Producción	**Reglas Semánticas**
programa → declaraciones funciones	STS.push(nuevaTS()) STT.push(nuevaTT()) dir = 0 programa.code = sentencias.code TablaDeCadenas = nuevaTablaDeCadenas()
declaraciones → tipo lista_var ; declaraciones	Tipo = tipo.tipo
declaraciones → tipo_registro lista_var ; declaraciones	Tipo = tipo_registro.tipo
declaraciones → ε	declaraciones.tipo = base
tipo_registro → estructura inicio declaraciones fin	STS.push(nuevaTS()) STT.push(nuevaTT()) pila_direcciones.push(dir) dir = 0 dir = pila_direcciones.pop( ) Ts = STS.pop( ) Tt = STT.pop( ) ts.tt = Tt tipo = STT.getCima().append(“estructura”,tam,Ts)
tipo → base tipo_arreglo	Base = base.base Tipo.tipo = tipo.arreglo.tipo
base → ent	base.base = ent
base → real	base.base = real
base → dreal	base.base = dreal
base → car	base.base = car
base → sin	base.base = sin
tipo_arreglo → [num] $tipo\_arreglo_1$	**Si** *num*.tipo = ent **Si** *num*.dir > 0 tipo_arreglo.tipo = TT.append(“array”,num.dir, $tipo\_arreglo_1$ .tipo) **Sino** Error(“El indice debe ser mayor a 0”) **Sino** Error(“El indice debe ser entero”) **Fin Si**
tipo_arreglo → ε	tipo_arreglo.tipo = base
lista_var → lista_var, id	**Si** !TS.existe(id) Entonces STS.getCima().append(id, dir, Tipo, ‘var’,nulo,-1) Dir ← dir + STT.getCima().getTam(Tipo) **Sino**

	error(“El id ya existe”) **Fin Si**
lista_var → id	**Si** !TS.existe(id) Entonces STS.getCima().append(id, dir, Tipo, ‘var’,nulo,-1) Dir ← dir + STT.getCima().getTam(Tipo) **Sino** error(“El id ya fue declarado”) **Fin Si**
funciones → def tipo id(argumentos) inicio declaraciones sentencias fin funciones	**Si** !STS.getCima().existe(id) **Entonces** STS.push(nuevaTS()) Sdir.push(dir) dir=0 lista_retorno = nuevaLista() **Si** cmpRet(lista_retorno, tipo.tipo) **Entonces** L =nuevaEtiqueta() backpatch(sentencias.nextlist, L) F.code = etiqueta(id) || sentencias.code || etiqueta(L) **Sino** Error(“el valor no corresponde al tipo de la Función”) **Fin Si** STS.pop() dir = pila_direcciones.pop( ) **Sino** Error(“El id ya fue declarado”) **Fin Si**
funciones → ε	
argumentos → lista_arg	argumentos.lista = lista_arg.lista argumentos.num = lista_arg.num
argumentos → sin	argumentos.lista = nulo argumentos.num = 0
lista_arg → $lista\_arg_1$, arg	lista_arg.lista = $lista\_arg_1$ .lista lista_arg.lista.append(arg.tipo) lista_arg.num = lista_arg.num +1
lista_arg → arg	lista_arg.lista = nuevaLista( ) lista_arg.lista.append(arg.tipo)
arg → tipo_arg id	**Si** STS.getCima().getId(id.lexval)= -1 Entonces STS.getCima().addSym(id.lexval,tipo,dir,”var”) dir = dir + STT.getCima().getTam(tipo) **Sino** Error(“El identificador ya fue declarado”) **Fin** arg.tipo = tipo_arg.tipo
tipo_arg → base param_arr	base = base.tipo tipo_arg.tipo = param_arr.tipo

