

Universidad Nacional Autónoma de México Facultad de Ciencias Compiladores Gramática



Profesor: Adrián Ulises Mercado Martínez Ayudante: Karla Adriana Ezquivel Guzman Ayudante: Carlos Gerado Hernández Acosta

Gramática Final

- 1. programa \rightarrow declaraciones $\setminus n$ funciones
- 2. declaraciones \to tipo lista_var \n declaraciones | tipo_registro lista_var \n declaraciones | ε
- 3. tipo_registro \rightarrow registro $\setminus n$ inicio declaraciones $\setminus n$ fin
- 4. tipo \rightarrow base tipo_arreglo
- 5. base \rightarrow ent | real | dreal | car | sin
- 6. tipo_arreglo \rightarrow [num] tipo_arreglo | ε
- 7. lista_var \rightarrow lista_var , id | id
- 8. funciones \rightarrow func tipo id(argumentos) inicio $\setminus n$ declaraciones sentencias $\setminus n$ fin $\setminus n$ funciones $\mid \varepsilon$
- 9. argumentos \rightarrow listar_arg | sin
- 10. lista_arg \rightarrow lista_arg arg | arg
- 11. $arg \rightarrow tipo_arg id$
- 12. tipo_arg \rightarrow base param_arr
- 13. param_arr \rightarrow [] param_arr $\mid \varepsilon$
- 14. sentencias \rightarrow sentencias $\backslash n$ sentencia | sentencia
- 15. sentencia → si expresion_booleana entonces \n sentencias \n fin | si expresion_booleana \n sentencias \n sino \n sentencias \n fin | mientras \n expresion_booleana hacer \n sentencias \n fin | hacer \n sentencia \n mientras que expresion_booleana | id := expresion | escribir expresion | leer variable | devolver | devolver expresion | terminar
- 16. expresion_booleana → expresion_booleana oo expresion_booleana
 | expresion_booleana yy expresion_booleana
 | no expresion_booleana
 | relacional | verdadero | falso
- 17. relacional → relacional < relacional | relacional > relacional | relacional <= relacional | relacional |
- 18. expresion → expresion + expresion | expresion expresion
 | expresion * expresion | expresion / expresion
 | expresion % expresion | (expresion)
 | variable | num | cadena | caracter | id(parametros)
- 19. variable \rightarrow id arreglo | id.id
- 20. arreglo \rightarrow id [expression] arreglo | ε
- 21. parametros \rightarrow lista_param | ε
- 22. lista_param \rightarrow lista_param , expresion | expresion

PRODUCCIÓN	REGLAS SEMÁNTICAS
ho programa $ ightarrow$ declaraciones $ ho$ funciones	$ \operatorname{dir}=0 $
programa \rightarrow declaraciones $\backslash n$ funciones	StackTT = newStackTT()
	StackTS = newStackTS()
	ts = newSymTab()
	ts = newSymTab() $tt = newTypeTab()$
	StackTT.push(tt)
	StackT1.push(tt) StackTS.push(ts)
	TablaDeCadenas = newTablaCadenas()
$\text{declaraciones} \rightarrow \text{tipo lista_var } \setminus n \text{ declaraciones}$	type = tipo.tipo
declaraciones \rightarrow tipo insta-var \n declaraciones declaraciones	type = tipo.tipo type = tipo_registro.tipo
declaraciones $\rightarrow \epsilon$	type — tipo_registro.tipo
tipo_registro \rightarrow registro $\setminus n$	ts = newSymTab()
inicio declaraciones $\setminus n$ fin	tt = newTypeTab()
micio deciaraciones (n mi	StackDir.push(dir)
	dir = 0
	$\frac{dH}{dt} = 0$ StackTT.push(tt)
	StackT1.push(tt) StackTS.push(ts)
	dir = StackDir.pop()
	tt1 = StackDir.pop() $tt1 = StackTT.pop()$
	StackTS.getCima().setTT(tt1)
	ts1 = StackTS.pop()
	dir = Stack Dir.pop()
	type = StackTT.getCima().addTipo("registro",0, ts1)
$tipo \rightarrow base tipo_arreglo$	base = base.tipo
tipo - base tipo_arregio	tipo.tipo = tipo_arreglo.tipo
$\mathrm{base} o \mathbf{ent}$	base.tipo = ent
$base \rightarrow \mathbf{real}$	base.tipo = real
$base \rightarrow \mathbf{dreal}$	base.tipo = Ireal
base \rightarrow car	base.tipo = car
$base \rightarrow sin$	base.tipo = car base.tipo = sin
$\begin{array}{c} \text{base} \rightarrow \text{sin} \\ \text{tipo_arreglo} \rightarrow [\text{num}] \text{ tipo_arreglo}_1 \end{array}$	
tipo_arregio \(\) [num] tipo_arregio1	$\mathbf{si} \ num.tipo = ent \ \mathbf{y} \ num.val > 0 \ \mathbf{entonces}$
	tipo_arreglo.tipo = StackTT.getCima().addTipo("array",
	num.val, tipo_arreglo ₁ .tipo)
	en otro caso
	Error("El índice tiene que ser entero y mayor que cero")
tipo_arreglo $\rightarrow \varepsilon$	fin tipo_arreglo.tipo = base
lista_var \rightarrow lista_var , id	+
nouse / nouse / no	si StackTS.getCima().getId(id.lexval) = -1 entonces
	StackTS.getCima().addSym(id.lexval, tipo, dir, "var")
	dir = dir + StackTT.getCima().getTam(tipo)
	en otro caso France ("Fl. identificador ya fue declarado")
	Error("El identificador ya fue declarado") fin
$lista_var \rightarrow id$	
inducation / ICI	si StackTS.getCima().getId(id.lexval) = -1 entonces
	en otro caso Error("El identificador ya fue declarado")
	fin
	1111

PRODUCCIÓN	REGLAS SEMÁNTICAS
funciones $ o$ func tipo id(argumentos) inicio \n declaraciones sentencias \n fin \n funciones	$ \begin{array}{c} \mathbf{si} \; StackTS.getFondo().getId(id.lexval) \neq -1 \; \mathbf{entonces} \\ \; \; \; StackTS.getFondo().addSym(id.lexval,tipo,,"func") \\ \; \; StackDir.push(dir) \\ \; \; \; FuncType = tipo.tipo \\ \; \; \; FuncReturn = false \\ \; \; dir = 0 \\ \; \; \; StackTT.push(tt) \\ \; \; \; \; StackTS.push(ts) \\ \; \; dir = StackDir.pop() \\ \; \; \; \; add_quad(code,'label',-,-,id.lexval) \\ \; \; L = newLabel() \\ \; \; backpatch(code,sentencias.next,L) \\ \; \; \; add_quad(code,'label',-,-,L) \\ \; \; \; \; \; StackTT.pop() \\ \; \; \; \; \; \; \; \; \; \; \; \; \; \; \; \; \; \; $
funciones $\rightarrow \varepsilon$	
$argumentos \rightarrow listar_arg$	argumentos.lista = lista_arg.lista
$argumentos \rightarrow sin$	argumentos.lista = nulo
$lista_arg \rightarrow lista_arg arg_1$	lista_arg.lista = lista_arg_1.lista
1. /	lista_arg.lista.add(arg.tipo)
$lista_arg \rightarrow arg$	lista_arg.lista = newListaParam() lista_arg.lista.add(arg.tipo)
$arg \rightarrow tipo_arg id$	si StackTS.getCima().getId(id.lexval) = -1 entonces StackTS.getCima().addSym(id.lexval, tipo, dir, "var") dir = dir + StackTT.getCima().getTam(tipo) en otro caso Error("El identificador ya fue declarado") fin arg.tipo = tipo_arg.tipo
$tipo_arg \rightarrow base param_arr$	base = base.tipo tipo_arg.tipo = param_arr.tipo
$param_arr \rightarrow [] param_arr_1$	param_arr.tipo = StackTT.getCima().addTipo("array", -, param_arr ₁ .tipo)
$param_arr \to \varepsilon$	param_arr.tipo = base
sentencias \rightarrow sentencias $\backslash n$ sentencia	L = newLabel() backpatch(code, sentecias.listnext, L) sentencias.listnext = sentencia.listnext
$sentencias \rightarrow sentencia$	sentencias.listnext = sentencia.listnext
sentencia $ ightarrow$ si expresion_booleana entonces $ angle n$ sentencias $ angle n$ fin	L = newLabel() backpatch(code, expresion_booleana.listtrue, L) sentencia.listnext = combinar(expresion_booleana.listfalse, sentencias.listnext)
sentencia $ o$ si expresion_booleana $\ n$ sentencias $_1 \ n$ sino $\ n$ sentencias $_2 \ n$ fin	L = newLabel() L1 = newLabel() backpatch(code, expresion_boolean.listtrue, L) backpatch(code, expresion_boolean.listfalse, L1) sentencia.listnext = combinar(sentencias ₁ .listnext, sentencias ₂ .listnext)

