}
SENTENCIA -> VARIABLE ASIGN2 EXPRESION SCOLON	SENTENCIA -> VARIABLE := EXPRESION ; {dir = reducir(EXPRESION.dir, EXPRESION.tipo, VARIABLE.tipo) Si VARIABLE.code_est = true Entonces genCode(VARIABLE.base’[’VARIABLE.des’]’ ’=’ dir) Sino genCode(VARIABLE.dir ’=’ dir) Fin Si}
SENTENCIA -> WRITE EXPRESION SCOLON	SENTENCIA -> WRITE EXPRESION ; {gen(”ESCRIBIR” EXPRESION.dir)}
SENTENCIA -> READ VARIABLE SCOLON	SENTENCIA -> READ EXPRESION ; {gen(”LEER” EXPRESION.dir)}
SENTENCIA -> RETURN SCOLON	SENTENCIA -> RETURN; {genCode(”DEVOLVER”)}
SENTENCIA -> RETURN EXPRESION SCOLON	SENTENCIA -> RETURN EXPRESION; {genCode(”DEVOLVER” expresion.dir) index = nuevoIndex()

SENTENCIA -> TERM SCOLON	SENTENCIA.nextlist = nuevoIndexList(index)
	genCode(”goto” index)}
	SENTENCIA -> END; {
	index = nuevoIndex();
	SENTENCIA.nextlist = nuevoIndexList(index)
	genCode(”goto” index)}

SENTENCIA -> START SENTENCIA_P END	SENTENCIA -> START SENTENCIAS_P END {SENTENCIA.nextlist = SENTENCIA_P.nextlist}
CASES -> CASES_P_CASE NUM DCOLON SENTENCIA	CASES -> CASES_P_CASE NUM:{L = nuevaEtiqueta() genCode(”etiqueta” L) } SENTENCIA {CASES.nextlist = combinar(CASES.nextlist, SENTENCIA1.nextlist) CASES.prue = CASE1.prue CASES.prue.append(if ”??” ”==” NUM.dir ”goto” L)}
CASES -> CASE NUM DCOLON SENTENCIA	CASES -> CASE NUM: {CASES.prue = nuevoCode() L = nuevoEtiqueta() /*Indica el inicio del codigo para la SENTENCIA*/ genCode(”etiqueta” L) } SENTENCIA {CASES.prue.añadir(if ”??” ”==” NUM.dir ”goto” L) CASES.nextlist = SENTENCIA.nextlist}
PREDETERMINADO -> PRED DCOLON SENTENCIA	PREDETERMINADO -> PRED: {PREDETERMINADO.prue = nuevoCode() L = nuevoEtiqueta() /*Inicio de código para la sentencia*/ genCode(”etiqueta” L) } SENTENCIA {PREDETERMINADO.prue.añadir(”goto” L)}

PRDETERMINADO -> Epsilon	PREDETERMINADO -> Epsilon {PREDETERMINADO.prue = NULL}
E_BOOL -> E_BOOL1 AND E_BOOL2	E_BOOL -> E_BOOL AND E_BOOL {L = nuevaEtiqueta() backpatch(E_BOOL.trueList, L) E_BOOL.trueList = E_BOOL1.trueList E_BOOL.falseList = combinar(E_BOOL1.falseList, E_BOOL2.falseList) genCode(etiqueta L)}
E_BOOL -> E_BOOL1 OR E_BOOL2	E_BOOL → E_BOOL1 OR E_BOOL2{ L = nuevaEtiqueta() backpatch(E_BOOL.falseList, L) E_BOOL.trueList = combinar(E_BOOL1.trueList, E_BOOL2.trueList) E_BOOL.falseList = E_BOOL2.falseList genCode(etiqueta L)}
E_BOOL -> NOT E_BOOL1	E_BOOL -> NOT E_BOOL1 {E_BOOL.trueList = E_BOOL1.falseList E_BOOL.falseList = E_BOOL.trueList E_BOOL -> RELACIONAL op E_BOOL -> RELACIONAL op {E_BOOL.trueList = RELACIONAL op.trueList E_BOOL.falseList = RELACIONAL op.falseList}
E_BOOL -> RELACIONAL	E_BOOL.trueList = RELACIONAL.trueList E_BOOL.falseList = RELACIONAL.falseList
E_BOOL -> TRUE	E_BOOL -> TRUE {index0 = nuevoIndex() E_BOOL.trueList = nuevoIndexList(index0) genCode('goto' index0)}
E_BOOL -> FALSE	E_BOOL -> FALSE {index0 = nuevoIndex() e bool.falseList = nuevoIndexList(index0) genCode('goto' index0)}
RELACIONAL -> EXPRESION	RELACIONAL.dir = EXPRESION.dir RELACIONAL.tipo = EXPRESION.tipo RELACIONAL.trueList = NULL RELACIONAL.falseList = NULL
RELACIONAL -> RELACIONAL1 MORE RELACIONAL2	RELACIONAL -> RELACIONAL1 > RELACIONAL2{index0 = nuevoIndex() index1 = nuevoIndex() RELACIONAL.trueList = nuevoIndexList(index0) RELACIONAL.falseList = nuevoIndexList(index1) genCode('if' RELACIONAL1.dir > RELACIONAL2 'goto' index0) genCode('goto' index1)
RELACIONAL -> RELACIONAL1 LESS RELACIONAL2	RELACIONAL -> RELACIONAL1 < RELACIONAL2 {index0 = nuevoIndex() index1 = nuevoIndex() RELACIONAL.trueList = nuevoIndexList(index0) RELACIONAL.falseList = nuevoIndexList(index1)