param_arr → [ ] $param\_arr_1$	param_arr.tipo = STT.append("array",-, $param\_arr_1.tipo$)
param_arr → ε	param_arr.tipo = base
sentencias → $sentencias_1$ sentencia	L =nuevaEtiqueta() backpatch($sentencias_1$ .nextlist, L) sentencias.nextlist = sentencia.nextlist sentencias.code = $sentencias_1$ .code ‖etiqueta(L) ‖ sentencia.code
sentencias → sentencia	sentencias.nextlist = sentencia.nextlist sentencias.code = sentencia.code
sentencia → si e_bool entonces $sentencias_1$ fin	L =nuevaEtiqueta() backpatch(e_bool.truelist, L) sentencia.nextlist= combinar(e_bool.falselist, $sentencia_1$ .nextlist) sentencia.code = e_bool.code ‖ etiqueta(L) ‖ $sentencia_1$ .code
sentencia → si e_bool entonces $sentencia_1$ sino $sentencia_2$ fin	L1 = nuevaEtiqueta() L2 = nuevaEtiqueta() backpatch(e_bool.truelist, L1) backpatch(e_bool.falselist, L2) sentencia.nextlist = combinar($sentencia_1$ .nextlist, $sentencia_2$ .nextlist) sentencia.code = e_bool.code ‖ etiqueta(L1) ‖ $sentencia_1$ .code ‖ gen('goto' $sentencia_1$ .nextlist[0]) ‖ etiqueta(L2) ‖ $sentencia_2$ .code
sentencia → mientras e_bool hacer $sentencia_1$ fin	L1 = nuevaEtiqueta() L2 = nuevaEtiqueta() backpatch($sentencia_1$ .nextlist, L1) backpatch(e_bool.truelist, L2) sentencia.nextlist = e_bool.falselist sentencia.code = etiqueta(L1) ‖e_bool.code ‖ etiqueta(L2) ‖ $sentencia_1$ .code ‖ gen('goto' $sentencia_1$ .nextlist[0])
sentencia → hacer $sentencia_1$ mientras e_bool;	L1 = nuevaEtiqueta() L2 = nuevaEtiqueta() backpatch($sentencia_1$ .nextlist, L1) backpatch(e_bool.truelist, L2) sentencia.nextlist = e_bool.falselist sentencia.code = etiqueta(L2) ‖ $sentencia_1$ .code ‖ etiqueta(L1) ‖ e_bool.code ‖ gen('goto' $sentencia_1$ .nextlist[0])
sentencia → segun (variable) hacer casos predeterminado fin	L1 = nuevaEtiqueta() L2 = nuevaEtiqueta() backpatch(sentencia.truelist, L1) backpatch(sentencia.falselist, L2)

	sentencia.nextlist = combinar(casos.nextlist, predeterminado.nextlist) sentencia.code = variable.code || etiqueta(L1) || casos.code || gen('goto' casos.nextlist[0]) || etiqueta(L2) || predeterminado.code
sentencia → variable := expresion ;	sentencia.nextlist = nulo Si TS.existe(variable) Entonces tipo_variable = TS.getTipo(variable) t = reducir(expresion.dir, expresion.tipo, tipo_variable) sentencia.code=gen(variable '=' t) Sino error("La variable no ha sido declarada") Fin Si
sentencia → escribir expresion ;	sentencia.code=gen("print" expresion.dir) sentencia.listnext = nulo
sentencia → leer variable ;	sentencia.code=gen("scan" expresion.dir) sentencia.listnext = nulo
sentencia → devolver ;	sentencia.nextlist = nulo sentencia.code = gen(return)
sentencia → devolver expresion ;	sentencia.nextlist = nulo lista_retorno.append(expresion.tipo) sentencia.code = gen(return expresion.dir)
sentencia → terminar ;	L = nuevaEtiqueta() sentencia.code=gen('goto' L) sentencia.nextlist = nuevaLista() sentencia.nextlist.add(L)
sentencia → inicio sentencias fin	sentencia.nextlist = sentencias.nextlist
casos → caso num : sentencia casos1	L1 = nuevaEtiqueta() L2 = nuevaEtiqueta() backpatch(num.truelist, L1) backpatch(num.falselist, L2) casos.nextlist = combinar(sentencia.nextlist, casos1.nextlist) casos.code = num.dir || etiqueta(L1) || *sentencia* .code || gen('goto' sentencia.nextlist[0]) || etiqueta(L2) || casos1.code
casos → caso num : sentencia	L =nuevaEtiqueta() backpatch(num.truelist, L) casos.code = num.dir || etiqueta(L) || sentencia.code || gen('goto' sentencia.nextlis[0])
predeterminado → pred : sentencia	L =nuevaEtiqueta() backpatch(num.falselist, L) prdeterminado.code = num.dir || etiqueta(L) || sentencia.code || gen('goto' sentencia.nextlis[0])