L = newLabel() backpatch(code, expresion boolean listtrue, L) backpatch(code, expresion boolean listtrue, L) backpatch(code, sentencias listnext, L1) sentencia. Istnext = expresion booleana. Istifake add.quad(code, 'lbebe', ', -, 'lbebe', ', -, 'l' + 'd)	PRODUCCIÓN	REGLAS SEMÁNTICAS
\(\frac{\text{N sentencius \n fin} \) L1 = mewl.abel() backpatch(code, sentencius listmext, L) backpatch(code, sepresion, boolean, listfuse L1) sentencia stinct = expresion boolean listfuse add.quad(code, "goto", -, -, L) mientrus que expresion booleana L1 = mewl.abel() sentencia stinct = expresion booleana L2 = mewl.abel() sentencia stinct = captesion sentencia stinct = captesion sentencia stinct = captesion stinct stinct = captesion stinct = captesi	$sentencia \rightarrow mientras \setminus n$ expresion_booleana hacer	L = newLabel()
backpatch(code, sentencia.listnext, L) backpatch(code, expression.boolean.listnext L1) sentencia.listnext = expression boolean.listnext L1) sentencia.listnext = expression boolean.listnext L2) backpatch(code, sentencia.listnext, L1) backpatch(code, sentencia.listnext, L1) backpatch(code, sentencia.listnext, L1) backpatch(code, sentencia.listnext, L1) sentencia.listnext = expression boolean.listlase add.quad(code, lnbc*r, -, L) sentencia.listnext = servesion.boolean.listlase add.quad(code, lnbc*r, -, L) sentencia.listnext = servesion.dir, expression.tip, t) add.puad(code, servesion.dir, expression.tip, t) add.puad(code, servesion.dir, expression.tip, t) add.puad(code, servesion.dir, expression.tip, t) sentencia → variable := expresion sentencia → servibir expresion sentencia → servipir expresion.booleana, listiture, exp	• =	
sentencia → hacer \n sentenciaso \n mientras que expresion.booleana listifalse add.quad(code, "goto", ¬, ¬, L) L = newLabel() backpatch(code, expresion.booleana.listifalse add.quad(code, "abet", ¬, ¬, L) sentencia → id := expresion sentencia → id = expresion.id = expresion.inp, variable.inp, oad pada((ode, "a = a, "a, -, variable.base[variable.dir]) sentencia → id = expresion.id = ex	, , ,	
add.quad(code, "goto", -, -, L) L = newLabel() backpatch(code, expresion.boolean listtrue, L) backpatch(code, expresion.boolean.listtrue, L) backpatch(code, expresion.boolean.listtrue, L) backpatch(code, expresion.booleana, listfabe add.quad(code, "Boto", -, -, L) sentencia → id := expresion sentencia → stackTS.getCimad).getTip(id.lexeval) d = StackTS.getCimad).getTip(id.lexeval) d = StackTS.getCimad).getTip(id.lexeval) d = StackTS.getCimad).getTip(id.lexeval) d = reducir(expresion.tipo,) getTipo(id.lexeval) d = reducir(expresion.tipo, yariable.tipo) add.quad(code, " = ", α, -, variable.tipo) add.quad(code, " = ", α, -, variable.base[variable.dir]) sentencia → sentencia → terrinable sentencia → leer variable sentencia → leer variable sentencia → devolver sentencia → devolver sentencia → devolver sentencia → terrina función debe retornar algún valor de tipo " in FuncType in FuncType in FuncType in FuncType in in		backpatch(code, expresion_boolean.listtrue, L1)
L = newLabel() backpatch(code, expresion_boolean listtrue, L) backpatch(code, expresion_boolean listtrue, L) backpatch(code, expresion_boolean listtrue, L) backpatch(code, expresion_boolean listtrue) backpatch(code, expresion_boolean listtake add.quad(code, "label", -, -, L)		sentencia.listnext = expresion_booleana.listfalse
L = new Label() backpatch(code, expresion boolean listtrue, L) backpatch(code, sentencias listnext, L1) sentencia listnext = expresion booleanalistfabe add,quad(code, "label", ", ", L1) sentencia listnext = expresion booleanalistfabe add,quad(code, "label", ", ", ", ", ", ", ", ", ", ", ", ", "		add_quad(code, "goto", -, -, L)
backpatch(code, expresion booleau.listrue, L) backpatch(code, sentencias listnext, L1) sentencia → id := expresion sentencia → id := expresion id := id id expresion id id expresion id expresion id expresion id expresion	$\operatorname{sentencia} o \operatorname{\mathbf{hacer}} \setminus n \operatorname{sentencias}_2 \setminus n$	
$\begin{array}{c} \text{backpatch}(\text{code, sentencia, listnext, }L1) \\ \text{sentencia, listnext} = \exp(\text{resion, booleana, listfalse} \\ \text{add, quad}(\text{code, "label",, L}) \\ \\ \text{sentencia} \rightarrow \text{id} := \exp(\text{resion}) \\ \\ \begin{array}{c} \text{si } Stack TS.getCma(l).gettl/got(al.exal)} \\ d = Stack TS.getCma(l).gettl/got(al.exal) \\ d = Stack TS.getCma(l).gettl/got(al.exal) \\ d = Control Case \\ l = Control Ca$	mientras que expresion_booleana	L = newLabel()
$\begin{array}{c} \text{si } StackTS.getCima().getId(id.lexval) \neq -1 \text{ entonces} \\ t = StackTS.getCima().getIlq(id.lexval) \\ d = StackTS.getCima().getIlp(id.lexval) \\ a = reducir(expresion.dir, expresion.tipo, t) \\ add apuad(code, "= ", \alpha, ", "Id" + d) \\ en otro caso \\ Error("El identificador no ha sido declarado") \\ fin \\ sentencia \rightarrow variable := expresion \\ a = reducir(expresion.dir, expresion.tipo, variable.tipo) \\ add.quad(code, "= ", a, -, variable.base[variable.dir]) \\ sentencia \rightarrow sertencia.listnext= nulo \\ sentencia \rightarrow sertencia.listnext= nulo \\ sentencia \rightarrow learn variable \\ sentencia \rightarrow sertencia.listnext= nulo \\ sentencia \rightarrow learn variable \\ sentencia \rightarrow devolver \\ \hline si Finant flype = sin entonces \\ add.quad(code, "sean", -, -, variable.dir) \\ sentencia \rightarrow devolver \\ \hline sentencia \rightarrow devolver \\ \hline si Finant flype = sin entonces \\ add.quad(code, "return", -, -, -) \\ en otro caso \\ Error("La función debe retornar algún valor de tipo " + FuncType) \\ add.quad(code, "return", expresion.dir, -, -) \\ FuncType = sin entonces \\ a = reducir(expresion.dir, expresion.dir, -, -) \\ FuncType = sin entonces \\ a = reducir(expresion.dir, expresion.dir, -, -) \\ FuncType = sin entonces \\ a = reducir(expresion.dir, -, -) \\ FuncType = sin entonces \\ a = reducir(expresion.dir, -, -) \\ FuncType = sin entonces \\ a = reducir(expresion.dir, -, -) \\ FuncType = sin entonces \\ a = reducir(expresion.dir, -, -) \\ FuncType = sin entonces \\ a = reducir(expresion.dir, -, -, -) \\ a = reducir(expresion.dir, -, -, -, -) \\ a = reducir(expresion.dir, -, -, -, -, -, -, -, -, -, -, -, -, -,$		
$ \begin{vmatrix} t = StackTS.getCima().getDipo(id.lexval) \\ d = StackTS.getCima().getDip(id.lexval) \\ a = reducir(expresion.dir, expresion.tipo, t) \\ add.quad(code, " = ", \alpha, -, "Id" + d) \\ en otro caso Error("El identificador no ha sido declarado") \\ fin \\ sentecia.listnext= nulo \\ sentencia \rightarrow variable := expresion \alpha = reducir(expresion.dir, expresion.tipo, variable.tipo) \\ add.quad(code, " = ", \alpha, -, variable.base[variable.dir]) \\ sentecia.listnext= nulo \\ sentencia \rightarrow escribir expresion add.quad(code, " = ", \alpha, -, variable.base[variable.dir]) \\ sentencia \rightarrow leer variable add.quad(code, " = "in", expresion.dir, -, -) \\ sentencia \rightarrow leer variable add.quad(code, " = "in", -, -, variable.dir) \\ sentencia \rightarrow devolver $		add_quad(code, "label", -, -, L)