	genCode(’if’ RELACIONAL1.dir < RELACIONAL2 ’goto’ index0) genCode(’goto’ index1)}
RELACIONAL -> RELACIONAL1 MOREEQ RELACIONAL2	RELACIONAL -> RELACIONAL1 >= RELACIONAL2{index0 = nuevoIndex() index1 = nuevoIndex() RELACIONAL.trueList = nuevoIndexList(index0) RELACIONAL.falseList = nuevoIndexList(index1) genCode(’if’ RELACIONAL1.dir >= RELACIONAL2 ’goto’ index0) genCode(’goto’ index1)}
RELACIONAL -> RELACIONAL1 LESSEQ RELACIONAL2	RELACIONAL -> RELACIONAL1 <= RELACIONAL2{index0 = nuevoIndex() index1 = nuevoIndex() RELACIONAL.trueList = nuevoIndexList(index0) RELACIONAL.falseList = nuevoIndexList(index1) genCode(’if’ RELACIONAL1.dir <= RELACIONAL2 ’goto’ index0) genCode(’goto’ index1)}
RELACIONAL -> RELACIONAL1 NOTEQ RELACIONAL2	RELACIONAL -> RELACIONAL1 <> RELACIONAL2{index0 = nuevoIndex() index1 = nuevoIndex() RELACIONAL.trueList = nuevoIndexList(index0) RELACIONAL.falseList = nuevoIndexList(index1) genCode(’if’ RELACIONAL1.dir > RELACIONAL2 ’goto’ index0) genCode(’goto’ index1)}
RELACIONAL -> RELACIONAL1 ASIGN RELACIONAL2	RELACIONAL -> RELACIONAL1 = RELACIONAL2{index0 = nuevoIndex() index1 = nuevoIndex() RELACIONAL.trueList = nuevoIndexList(index0) RELACIONAL.falseList = nuevoIndexList(index1) genCode(’if’ RELACIONAL1.dir = RELACIONAL2 ’goto’ index0) genCode(’goto’ index1)}
EXPRESION -> LPAR EXPRESION1 RPAR	EXPRESION ->(EXPRESION1){EXPRESION.tipo = EXPRESION1.tipo EXPRESION.dir = EXPRESION1.dir}
EXPRESION -> VARIABLE	EXPRESION -> VARIABLE{EXPRESION.tipo = VARIABLE.tipo EXPRESION.dir = VARIABLE.dir}
EXPRESION -> NUM	EXPRESION -> NUM{EXPRESION.tipo = NUM.tipo EXPRESION.dir = NUM.dir}
EXPRESION -> CARAC	EXPRESION -> CARACTER{ EXPRESION.type =’carac’ Si TCadenas.existe(carac) Entonces EXPRESION.dir= TCadenas.getIndexStr(carac)