predeterminado → ε	prdeterminado.code = nulo
e_bool → e_bool_1 o e_bool_2	L = nuevaEtiqueta() backpatch(e_bool_1 .falselist, L) e_bool.truelist = combinar(e_bool_1 .truelist, e_bool_2 .truelist) e_bool .falselist = e_bool_2 .falselist e_bool.code = e_bool_1 .code || etiqueta(L) || e_bool_2 .code
e_bool → e_bool_1 y e_bool_2	L = nuevaEtiqueta() backpatch(e_bool_1 .truelist, L) e_bool.truelist = e_bool_2 .truelist e_bool.falselist = combinar(e_bool_1 .falselist, e_bool_2 .falselist) e_bool .code = e_bool_1 .code || etiqueta(L) || e_bool_2 .code
e_bool →no e_bool_1	e_bool.truelist = e_bool_1 .falselist e_bool.falselist = e_bool_1 .truelist e_bool.code = e_bool_1 .code
e_bool → relacional	e_bool.truelist = relacional.truelist e_bool.falselist = relacional.falselist
e_bool → verdadero	t0 = nuevaEtiqueta() e_bool.truelist = crearlista() e_bool.truelist.add(t0) e_bool.code = gen('goto' t0) e_bool.falselist = nulo
e_bool → falso	t0 = nuevaEtiqueta() e_bool.truelist = nulo e_bool.falselist = crearlista() e_bool.falseelist.add(t0) e_bool.code = gen('goto' t0)
relacional → relacional1 > relacional2	relacional.dir = nuevaTemporal relacional.tipo = max(relacional1.tipo , relacional2.tipo) t1= ampliar(relacional1.dir,relacional1.tipo, relacional.tipo) t2= ampliar(relacional2.dir, relacional2.tipo, relacional.tipo) relacional.code = gen(relacional.dir'=' t1'>'t2)
relacional → relacional1 < relacional2	relacional.dir = nuevaTemporal relacional.tipo = max(relacional1.tipo , relacional2.tipo) t1= ampliar(relacional1.dir,relacional1.tipo, relacional.tipo) t2= ampliar(relacional2.dir, relacional2.tipo, relacional.tipo) relacional.code = gen(relacional.dir'=' t1'<'t2)
relacional → relacional1 <= relacional2	relacional.dir = nuevaTemporal

	relacional.tipo = max(relacional1.tipo , relacional2.tipo) t1= ampliar(relacional1.dir,relacional1.tipo, relacional.tipo) t2= ampliar(relacional2.dir, relacional2.tipo, relacional.tipo) relacional.code = gen(relacional.dir'=' t1'<='t2)
relacional → relacional1 >= relacional2	relacional.dir = nuevaTemporal relacional.tipo = max(relacional1.tipo , relacional2.tipo) t1= ampliar(relacional1.dir,relacional1.tipo, relacional.tipo) t2= ampliar(relacional2.dir, relacional2.tipo, relacional.tipo) relacional.code = gen(relacional.dir'=' t1'>='t2)
relacional → relacional1 < > relacional2	relacional.dir = nuevaTemporal relacional.tipo = max(relacional1.tipo , relacional2.tipo) t1= ampliar(relacional1.dir,relacional1.tipo, relacional.tipo) t2= ampliar(relacional2.dir, relacional2.tipo, relacional.tipo) relacional.code = gen(relacional.dir'=' t1'<>'t2)
relacional → relacional1 = relacional2	relacional.dir = nuevaTemporal relacional.tipo = max(relacional1.tipo , relacional2.tipo) t1= ampliar(relacional1.dir,relacional1.tipo, relacional.tipo) t2= ampliar(relacional2.dir, relacional2.tipo, relacional.tipo) relacional.code = gen(relacional.dir'=' t1'='t2)
relacional → expresion	relacional.dir = expresion.dir relacional.code = expresion.code
expresion → expresion1 + expresion2	expresion.dir = nuevaTemporal expresion.tipo = max(expresion1.tipo , expresion2.tipo) t1= ampliar(expresion1.dir, expresion1.tipo, expresion.tipo) t2= ampliar(expresion2.dir, expresion2.tipo, expresion.tipo) expresion.code = gen(expresion.dir'=' t1'+'t2)
expresion → expresion1 - expresion2	expresion.dir = nuevaTemporal expresion.tipo = max(expresion1.tipo , expresion2.tipo) t1= ampliar(expresion1.dir, expresion1.tipo, expresion.tipo) t2= ampliar(expresion2.dir, expresion2.tipo, expresion.tipo) expresion.code = gen(expresion.dir'=' t1'-'t2)
expresion → expresion1 * expresion2	expresion.dir = nuevaTemporal expresion.tipo = max(expresion1.tipo , expresion2.tipo) t1= ampliar(expresion1.dir, expresion1.tipo, expresion.tipo)