$ \begin{vmatrix} d = StackTS.getCima(),getDir(id.lexval) \\ \alpha = reducir(expresion.dir, expresion.tipo, t) \\ add.quad(code," = ", \alpha, -, "ld" + d) \\ en otro caso \\ Error("El identificador no ha sido declarado") \\ fin \\ sentecia.listnext= nulo $	$sentencia \rightarrow id := expression$	$\mathbf{si}\ StackTS.getCima().getId(id.lexval) \neq -1\ \mathbf{entonces}$
$ \begin{array}{c} \alpha = reducir(expresion.dir, expression.tipo, t) \\ add.quad(code, " = ", \alpha, -, "Id" + d) \\ en otro caso \\ Error("El identificador no ha sido declarado") \\ fin \\ sentecia.listnext= nulo \\ sentencia \rightarrow variable := expresion $		
$\begin{array}{c} \begin{array}{c} \textbf{en otro caso} \\ & \textbf{Error}^{\text{re}} \textbf{Ei dentificador no ha sido declarado"}) \\ \textbf{fin} \\ \textbf{sentecia.} \textbf{listnext=} \textbf{nulo} \\ \\ \textbf{sentencia} \rightarrow \textbf{variable} := \text{expresion} \\ \textbf{ac} = \sigma \text{erducir}(\text{expresion.dir, expresion.tipo, variable.tipo}) \\ add_quad(code, "= ", a, -, variable.base[variable.dir]) \\ \textbf{sentecia.} \textbf{listnext=} \textbf{nulo} \\ \textbf{sentencia} \rightarrow \textbf{escribir} \textbf{expresion} \\ \textbf{add.} \textbf{quad}(code, "print", expresion.dir, -, -) \\ \textbf{sentecia.} \textbf{listnext=} \textbf{nulo} \\ \textbf{sentencia} \rightarrow \textbf{devolver} \\ \textbf{sine} \textbf{Func} \textbf{Type} = sin \textbf{entonces} \\ \textbf{add.} \textbf{quad}(code, "scan", -, -, variable.dir) \\ \textbf{sentecia.} \textbf{listnext=} \textbf{nulo} \\ \textbf{sentencia} \rightarrow \textbf{devolver} \\ \textbf{sine} \textbf{Func} \textbf{Type} = sin \textbf{entonces} \\ \textbf{add.} \textbf{quad}(code, "return", -, -, -) \\ \textbf{en otro caso} \\ \textbf{Error''La función debe retornar algún valor de tipo " + + Func Type fin sentencia. listnext= nulo \\ \textbf{sentencia} \rightarrow \textbf{devolver} \textbf{expresion} \\ \textbf{sentencia} \textbf{listnext=} \textbf{nulo} \\ sent$		
$\begin{array}{c} \mid & \operatorname{Error}("\operatorname{El} \operatorname{identificador} \operatorname{no} \operatorname{ha} \operatorname{sido} \operatorname{declarado"}) \\ & \operatorname{fin} \\ & \operatorname{sentecia.} \operatorname{listnext=nulo} \\ & \operatorname{sentecia.} \operatorname{listnext=nulo} \\ & \operatorname{ac} = \operatorname{reducir}(\operatorname{expresion.tip, expresion.tip, variable.tipo}) \\ & \operatorname{add.quad}(\operatorname{code, "print", expresion.tip, -p.} \\ & \operatorname{sentecia.} \operatorname{listnext=nulo} \\ & \operatorname{sentecia.} \operatorname{listnext=nulo} \\ & \operatorname{sentencia} \to \operatorname{escribir} \operatorname{expresion} \\ & \operatorname{add.quad}(\operatorname{code, "scan", -, -, variable.base}[\operatorname{variable.dir}]) \\ & \operatorname{sentencia} \to \operatorname{leer} \operatorname{variable} \\ & \operatorname{sentecia.} \operatorname{listnext=nulo} \\ & \operatorname{sentencia} \to \operatorname{devolver} \\ & \operatorname{sentecia.} \operatorname{listnext=nulo} \\ & \operatorname{sentencia} \to \operatorname{devolver} \operatorname{expresion} \\ & \operatorname{devolver} \\ & \operatorname{devolver} \\ & \operatorname{devolver} \operatorname{expresion} \\ & \operatorname{devolver} \\ & de$		
$ \begin{array}{c} \textbf{fin} \\ \textbf{sentecia.} \\ \textbf{istnext= nulo} \\ \textbf{sentencia} \rightarrow \textbf{variable} := \textbf{expresion} \\ \textbf{ad} & - \textbf{reducir}(\textbf{expresion.dir}, \textbf{expresion.tipo}, \textbf{variable.tipo}) \\ \textbf{add.} quad(code," = ", \alpha, -, variable.base[variable.dir]) \\ \textbf{sentecia.} \\ \textbf{istnext= nulo} \\ \textbf{sentencia} \rightarrow \textbf{escribir} \textbf{expresion} \\ \textbf{add.} \\ \textbf{quad}(\textbf{code}, "print", \textbf{expresion.dir}, -, -) \\ \textbf{sentencia} \rightarrow \textbf{leer} \textbf{ variable} \\ \textbf{sentencia} \rightarrow \textbf{leer} \textbf{ variable} \\ \textbf{sentencia} \rightarrow \textbf{devolver} \textbf{ expresion} \\ \textbf{sentencia.} \textbf{ istnext= nulo} \\ \textbf{sentencia} \rightarrow \textbf{devolver} \textbf{ expresion.} \\ \textbf{sentencia.} \textbf{ istnext= nulo} \\ \textbf{sentencia.} \rightarrow \textbf{devolver} \textbf{ expresion.} \\ devolve$		
$sentencia \rightarrow variable := expresion $		
$ \begin{array}{c} \operatorname{sentencia} \to \operatorname{variable} := \operatorname{expresion} & \alpha = \operatorname{reducir}(\operatorname{expresion.dir}, \operatorname{expresion.tipo}, \operatorname{variable.tipo}) \\ add.quad(\operatorname{code}, "=", \alpha, -, \operatorname{variable.base}[\operatorname{variable.dir}]) \\ \operatorname{sentencia} \to \operatorname{escribir} \operatorname{expresion} & \operatorname{add.quad}(\operatorname{code}, "\operatorname{print"}, \operatorname{expresion.dir}, -, -) \\ \operatorname{sentencia} \to \operatorname{leer} \operatorname{variable} & \operatorname{add.quad}(\operatorname{code}, "\operatorname{scan"}, -, \operatorname{variable.dir}) \\ \operatorname{sentencia} \to \operatorname{devolver} & \operatorname{sintencia} \to \operatorname{devolver} & \operatorname{sintencia} \to \operatorname{devolver} \\ \operatorname{sentencia} & \operatorname{devolver} & \operatorname{sintencia} & \operatorname{devolver} \\ \operatorname{sentencia} & \operatorname{devolver} & \operatorname{cyresion} & \operatorname{sintencia} & \operatorname{devolver} & \operatorname{cyresion} \\ \operatorname{sentencia} & \operatorname{devolver} & \operatorname{cyresion} & \operatorname{sintencia} & \operatorname{devolver} & \operatorname{cyresion} \\ \operatorname{sentencia} & \operatorname{devolver} & \operatorname{cyresion} & \operatorname{sintencia} & \operatorname{sintencia} & \operatorname{sintencia} & \operatorname{devolver} & \operatorname{cyresion} \\ \operatorname{sentencia} & \operatorname{devolver} & \operatorname{cyresion} & \operatorname{sintencia} & $		
	contancia / variable :- expresion	
sentencia → escribir expresion sentencia → escribir expresion sentencia → leer variable sentencia → leer variable sentencia → devolver sentencia → devolver expresion sentencia → terminar sentencia → terminar I = newIndex() add.quad(code, "return", expresion.dir, -, -) Func Return = true en otro caso Error("La función no puede retornar algún valor de tipo ") fin sentencia → terminar I = newIndex() add.quad(code, "goto", -, -, 1) sentencia.listnext = nuel.list() sentencia.listnext. = newList() sentencia.listnext. = newList() sentencia.listnext. = combinar(expresion.booleana ₁ .listfalse, L) expresion.booleana ₂ .listfrue	sentencia → variable :— expresion	