	Sino EXPRESION.dir=TCadenas.insert(carac) Fin Si}
EXPRESION -> EXPRESION1 PLUS EXPRESION2	EXPRESION -> EXPRESION1 + EXPRESION2{EXPRESION.tipo = max(EXPRESION1.tipo, EXPRESION2.tipo) EXPRESION.dir = nuevoTempo() dir1 = ampliar(EXPRESION1.dir, EXPRESION1.tipo, EXPRESION.tipo) dir2 = ampliar(EXPRESION2.dir, EXPRESION2.tipo, EXPRESION.tipo) getCode(EXPRESION.dir '=' dir1 '+' dir2)}
EXPRESION -> EXPRESION1 MINUS EXPRESION2	EXPRESION -> EXPRESION1 − EXPRESION2{EXPRESION.tipo = max(EXPRESION1.tipo, EXPRESION2.tipo) EXPRESION.dir = nuevoTempo() dir1 = ampliar(EXPRESION1.dir, EXPRESION1.tipo, EXPRESION.tipo) dir2 = ampliar(EXPRESION2.dir, EXPRESION2.tipo, EXPRESION.tipo) getCode(EXPRESION.dir '=' dir1 '-' dir2)}
EXPRESION -> EXPRESION1 MUL EXPRESION2	EXPRESION -> EXPRESION1 * EXPRESION2{EXPRESION.tipo = max(EXPRESION1.tipo, EXPRESION2.tipo) EXPRESION.dir = nuevoTempo() dir1 = ampliar(EXPRESION1.dir, EXPRESION1.tipo, EXPRESION.tipo) dir2 = ampliar(EXPRESION2.dir, EXPRESION2.tipo, EXPRESION.tipo) getCode(EXPRESION.dir '=' dir1 '*' dir2)}
EXPRESION -> EXPRESION 1 DIV EXPRESION2	EXPRESION -> EXPRESION1 / EXPRESION2{EXPRESION.tipo = max(EXPRESION1.tipo, EXPRESION2.tipo) EXPRESION.dir = nuevoTempo() dir1 = ampliar(EXPRESION1.dir, EXPRESION1.tipo, EXPRESION.tipo) dir2 = ampliar(EXPRESION2.dir, EXPRESION2.tipo, EXPRESION.tipo) getCode(EXPRESION.dir '=' dir1 '/' dir2)}
EXPRESION -> EXPRESION 1 MOD EXPRESION2	EXPRESION -> EXPRESION1 % EXPRESION{ Si EXPRESION1.tipo = entero and EXPRESION2.tipo = entero Entonces EXPRESION.tipo = max(EXPRESION1.tipo, EXPRESION2.tipo) EXPRESION.dir = nuevoTempo() dir1 = ampliar(EXPRESION1.dir, EXPRESION1.tipo, EXPRESION.tipo) dir2 = ampliar(EXPRESION2.dir, EXPRESION2.tipo, EXPRESION.tipo) getCode(EXPRESION.dir '=' dir1 '+' dir2) Sino Error("Sólo aplica en enteros") Fin Si}

VARIABLE -> ID
VARIABLE_COMP
	VARIABLE -> ID {
	Si TS_pila.getCima().existe(ID) Entonces
	ID_GLOBAL = ID
	Sino
	Error(“Variable no declarada”)
	Fin Si
	} VARIABLE_COMP {
	Si VARIABLE_COMP.code = true Entonces
	VARIABLE.dir=nuevoTempo()
	VARIABLE.tipo = VARIABLE_COMP.tipo
	genCode(VARIABLE.dir ’=’ ID’[’
	VARIABLE_COMP.des’]’)
	VARIABLE.base = ID.dir
	VARIABLE.code = true
	VARIABLE.des = VARIABLE_COMP.des
	Sino
	VARIABLE.dir = ID)
	VARIABLE.tipo = TS_pila.getCima().getTipo(ID)
	VARIABLE.code = false