	t2= ampliar(expresion2.dir, expresion2.tipo, expresion.tipo) expresion.code = gen(expresion.dir'=' t1'*'t2)
expresion → expresion1 / expresion2	expresion.dir = nuevaTemporal expresion.tipo = max(expresion1.tipo , expresion2.tipo) t1= ampliar(expresion1.dir, expresion1.tipo, expresion.tipo) t2= ampliar(expresion2.dir, expresion2.tipo, expresion.tipo) expresion.code = gen(expresion.dir'=' t1'/'t2)
expresion → expresion1 % expresion2	expresion.dir = nuevaTemporal expresion.tipo = max(expresion1.tipo , expresion2.tipo) t1= ampliar(expresion1.dir, expresion1.tipo, expresion.tipo) t2= ampliar(expresion2.dir, expresion2.tipo, expresion.tipo) expresion.code = gen(expresion.dir'=' t1'%'t2)
expresion → (expresion1)	expresion.dir = $expresion_1$ .dir expresion.tipo = $expresion_1$ .tipo
expresion → variable	Si TS.existe(variable) Entonces expresion.dir = variable.dir expresion.tipo = TS.getTipo(variable) Sino error("La variable no ha sido declarada") Fin Si
expresion → num	expresion.tipo = num.tipo expresion.dir = num.val
expresion → cadena	expresion.tipo = cadena expresion.dir = TablaDeCadenas.add(cadena)
expresion → caracter	expresion.tipo = caracter expresion.dir = TablaDeCadenas.add(caracter)
variable → id variable_comp	**Si** TS.existe(id) Entonces tipo_id = TS.getTipo(id) t = reducir(variable_comp.dir,variable_comp.tipo,tipo_id) **Sino** error("El id no ha sido declarado") **FinSi**
variable_comp → dato_est_sim	variable_comp.dir = dato_est_sim.dir variable.code = dato_est_sim.code
variable → arreglo	variable.dir = arreglo.dir variable.base = arreglo.base variable.tipo = arreglo.tipo
variable → (parametros)	variable.lista = parametros.lista

	variable .num = parametros .num
dato_est_sim → dato_est_sim.id	**Si !**TS.existe(id) Entonces STS.getFondo().append(id,dir,tipo) dir = dir + STT.getFondo().getTam(tipo) **Sino** error("El id no ha sido declarado") **FinSi**
dato_est_sim → ε	
arreglo → [expresion]	t = nuevaTemporal() arreglo.dir = nuevaTemporal() arreglo.tipo = array arreglo.tam = TT.getTam(expresion.tipo) arreglo.base = expresion.base arreglo.code =gen(t'=' expresion.dir '*' arreglo.tam)
arreglo → $arreglo_1$ [expresion]	**Si** TT.getNombre(arreglo1.tipo) = array Entonces t = nuevaTemporal() arreglo.dir = nuevaTemporal() arreglo.tipo = TT.getTipoBase($arreglo_1$ .tipo) arreglo.tam = TT.getTam(arreglo.tipo) arreglo.base = $arreglo_1$ .base arreglo.code =gen(t'=' expresion.dir '*' arreglo.tam)|| gen(arreglo.dir'=' $arreglo_1$ .dir '+' t) **Sino** error("La variable asociada no es un arreglo") **Fin Si**
parametros → lista_param	parametros .lista = lista_param .lista parametros .num = lista_param .num
parametros → ε	parametros.lista = nulo parametros.num =0
lista_param → $lista\_param_1$, expresion	lista_param .lista = $lista\_param_1$ .lista lista_param .lista.append(expresion.tipo) lista_param .num = $lista\_param_1$ .num +1
lista_param → expresion	lista_param .lista = nuevaLista() lista_param .lista.append(expresion.tipo) lista_param .num = 1