sentencia → escribir expresion add_quad(code, "print", expresion.dir, -,-) sentencia.listnext= nulo add_quad(code, "scan", -, -, variable.dir) sentencia → devolver si FuncType = sin entonces add_quad(code, "return", -, -, -) en otro caso Error("La función debe retornar algún valor de tipo " + FuncType) fin sentencia.listnext= nulo sentencia → devolver expresion sentencia → devolver expresion sentencia → devolver expresion sentencia → terminar sentencia → terminar Si FuncType ≠ sin entonces α = reducir(expresion.dir, expresion.tipo, FuncType) add_quad(code, "return", expresion.dir, -, -) FuncReturn = true en otro caso Error("La función no puede retornar algún valor de tipo ") fin sentencia.listnext= nulo sentencia → terminar I = newIndex() add_quad(code, "goto", -, -, I) sentencia.listnext= newList() sentencia.listnext= newList() sentencia.listnext= newList() sentencia.listnext= combinar(expresion booleana₁ listfalse, L) expresion.booleana.listfurue = combinar(expresion booleana₁.listfurue, expresion.booleana.listfalse= expresion.booleana.listfalse		
$sentencia \rightarrow leer \ variable$ $sentencia - listnext = nulo$ $sentencia - listnext = leer \ leer \ variable \cdot lie = leer \ $	sentencia \rightarrow escribir expresion	
$\begin{array}{c} \text{sentencia} \rightarrow \text{devolver} \\ \text{si } Func Type = sin \text{ entonces} \\ add_u ad(code, "return", -, -, -) \\ add_u ad(code, "return", -, -, -) \\ en \text{ otro } \mathbf{caso} \\ Error("La función debe retornar algún valor de tipo") \\ + Func Type \\ \mathbf{fin} \\ \text{sentencia} \rightarrow \mathbf{devolver} \text{ expresion} \\ \text{sentencia} \rightarrow \mathbf{devolver} \text{ expresion} \\ \text{sentencia} \rightarrow \mathbf{devolver} \text{ expresion} \\ \text{sentencia} \rightarrow \mathbf{devolver} \text{ expresion} \\ \text{sentencia} \rightarrow \mathbf{devolver} \text{ expresion} \\ \text{sentencia} \rightarrow \mathbf{devolver} \text{ expresion} \\ \text{sentencia} \rightarrow \mathbf{devolver} \text{ expresion} \\ \text{sentencia} \rightarrow \mathbf{devolver} \text{ expresion} \\ \text{sentencia} \rightarrow \mathbf{devolver} \text{ expresion} \\ \text{sentencia} \rightarrow \mathbf{devolver} \text{ expresion} \\ \text{sentencia} \rightarrow \mathbf{devolver} \text{ expresion.dir}, exp$	sentencia \rightarrow leer variable	
$ \begin{vmatrix} add.quad(code, "return", -, -, -) \\ en otro caso \\ Error("La función debe retornar algún valor de tipo " \\ + FuncType) \\ fin \\ sentencia . listnext= nulo $ $ \begin{aligned} si & \mathit{FuncType} \neq sin \text{ entonces} \\ & \alpha = reducir(expresion.dir, expresion.tipo, \\ & \mathit{FuncType} \\ & add.quad(code, "return", expresion.dir, -, -) \\ & \mathit{FuncReturn} = true \\ & en otro caso \\ & Error("La función no puede retornar algún valor de tipo ") \\ & fin \\ & sentencia . listnext= nulo \end{aligned} $ $ \begin{aligned} & I = \text{newIndex}() \\ & \text{add.quad}(code, "goto", -, -, I) \\ & \text{sentencia. listnext} = \text{newList}() \\ & \text$		
	$sentencia \rightarrow devolver$	$\mathbf{si} \; FuncType = sin \; \mathbf{entonces}$
$ \begin{array}{c} & \operatorname{Error}(\text{``La función debe retornar algún valor de tipo ''} \\ & + \operatorname{FuncType}) \\ & \operatorname{fin} \\ & \operatorname{sentencia.listnext= nulo} \\ \\ & \operatorname{sentencia} \to \operatorname{devolver} \operatorname{expresion} \\ & \operatorname{si } \operatorname{FuncType} \neq \operatorname{sin } \operatorname{entonces} \\ & \alpha = \operatorname{reducir}(\operatorname{expresion.dir}, \operatorname{expresion.tipo}, \\ & \operatorname{FuncType}) \\ & \operatorname{add.quad}(\operatorname{code}, "\operatorname{return"}, \operatorname{expresion.dir}, -, -) \\ & \operatorname{FuncReturn} = \operatorname{true} \\ & \operatorname{en otro } \operatorname{caso} \\ & \operatorname{Error}(\text{``La función no puede retornar algún valor de tipo ''}) \\ & \operatorname{fin} \\ & \operatorname{sentencia.listnext= nulo} \\ \\ \operatorname{sentencia.listnext= nulo} \\ & \operatorname{I = newIndex}() \\ & \operatorname{add.quad}(\operatorname{code}, "\operatorname{goto"}, -, -, I) \\ & \operatorname{sentencia.listnext= newList}() \\ & \operatorname{sentencia.listnext.add}(I) \\ \\ \operatorname{expresion.booleana} \to \operatorname{expresion.booleana_1} \operatorname{oo} \\ & \operatorname{expresion.booleana_1.listfalse}, L) \\ & \operatorname{expresion.booleana_2.listfrue} \\ & \operatorname{expresion.booleana_2.listfrue}) \\ & \operatorname{expresion.booleana_2.listfrue}) \\ & \operatorname{expresion.booleana_2.listfrue} \\ & \operatorname{expresion.booleana_2.listfalse} \\ \end{array}$		$ add_quad(code, "return", -, -, -) $
$ \begin{array}{c} & + \operatorname{FuncType} \rangle \\ & \operatorname{fin} \\ & \operatorname{sentencia.listnext= nulo} \\ \\ \operatorname{sentencia} \to \operatorname{\mathbf{devolver}} \operatorname{expresion} \\ & \begin{array}{c} \operatorname{\mathbf{si}} \operatorname{\mathit{FuncType}} \neq \operatorname{\mathit{sin}} \operatorname{\mathbf{entonces}} \\ & \alpha = \operatorname{\mathit{reducir}}(\operatorname{expresion.dir}, \operatorname{expresion.tipo}, \\ \operatorname{\mathit{FuncType}}) \\ & \operatorname{\mathit{add.quad}}(\operatorname{\mathit{code}}, \operatorname{\mathit{"return"}}, \operatorname{\mathit{expresion.dir}}, -, -) \\ & \operatorname{\mathit{FuncReturn}} = \operatorname{\mathit{true}} \\ & \operatorname{\mathbf{enotro}} \operatorname{\mathbf{caso}} \\ & \operatorname{\mathit{Error}}(\operatorname{``La} \operatorname{función} \operatorname{no} \operatorname{puede} \operatorname{\mathit{retornar}} \operatorname{algún} \operatorname{\mathit{valor}} \operatorname{\mathit{de}} \operatorname{\mathit{tipo}} \operatorname{\mathit{``logo}}) \\ & \operatorname{\mathbf{fin}} \\ & \operatorname{\mathit{sentencia.listnext}} = \operatorname{nulo} \\ \\ \operatorname{\mathit{sentencia.listnext}} = \operatorname{\mathit{nulo}} \\ \\ \operatorname{\mathit{sentencia.listnext}} \\ & \operatorname{\mathit{add.quad}}(\operatorname{\mathit{code}}, \operatorname{"goto"}, -, -, \operatorname{\mathit{I}}) \\ & \operatorname{sentencia.listnext} = \operatorname{\mathit{newList}}() \\ & \operatorname{\mathit{sentencia.listnext.add}}(\operatorname{\mathit{I}}) \\ \\ \operatorname{\mathit{expresion.booleana}} \\ & \operatorname{\mathit{expresion.booleana_1.listfalse}}, \operatorname{\mathit{L}}) \\ & \operatorname{\mathit{expresion.booleana_2.listfalse}} \\ & \operatorname{\mathit{expresion.booleana_2.listfurue}} \\ & \operatorname{\mathit{expresion.booleana_2.listfalse}} \\ \\ & \operatorname{\mathit{expresion.booleana_2.listfalse}} \\ \end{array} $		
$\begin{array}{c} \textbf{fin} \\ \textbf{sentencia} \rightarrow \textbf{devolver} \ \textbf{expresion} \\ \textbf{si} \ \textit{FuncType} \neq \textit{sin} \ \textbf{entonces} \\ & \alpha = \text{reducir}(\textbf{expresion.dir}, \textbf{expresion.tipo}, \\ \textbf{FuncType} \\ & \textit{add.quad}(\textit{code}, "\textit{return"}, \textit{expresion.dir}, -, -) \\ \textbf{FuncReturn} = \text{true} \\ & \textbf{en} \ \textbf{otro} \ \textbf{caso} \\ & \text{Error}("\text{La} \ \text{función} \ \text{no} \ \text{puede} \ \text{retornar} \ \text{algún} \ \text{valor} \ \text{de} \ \text{tipo} \ ") \\ \textbf{fin} \\ & \text{sentencia}. \text{listnext=} \ \text{nulo} \\ \\ \textbf{sentencia}. \text{listnext=} \ \text{nulo} \\ \\ \textbf{expresion.booleana} \rightarrow \textbf{expresion.booleana}_1 \ \textbf{oo} \\ & \text{expresion.booleana}_2 \\ & \text{expresion.booleana}_2 \\ & \text{expresion.booleana}_2. \text{listfalse}, \ \textbf{L}) \\ & \text{expresion.booleana}_2. \text{listfure} \\ & \text{expresion.booleana}_2. \text{listfalse} \\ & \text{expresion.booleana}_2. \text{listfalse} \\ & \text{expresion.booleana}_2. \text{listfalse} \\ & \text{expresion.booleana}_2. \text{listfalse} \\ \\ \\ & \text{expresion.booleana}_2. $,