VARIABLE_COMP -> DATO_EST_SIM	VARIABLE_COMP -> DATO_EST_SIM{ VARIABLE_COMP.tipo = DATO_EST_SIM.tipo VARIABLE_COMP.des = DATO_EST_SIM.des VARIABLE_COMP.code_est = DATO_EST_SIM.code_est}
VARIABLE_COMP -> ARREGLO	VARIABLE_COMP -> ARREGLO {VARIABLE_COMP.tipo = ARREGLO.tipo VARIABLE_COMP.des = ARREGLO.dir VARIABLE_COMP.code_est = true}
VARIABLE_COMP ->LPAR PARAMETROS RPAR	DATO_EST_SIM -> DATO_EST_SIM.ID {Si DATO_EST_SIM1.estruct = true Entonces Si DATO_EST_SIM1.tabla.existe(ID) Entonces DATO_EST_SIM.des = DATO_EST_SIM1.des + DATO_EST_SIM.tabla1.getDir(ID) TIPO_TEMPO=DATO_EST_SIM1.tabla.getTipo(ID) estTemp = DATO_EST_SIM1.tabla .tablaTipos.getName(TIPO_TEMPO) Si estTemp = ’Estructura’ Entonces DATO_EST_SIM.estruct= true DATO_EST_SIM.tabla= DATO_EST_SIM.tabla .tablaTipos.getTipoBase(TIPO_TEMPO).tabla Sino DATO_EST_SIM.estruct= false DATO_EST_SIM.tabla= NULL DATO_EST_SIM.type = DATO_EST_SIM1.tabla.getTipo(ID) FinSi DATO_EST_SIM.code_est=true Sino Error(“Variable no declarada”)

	FinSi Sino Error(“No es una estructura") FinSi}
DATO_EST_SIM -> Epsilon	DATO_EST_SIM -> Epsilon {tipoTempo = TS_pila.getCima().getTipo(ID) Si TT_pila.getCima().getName(tipoTempo) =’struct’ Entonces DATO_EST_SIM.estructura= true DATO_EST_SIM.tabla= TT_pila.getCima() .getTipoBase(tipoTempo).tabla DATO_EST_SIM.des = 0 Sino DATO_EST_SIM.estructura= false DATO_EST_SIM.type = TT_pila.getCima().getTipo(ID) Fin Si DATO_EST_SIM.code_est=false}
ARREGLO -> LPAR EXPRESION RPAR	ARREGLO -> [EXPRESION] {ARREGLO.tipo = TS_pila.getCima().getTipo(ID_GLOBAL) Si TT_pila.getCima().getName(ARREGLO.tipo) = ’arreglo’ Entonces Si EXPRESION.tipo = entero Entonces TIPO_TEMPO = TT_pila.getCima().getTipoBase(ARREGLO.tipo) tam = TT_pila.getCima().getTam(TIPO_TEMPO) ARREGLO.dir = nuevoTempo() genCode(ARREGLO.dir’=’ EXPRESION.dir ’*’ tam) Sino Error(“No es entero el tamaño del arreglo”) Fin Si Sino Error(“Variable asociada no es tipo arreglo”) Fin Si}
ARREGLO_P -> LPAR EXPRESION RPAR ARREGLO1	ARREGLO → ARREGLO1 [EXPRESION]{ARREGLO.tipo = TS_pila.getCima().getTipo(ARREGLO1.tipo) Si TT_pila.getCima().getName(ARREGLO.tipo) = ’arreglo’ Entonces Si EXPRESION.tipo = entero Entonces TIPO_TEMPO = TT_pila.getCima().getTipoBase(ARREGLO.tipo) tam = TT_pila.getCima().getTam(TIPO_TEMPO) DIR_TEMPO = nuevoTempo() ARREGLO.dir = nuevoTempo() genCode(DIR_TEMPO’=’ EXPRESION.dir ’*’ tam) genCode(ARREGLO.dir’=’ ARREGLO1.dir ’+’ DIR_TEMPO) Sino Error(“NO es entero el tamaño del arreglo”) Fin Si Sino

	Error(“Variable asociada no es tipo arreglo”) Fin Si}
PARAMETROS -> LISTA_PARAM	PARAMETROS -> LISTA_PARAM{PARAMETROS.lista = LISTA_PARAM.lista PARAMETROS.num = LISTA_PARAM.num}
PARAMETROS -> Epsilon	PARAMETROS -> Epsilon {PARAMETROS.lista = NULL PARAMETROS.num = 0}
LISTA_PARAM -> EXPRESION LISTA_PARAM_P	LISTA_PARAM -> LISTA_PARAM1 , EXPRESION{ LISTA_PARAM.lista = LISTA_PARAM1.lista LISTA_PARAM.lista.añadir(EXPRESION.tipo) LISTA_PARAM.num = LISTA_PARAM1 + 1}
LISTA_PARAM_P -> COMMA EXPRESION LISTA_PARAM_P	LISTA_PARAM -> EXPRESION{ LISTA_PARAM.lista = nuevaLista() LISTA_PARAM.lista.añadir(EXPRESION.tipo) LISTA_PARAM.num = 1}