$\begin{array}{c} \text{sentencia.listnext= nulo} \\ \\ \text{sentencia} \rightarrow \text{devolver} \; \text{expresion} \\ \\ \text{si } \textit{FuncType} \neq \textit{sin entonces} \\ \\ \text{ca} \; \text{reducir}(\text{expresion.dir, expresion.tipo}, \\ \text{FuncType}) \\ \text{add.quad}(\textit{code}, "\textit{return"}, \textit{expresion.dir}, -, -) \\ \text{FuncReturn} \; = \text{true} \\ \text{en otro } \textit{caso} \\ \\ \text{Error}(\text{"La función no puede retornar algún valor de tipo"}) \\ \text{fin} \\ \text{sentencia} \rightarrow \text{terminar} \\ \\ \text{I} \; = \text{newIndex}() \\ \text{add.quad}(\textit{code}, "\textit{goto"}, -, -, I) \\ \text{sentencia.listnext} \; = \text{newList}() \\ \text{sentencia.listnext} \; = \text{newList}() \\ \text{sentencia.listnext.add}(I) \\ \\ \text{expresion.booleana} \rightarrow \text{expresion.booleana}_1 \; \textbf{oo} \\ \text{expresion.booleana}_2 \\ \\ \text{expresion.booleana}_2. \text{listfalse}, \; L) \\ \text{expresion.booleana}_2. \text{listfurue} \\ \text{expresion.booleana}_2. \text{listfurue}, \\ \text{expresion.booleana}_2. \text{listfurue}, \\ \text{expresion.booleana}_2. \text{listfalse} \\ \end{array}$		1
$\begin{array}{c} \textbf{si } \textit{FuncType} \neq \textit{sin } \textbf{entonces} \\ & \alpha = \text{reducir}(\text{expresion.dir}, \text{expresion.tipo}, \\ & \text{FuncType}) \\ & \textit{add_quad}(\textit{code}, "\textit{return}", \textit{expresion.dir}, -, -) \\ & \text{FuncReturn} = \text{true} \\ & \textbf{en otro caso} \\ & \text{Error}(\text{"La función no puede retornar algún valor de tipo ")} \\ & \textbf{fin} \\ & \text{sentencia} \rightarrow \textbf{terminar} \\ & I = \text{newIndex}() \\ & \text{add_quad}(\text{code}, "\textit{goto"}, -, -, I) \\ & \text{sentencia.listnext} = \text{newList}() \\ & \text{sentencia.listnext.add}(I) \\ & \text{expresion_booleana}_2 \\ & \text{expresion_booleana}_2. \\ & \text{listfulse} \in \text{combinar}(\text{expresion_booleana}_1. \text{listfalse}, L) \\ & \text{expresion_booleana}_2. \text{listfulse} \\ &$		
$\begin{array}{c ccccccccccccccccccccccccccccccccccc$	gentencia develven evenegien	
$\begin{array}{c c} FuncType) & add_quad(code,"return",expresion.dir,-,-) \\ FuncReturn = true \\ \textbf{en otro caso} \\ & Error("La función no puede retornar algún valor de tipo") \\ \textbf{fin} \\ sentencia.listnext= nulo \\ \\ sentencia.listnext= nulo \\ \\ \hline sentencia.listnext= newList() \\ add_quad(code, "goto", -, -, I) \\ sentencia.listnext = newList() \\ sentencia.listnext = newList() \\ sentencia.listnext.add(I) \\ \hline expresion_booleana \rightarrow expresion_booleana_1 \ \textbf{oo} \\ expresion_booleana_2 \\ \hline \\ backpatch(code, expresion_booleana_1.listfalse, L) \\ expresion_booleana_2.listtrue \\ expresion_booleana_2.listtrue) \\ expresion_booleana_2.listfalse \\ \hline \end{array}$	sentencia \rightarrow devolver expresion	
		· - · · · · · · · · · · · · · · · · · ·
$\begin{array}{c c} & FuncReturn = true \\ \textbf{en otro caso} \\ & Error("La función no puede retornar algún valor de tipo") \\ \textbf{fin} \\ sentencia \rightarrow \textbf{terminar} \\ & I = newIndex() \\ add_quad(code, "goto", -, -, I) \\ sentencia.listnext = newList() \\ sentencia.listnext = newList() \\ sentencia.listnext.add(I) \\ \\ expresion_booleana \rightarrow expresion_booleana_1 \ \textbf{oo} \\ expresion_booleana_2 \\ & backpatch(code, expresion_booleana_1.listfalse, L) \\ expresion_booleana_2.listtrue = combinar(expresion_booleana_1.listtrue, \\ expresion_booleana_2.listtrue) \\ expresion_booleana_2.listfalse \\ & expresion_booleana_2.listfalse \\ \end{array}$		
$\begin{array}{c} \textbf{en otro caso} \\ \ \text{Error}(\text{"La función no puede retornar algún valor de tipo"}) \\ \textbf{fin} \\ \text{sentencia.listnext= nulo} \\ \\ \textbf{Sentencia} \rightarrow \textbf{terminar} & I = \text{newIndex}() \\ \text{add_quad}(\text{code, "goto", -, -, 1}) \\ \text{sentencia.listnext= newList}() \\ \text{sentencia.listnext= newList}() \\ \text{sentencia.listnext.add}(I) \\ \\ \textbf{expresion_booleana} \rightarrow \textbf{expresion_booleana}_1 \ \textbf{oo} \\ \text{expresion_booleana}_2 & \text{backpatch}(\text{code, expresion_booleana}_1.\text{listfalse, L}) \\ \text{expresion_booleana}_2.\text{lissttrue}) \\ \text{expresion_booleana}_2.\text{listfalse} \\ \text{expresion_booleana}_2.\text{listfalse} \\ \text{expresion_booleana}_2.\text{listfalse} \\ \text{expresion_booleana}_2.\text{listfalse} \\ \end{array}$		
$\begin{array}{c} & \mid \operatorname{Error}(\text{``La funci\'on no puede retornar alg\'un valor de tipo ''}) \\ & \operatorname{fin} \\ & \operatorname{sentencia.listnext=nulo} \\ \\ \operatorname{sentencia} \to \operatorname{terminar} & I = \operatorname{newIndex}() \\ & \operatorname{add_quad}(\operatorname{code}, \text{``goto''}, , , I) \\ & \operatorname{sentencia.listnext} = \operatorname{newList}() \\ & \operatorname{sentencia.listnext.add}(I) \\ \\ \operatorname{expresion_booleana} \to \operatorname{expresion_booleana}_1 \operatorname{oo} \\ & \operatorname{expresion_booleana}_2 \\ & \operatorname{backpatch}(\operatorname{code}, \operatorname{expresion_booleana}_1.\operatorname{listfalse}, L) \\ & \operatorname{expresion_booleana}_2.\operatorname{listfure}) \\ & \operatorname{expresion_booleana}_2.\operatorname{listfalse} \\ & \operatorname{expresion_booleana}_2.\operatorname{listfalse} \\ & \operatorname{expresion_booleana}_2.\operatorname{listfalse} \\ & \operatorname{expresion_booleana}_2.\operatorname{listfalse} \\ \end{array}$		
$\begin{array}{c} \textbf{fin} \\ \text{sentencia.listnext= nulo} \\ \\ \textbf{Sentencia} \rightarrow \textbf{terminar} & \textbf{I} = \text{newIndex}() \\ \text{add_quad}(\text{code, "goto", -, -, I}) \\ \text{sentencia.listnext} = \text{newList}() \\ \text{sentencia.listnext.add}(\textbf{I}) \\ \\ \textbf{expresion_booleana} \rightarrow \text{expresion_booleana}_1 \ \textbf{oo} \\ \text{expresion_booleana}_2 & \textbf{backpatch}(\text{code, expresion_booleana}_1.\text{listfalse, L}) \\ \text{expresion_booleana}_1.\text{listtrue} = \text{combinar}(\text{expresion_booleana}_1.\text{listtrue}, \\ \text{expresion_booleana}_2.\text{listfalse} \\ \text{expresion_booleana}_2.\text{listfalse} \\ \text{expresion_booleana}_2.\text{listfalse} \\ \end{array}$		
$ \begin{array}{c} sentencia \rightarrow \textbf{terminar} & I = newIndex() \\ add_quad(code, "goto", -, -, I) \\ sentencia.listnext = newList() \\ sentencia.listnext.add(I) \\ expresion_booleana \rightarrow expresion_booleana_1 \ \textbf{oo} \\ expresion_booleana_2 & backpatch(code, expresion_booleana_1.listfalse, L) \\ expresion_booleana.listtrue = combinar(expresion_booleana_1.listtrue, \\ expresion_booleana_2.lissttrue) \\ expreson_booleana.listfalse = expresion_booleana_2.listfalse \\ \end{array} $		
$ \begin{array}{c} add_quad(code, "goto", -, -, I) \\ sentencia.listnext = newList() \\ sentencia.listnext.add(I) \\ \\ expresion_booleana \rightarrow expresion_booleana_1 \ oo \\ expresion_booleana_2 \\ \\ expresion_booleana_1.listfalse, L) \\ expresion_booleana_1.listtrue = combinar(expresion_booleana_1.listtrue, \\ expresion_booleana_2.lissttrue) \\ expresion_booleana_1.listfalse \\ expresion_booleana_2.listfalse \\ \end{array} $		sentencia.listnext= nulo
$\begin{array}{c} sentencia.listnext = newList() \\ sentencia.listnext.add(I) \\ \\ expresion_booleana \rightarrow expresion_booleana_1 \ oo \\ expresion_booleana_2 \\ \\ expresion_booleana_2 \\ \\ expresion_booleana.listtrue = combinar(expresion_booleana_1.listtrue, \\ expresion_booleana_2.lissttrue) \\ expresion_booleana.listfalse = expresion_booleana_2.listfalse \\ \end{array}$	$sentencia \rightarrow terminar$	
$\begin{array}{c} sentencia.listnext = newList() \\ sentencia.listnext.add(I) \\ \\ expresion_booleana \rightarrow expresion_booleana_1 \ oo \\ expresion_booleana_2 \\ \\ expresion_booleana_2 \\ \\ expresion_booleana.listtrue = combinar(expresion_booleana_1.listtrue, \\ expresion_booleana_2.lissttrue) \\ expresion_booleana.listfalse = expresion_booleana_2.listfalse \\ \end{array}$		
$\begin{array}{c} sentencia.listnext.add(I) \\ expresion_booleana \rightarrow expresion_booleana_1 \ oo \\ expresion_booleana_2 \\ & backpatch(code, expresion_booleana_1.listfalse, L) \\ expresion_booleana.listtrue = combinar(expresion_booleana_1.listtrue, \\ expresion_booleana_2.lissttrue) \\ expreson_booleana.listfalse = expresion_booleana_2.listfalse \\ \end{array}$		
expresion_booleana ₂ backpatch(code, expresion_booleana ₁ .listfalse, L) expresion_booleana.listtrue = combinar(expresion_booleana ₁ .listtrue, expresion_booleana ₂ .lissttrue) expresion_booleana ₂ .listfalse		· ·
expresion_booleana.listtrue = combinar(expresion_booleana ₁ .listtrue, expresion_booleana ₂ .lissttrue) expreson_booleana.listfalse= expresion_booleana ₂ .listfalse	expresion_booleana \rightarrow expresion_booleana $_1$ oo	
expresion_booleana ₂ .lissttrue) expreson_booleana.listfalse= expresion_booleana ₂ .listfalse	$expresion_booleana_2$	
expreson_booleana.listfalse= expresion_booleana ₂ .listfalse		\ -
11 1/ 1 11 11 T\		
add_quad(code, "label", -, -, L)		add_quad(code, "label", -, -, L)

PRODUCCIÓN	REGLAS SEMÁNTICAS
expresion_booleana \rightarrow expresion_booleana ₁ yy	L = newLabel()
$expresion_booleana_2$	backpatch(code, expresion_booleana ₁ .listtrue, L)
	expresion_booleana.listtrue = expresion_booleana2.listtrue
	expresion_booleana.listfalse = combinar(expresion_booleana ₁ .listfalse,
	expresion_booleana ₂ .lisstfalse)
	add_quad(code, "label", -, -, L)
expresion_booleana \rightarrow no expresion_booleana ₁	expresion_booleana.listtrue = expresion_booleana ₁ .listfalse
	expresion_booleana.listfalse = expresion_booleana ₁ .listtrue
expresion_booleana →relacional	expresion_booleana.listtrue = relacional.listtrue
	expresion_booleana.listfalse = relacional.listfalse
expresion_booleana → verdadero	I = newIndex()
1	expresion_booleana.listtrue = newList()
	$expresion_booleana.listtrue.add(I)$
	add_quad(code, "goto", -, -, I)
	expresion_booleana.listfalse = nulo
expresion_booleana \rightarrow falso	I = newIndex()
	expresion_booleana.listtrue = nulo
	expresion_booleana.listfalse = newList()
	$expresion_booleana.listfalse.add(I)$
	add_quad(code, "goto", -, -, I)
$relacional \rightarrow relacional_1 < relacional_2$	relacional.listtrue = newList()
rotational / rotational Crotational	relacional.listfalse = newList()
	I= newIndex(), I1 = newIndex()
	relacional.listtrue.add (I)
	relacional.listfalse.add(I1)
	relacional.tipo = $\max(\text{relacional1.tipo}, \text{relacional2.tipo})$
	$\alpha_1 = \text{ampliar}(\text{relacional}_1.\text{dir}, \text{relacional}_1.\text{tipo}, \text{relacional}_1.\text{tipo})$
	$\alpha_1 = \text{tampha}(\text{relacional}_1, \text{dir}, \text{relacional}_1, \text{tipo}, \text{relacional.tipo})$ $\alpha_2 = \text{ampliar}(\text{relacional}_2, \text{dir}, \text{relacional}_2, \text{tipo}, \text{relacional.tipo})$
	add_quad(code, " $<$ ", α_1 , α_2 , I)
	add_quad(code, "goto", -, -, I1)
$relacional \rightarrow relacional_1 > relacional_2$	relacional.listtrue = newList()
relacional / relacional / relacional	relacional.listfalse = newList()
	I = newIndex(), I1 = newIndex()
	relacional.listtrue.add (I)
	relacional.listfalse.add(I1)
	relacional.tipo = $\max(\text{relacional1.tipo}, \text{relacional2.tipo})$
	$\alpha_1 = \text{ampliar}(\text{relacional}_1.\text{dir}, \text{relacional}_1.\text{tipo}, \text{relacional}_1.\text{tipo})$
	$\alpha_1 = \text{ampha}(\text{relacional}_1.\text{dir}, \text{relacional}_1.\text{tipo}, \text{relacional}.\text{tipo})$ $\alpha_2 = \text{ampliar}(\text{relacional}_2.\text{dir}, \text{relacional}_2.\text{tipo}, \text{relacional}.\text{tipo})$
	add_quad(code, ">", α_1 , α_2 , I)
	add-quad(code, "goto", -, -, -, I1)
$relacional \rightarrow relacional_1 \le relacional_2$	relacional.listtrue = newList()
relacional — relacional — relacional2	relacional.listfalse = newList()
	I = newIndex(), I1 = newIndex()
	$\Gamma = \text{newIndex}(I), \Gamma = \text{newIndex}(I)$ relacional.listtrue.add(I)
	relacional.listfalse.add(I1)
	relacional.tipo = $\max(\text{relacional1.tipo}, \text{relacional2.tipo})$
	relacional.tipo = $\max(\text{relacional1.tipo}, \text{relacional2.tipo})$ $\alpha_1 = \text{ampliar}(\text{relacional1.dir}, \text{relacional1.tipo}, \text{relacional.tipo})$
	$\alpha_1 = \text{amphar(relacional_1.dir, relacional_1.tipo, relacional.tipo)}$ $\alpha_2 = \text{ampliar(relacional_2.dir, relacional_2.tipo, relacional.tipo)}$
	add_quad(code, " $<=$ ", α_1 , α_2 , I)
	add_quad(code, "goto", -, -, I1)

relacional \rightarrow relacional ₂ = relacional ₂ relacional ₂ relacional ₂ relacional ₂ relacional ₃ relacional ₄	PRODUCCIÓN	REGLAS SEMÁNTICAS
$ clacional. istfalse = newList() \\ L= newIndex(), II = newIndex() \\ clacional. istfalse.add(II) \\ relacional. istfalse.add(II) \\ relacional. istfalse.add(II) \\ relacional. istpo, relacional_ctipo, relacional_ctipo) \\ \alpha_1 = ampliar(relacional_a) dir, relacional_ctipo, relacional. tipo) \\ \alpha_2 = ampliar(relacional_a) dir, relacional_ctipo, relacional. tipo) \\ add quad(code, ">c^*, >=^*, \alpha_1, \alpha_2, II) \\ add quad(code, "soto", -, -, II) \\ relacional == relacional_2 relacional_istfalse. = newList() \\ l = newIndex(), II = newIndex() \\ l = newIndex(), II = newIndex() \\ relacional. istfalse. = newList() \\ l = newIndex(), II = newIndex() \\ relacional. istfalse. = newList() \\ relacional. istfalse. = newList() \\ relacional. istfalse. = newList() \\ relacional. istfalse. istfalse.$	$relacional \rightarrow relacional_1 >= relacional_2$	relacional.listtrue = newList()
$ \begin{vmatrix} & & & & & & & & & &$		
$ \begin{array}{c} \operatorname{relacional.listrue.add(I)} \\ \operatorname{relacional.listfalsc.add(II)} \\ \operatorname{relacional.litipo} = \max_{\alpha_1} \operatorname{relacional_1.tipo}, \operatorname{relacional_2.tipo}) \\ a_1 = \operatorname{ampliar}(\operatorname{relacional_1.dir}, \operatorname{relacional_1.tipo}, \operatorname{relacional.tipo}) \\ a_2 = \operatorname{ampliar}(\operatorname{relacional_2.dir}, \operatorname{relacional_2.tipo}, \operatorname{relacional.tipo}) \\ add_{\operatorname{quad}}(\operatorname{code}, ">=e^*, a_1, a_2, 1) \\ \operatorname{add.quad}(\operatorname{code}, ">=e^*, a_1, a_2, 1) \\ \operatorname{add.quad}(\operatorname{code}, ">=e^*, a_1, a_2, 1) \\ \operatorname{relacional} \rightarrow \operatorname{relacional_2} \\ \end{array} $		
$ \begin{array}{c} \text{relacional.listfalse.add(l1)} \\ \text{relacional.tipo} = \max(\text{relacional}1.\text{tipo}, \text{ relacional}2.\text{tipo}) \\ \alpha_1 = \operatorname{ampliar}(\text{relacional}1.\text{dir}, \text{ relacional}1.\text{tipo}, \text{ relacional.tipo}) \\ \alpha_2 = \operatorname{ampliar}(\text{relacional}2.\text{dir}, \text{ relacional}2.\text{tipo}, \text{ relacional.tipo}) \\ \alpha_2 = \operatorname{ampliar}(\text{relacional}2.\text{dir}, \text{ relacional}2.\text{tipo}, \text{ relacional.tipo}) \\ \text{add.quad}(\text{code}, "\text{goto"}, -, -, 11) \\ \text{relacional} - rela$		
$\begin{array}{c} \text{relacional.tipo} = \max(\text{relacional}.\text{tipo}, \text{relacional}.\text{tipo}) & \alpha_1 = \text{ampliar}(\text{relacional}.\text{dir}, \text{relacional}.\text{tipo}, \text{relacional}.\text{tipo}) \\ \alpha_2 = \text{ampliar}(\text{relacional}.\text{dir}, \text{relacional}.\text{tipo}, \text{relacional}.\text{tipo}) \\ \text{add.quad}(\text{code}, ">=", \alpha_1, \alpha_2, \mathbf{I}) \\ \text{add.quad}(\text{code}, "poto", -, -, \mathbf{II}) \\ \end{array}$ $\begin{array}{c} \text{relacional} \rightarrow \text{relacional}_1 = \text{relacional}_2 \\ \text{relacional.listfrue} = \text{newList}() \\ \text{relacional.listfrue} = \text{newList}() \\ \text{relacional.listfrue.add}(\mathbf{I}) \\ \text{relacional.listfrue.add}(\mathbf{I}) \\ \text{relacional.listfrue.add}(\mathbf{I}) \\ \text{relacional.listfrue.add}(\mathbf{I}) \\ \text{relacional.listfrue} = \text{newList}() \\ \text{relacional.listfrue} =$		
$ \begin{array}{c} \alpha_1 = \text{ampliar}(\text{relacional}_1 \text{dir}, \text{relacional}_1, \text{tipo}, \text{relacional}_1 \text{tipo}_1 \text{o}_2 = \text{ampliar}(\text{relacional}_2 \text{dir}, \text{relacional}_2, \text{tipo}_1 \text{relacional}_1 \text{tipo}_1 \text{add.quad}(\text{code}, ">=", \alpha_1, \alpha_2, 1, \alpha_2, 1) \text{add.quad}(\text{code}, "poto", -, -, 11) \\ \hline \\ \text{relacional}_1 \rightarrow \text{relacional}_1 == \text{relacional}_2 \\ \hline \\ \text{relacional.listrue}_1 = \text{newList}() \\ \text{relacional.listrue}_1 = \text{newList}() \\ \text{relacional.listrue}_2 = \text{newList}() \\ \text{lamewindex}(), 11 = \text{newIndex}() \text{relacional}_2, \text{tipo}_1 \text{relacional}_2, \text{tipo}_1 \text{relacional.listral}_2 \text{add}(1) \text{relacional}_1, \text{tipo}_1 \text{relacional}_2, \text{tipo}_2 \text{relacional}_2, \text{tipo}_1 \text{relacional}_2, \text{tipo}_2 \text{relacional}_2, \text{tipo}_1 \text{relacional}_2, \text{tipo}_2 \text{relacional}_2, \text{tipo}_2 $		
$\begin{array}{c} a_2 = \operatorname{ampliar(relacional_2, \operatorname{dir}, \operatorname{relacional_2, \operatorname{tipo}, \operatorname{relacional.tipo})} \\ \operatorname{add.quad(code, ">=", a_1, a_2, 1)} \\ \operatorname{add.quad(code, "poto", -, -, 11)} \\ \end{array}$ $\begin{array}{c} \operatorname{relacional} \rightarrow \operatorname{relacional}_1 == \operatorname{relacional}_2 \\ \end{array}$ $\begin{array}{c} \operatorname{relacional} \operatorname{distrue} = \operatorname{newList}() \\ \operatorname{relacional} \operatorname{distrue} = \operatorname{newList}() \\ \operatorname{relacional} \operatorname{distrue} \operatorname{add}(1) \\ \operatorname{relacional} \operatorname{distrue} $		
$ \begin{array}{c} \operatorname{add.quad(code, ">=", \alpha_1, \alpha_2, 1)} \\ \operatorname{add.quad(code, "goto", -, -, 11)} \\ \operatorname{relacional.} = \operatorname{relacional}_2 \\ \operatorname{relacional.} = \operatorname{relacional}_2 \\ \operatorname{relacional.} = \operatorname{relacional.} \\ re$		
$ \begin{array}{c} \operatorname{add.quad(code, "goto", \neg, \neg, 11)} \\ \operatorname{relacional} \to \operatorname{relacional}_1 == \operatorname{relacional}_2 \\ \operatorname{relacional.listrue} = \operatorname{newList}() \\ \operatorname{l= newIndex}(), \ \operatorname{l= newIndex}() \\ \operatorname{relacional.listrue.add}(1) \\ \operatorname{relacional.listrue.add}(1) \\ \operatorname{relacional.listrue.add}(1) \\ \operatorname{relacional.listrue.add}_1, \operatorname{dir, relacional, tipo, relacional.tipo}) \\ o_1 = \operatorname{ampliar}(\operatorname{relacional}_1, \operatorname{dir, relacional}_1, \operatorname{tipo, relacional.tipo}) \\ o_2 = \operatorname{ampliar}(\operatorname{relacional}_2, \operatorname{dir, relacional}_2, \operatorname{tipo, relacional.tipo}) \\ \operatorname{add.quad}(\operatorname{code}, "==", \alpha_1, \alpha_2, 1] \\ \operatorname{add.quad}(\operatorname{code}, "==", \alpha_1, \alpha_2, 1] \\ \operatorname{add.quad}(\operatorname{code}, "== \text{meV.list}() \\ \operatorname{relacional.listrue} = \operatorname{newList}() \\ \operatorname{l= newIndex}(), \ \operatorname{l= newIndex}(), \ \operatorname{l= newIndex}() \\ \operatorname{relacional.listrue} = \operatorname{newList}() \\ \operatorname{relacional.listrue} = \operatorname{newList}() \\ \operatorname{relacional.listrue} = \operatorname{newList}() \\ \operatorname{relacional.listrue} = \operatorname{newList}() \\ \operatorname{l= newIndex}(), \ $		
$ \begin{array}{c} \operatorname{relacional} \rightarrow \operatorname{relacional}_1 == \operatorname{relacional}_2 \\ \operatorname{relacional}_{\operatorname{list}false} = \operatorname{newList}() \\ \operatorname{I=newIndex}(), \operatorname{II=newIndex}() \\ \operatorname{relacional}_{\operatorname{list}false} = \operatorname{newList}() \\ \operatorname{I=newIndex}(), \operatorname{II=newIndex}() \\ \operatorname{relacional}_{\operatorname{list}false} = \operatorname{newList}() \\ \operatorname{relacional}_{\operatorname{list}false} = \operatorname{newList}() \\ \operatorname{relacional}_{\operatorname{lit}} = \operatorname{relacional}_{\operatorname{l}}_{\operatorname{tipo}}, \operatorname{relacional}_{\operatorname{l}}_{\operatorname{tipo}}, \operatorname{relacional}_{\operatorname{l}}_{\operatorname{tipo}}, \operatorname{relacional}_{\operatorname{l}}_{\operatorname{tipo}}, \operatorname{relacional}_{\operatorname{l}}_{\operatorname{tipo}}, \operatorname{relacional}_{\operatorname{l}}_{\operatorname{tipo}}, \operatorname{relacional}_{\operatorname{l}}_{\operatorname{tipo}}, \operatorname{relacional}_{\operatorname{l}}_{\operatorname{tipo}}, \operatorname{relacional}_{\operatorname{l}}_{\operatorname{lito}}, \operatorname{relacional}_{\operatorname{l}}_{\operatorname{lito}}, \operatorname{relacional}_{\operatorname{l}}_{\operatorname{lito}}, \operatorname{relacional}_{\operatorname{l}}_{\operatorname{lito}}, \operatorname{relacional}_{\operatorname{l}}_{\operatorname{lito}}, \operatorname{relacional}_{\operatorname{l}}_{\operatorname{lito}}, \operatorname{relacional}_{\operatorname{l}}_{\operatorname{lito}}, \operatorname{relacional}_{\operatorname{l}}_{\operatorname{lito}}, \operatorname{relacional}_{\operatorname{l}}_{\operatorname{l}}_{\operatorname{lito}}, \operatorname{relacional}_{\operatorname{l}}_{\operatorname{l}}_{\operatorname{lito}}, \operatorname{relacional}_{\operatorname{l}}_{\operatorname{l}}_{\operatorname{lito}}, \operatorname{relacional}_{\operatorname{l}}_{\operatorname{l}}_{\operatorname{lito}}, \operatorname{relacional}_{\operatorname{l}}_{\operatorname{l}}_{\operatorname{lito}}, \operatorname{relacional}_{\operatorname{l}}_{\operatorname{l}}_{\operatorname{lito}}, \operatorname{relacional}_{\operatorname{l}}_{\operatorname{l}}_{\operatorname{lito}}, \operatorname{relacional}_{\operatorname{l}}_{\operatorname{lito}}, \operatorname{relacional}_{\operatorname{l}}_{\operatorname{l}}_{\operatorname{lito}}, \operatorname{relacional}_{\operatorname{l}}_{\operatorname{l}}_{\operatorname{lito}}, \operatorname{relacional}_{\operatorname{l}}_{\operatorname{l}}_{\operatorname{lito}}, \operatorname{relacional}_{\operatorname{l}}_{\operatorname{lito}}, $		
$\begin{array}{c} \operatorname{relacional.listfalse} = \operatorname{newList}() \\ \operatorname{I=newIndex}(), \ \Pi = \operatorname{newIndex}() \\ \operatorname{relacional.listrue.add}(1) \\ \operatorname{relacional.listrue.add}(1) \\ \operatorname{relacional.listrue.add}(1) \\ \operatorname{relacional.listpo} = \operatorname{max}(\operatorname{relacional1.tipo}, \operatorname{relacional2.tipo}) \\ \alpha_1 = \operatorname{ampliar}(\operatorname{relacional1.dir}, \operatorname{relacional2.tipo}, \operatorname{relacional.tipo}) \\ \alpha_2 = \operatorname{ampliar}(\operatorname{relacional2.dir}, \operatorname{relacional2.tipo}, \operatorname{relacional.tipo}) \\ \operatorname{add.quad}(\operatorname{code}, "\operatorname{gue"}, -, \cdot, \Pi) \\ \\ \operatorname{relacional} \rightarrow \operatorname{relacional2} > \operatorname{relacional2} \\ \operatorname{relacional.listrue} = \operatorname{newList}() \\ \operatorname{relacional.listrue} = \operatorname{newList}() \\ \operatorname{relacional.listrue.add}(1) \\ \operatorname{relacional.listrue.add}(1) \\ \operatorname{relacional.listrue.add}(1) \\ \operatorname{relacional.listrue.add}(1) \\ \operatorname{relacional.listrue.add}(1) \\ \operatorname{relacional.listrue.add}(1) \\ \operatorname{relacional.lipo} = \operatorname{max}(\operatorname{relacional2.tipo}, \operatorname{relacional.tipo}) \\ \alpha_1 = \operatorname{ampliar}(\operatorname{relacional1.dir}, \operatorname{relacional2.tipo}, \operatorname{relacional.tipo}) \\ \alpha_2 = \operatorname{ampliar}(\operatorname{relacional2.dir}, \operatorname{relacional2.tipo}, \operatorname{relacional.tipo}) \\ \alpha_2 = \operatorname{ampliar}(\operatorname{relacional2.dir}, \operatorname{relacional2.tipo}, \operatorname{relacional.tipo}) \\ \operatorname{add.quad}(\operatorname{code}, "\circo", -, \cdot, \Pi) \\ \operatorname{relacional.dir} = \operatorname{expresion.tipo} \\ \operatorname{relacional.dir} = \operatorname{expresion.tipo} \\ \operatorname{relacional.dir} = \operatorname{expresion.tipo} \\ \operatorname{relacional.dir} = \operatorname{expresion.tipo}, \operatorname{expresion.tipo}) \\ \operatorname{add.quad}(\operatorname{code}, "+, -, -, -, -, -, -, -, -, -, -, -, -, -,$	$relacional \rightarrow relacional_1 == relacional_2$	
	<u> </u>	
$\begin{array}{c} \text{relacional.listTrue.add(I)} \\ \text{relacional.tipo} = \max(\text{relacional}_1.\text{tipo}, \text{ relacional}_2.\text{tipo}) \\ \alpha_1 = \text{ampliar}(\text{relacional}_1.\text{dir}, \text{ relacional}_2.\text{tipo}, \text{ relacional}_1.\text{tipo}, \text{ relacional}_2.\text{tipo}) \\ \alpha_2 = \text{ampliar}(\text{relacional}_2.\text{dir}, \text{ relacional}_2.\text{tipo}, \text{ relacional}_1.\text{tipo}) \\ \alpha_2 = \text{ampliar}(\text{relacional}_2.\text{dir}, \text{ relacional}_2.\text{tipo}, \text{ relacional}_1.\text{tipo}) \\ \text{add-quad}(\text{code}, "==", \alpha_1, \alpha_2, 1) \\ \text{add-quad}(\text{code}, "goto", -, -, 11) \\ \text{relacional} \rightarrow \text{ relacional}_2 \\ \text{relacional}.\text{ listTrue} = \text{newList()} \\ \text{relacional}.\text{ listTake} = \text{newList()} \\ \text{relacional}.\text{ listTake} = \text{newList()} \\ \text{relacional}.\text{ listTake}.\text{ add(I)} \\ \text{relacional}.\text{ listTake}.\text{ add(I)} \\ \text{relacional}.\text{ lipo} = \text{max}(\text{relacional}_2.\text{tipo}, \text{ relacional}_2.\text{tipo}) \\ \alpha_1 = \text{ampliar}(\text{relacional}_1.\text{dir}, \text{ relacional}_2.\text{tipo}, \text{ relacional}_1.\text{tipo}, \text{ relacional}_2.\text{tipo}) \\ \alpha_2 = \text{ampliar}(\text{relacional}_2.\text{dir}, \text{ relacional}_2.\text{tipo}, \text{ relacional}_2.\text{tipo}) \\ \alpha_2 = \text{ampliar}(\text{relacional}_2.\text{dir}, \text{ relacional}_2.\text{tipo}, \text{ relacional}_2.\text{tipo}) \\ \text{add-quad}(\text{code}, "oo', -, -, II) \\ \text{relacional} \rightarrow \text{ expresion} \\ \text{relacional} \rightarrow \text{ expresion}, \text{ dir} = \text{ expresion}.\text{ tipo} \\ \text{expresion} \rightarrow \text{ expresion}_1 + \text{ expresion}_2 \\ \text{expresion} \\ \text{expresion} \text{ dir} = \text{ newTemp()} \\ \alpha_1 = \text{ ampliar}(\text{expresion}_2.\text{dir}, \text{ expresion}_2.\text{tipo}, \text{ expresion}_2.\text{tipo}) \\ \text{expresion} \\ \text{dir} = \text{ newTemp()} \\ \alpha_1 = \text{ ampliar}(\text{expresion}_2.\text{dir}, \text{ expresion}_2.\text{tipo}, \text{ expresion}_2.\text{tipo}) \\ \text{expresion} \\ \text{dir} = \text{ newTemp()} \\ \alpha_1 = \text{ ampliar}(\text{expresion}_2.\text{dir}, \text{ expresion}_2.\text{tipo}, \text{ expresion}_2.\text{tipo}) \\ \text{expresion} \\ \text{dir} = \text{ newTemp()} \\ \alpha_1 = \text{ ampliar}(\text{expresion}_2.\text{dir}, \text{ expresion}_2.\text{tipo}, \text{ expresion}_2.\text{tipo}) \\ \text{expresion} \\ \text{dir} = \text{ newTemp()} \\ \alpha_1 = \text{ ampliar}(\text{expresion}_2.\text{dir}, \text{ expresion}_2.\text{tipo}, \text{ expresion}_2.\text{tipo}) \\ \text{expresion} \\ \text{dir} = $		
$\begin{array}{c} \text{relacional.listfalse.add}(\overline{11}) \\ \text{relacional.tipo} = \max(\text{relacional.tipo}, \text{ relacional}_2, \text{tipo}) \\ \alpha_1 = \text{ampliar}(\text{relacional}_1, \text{dir}, \text{ relacional}_2, \text{tipo}, \text{ relacional.tipo}) \\ \alpha_2 = \text{ampliar}(\text{relacional}_2, \text{dir}, \text{ relacional}_2, \text{tipo}, \text{ relacional.tipo}) \\ \alpha_2 = \text{ampliar}(\text{relacional}_2, \text{dir}, \text{ relacional}_2, \text{tipo}, \text{ relacional.tipo}) \\ \text{add.quad}(\text{code}, "==", \alpha_1, \alpha_2, 1) \\ \text{add.quad}(\text{code}, "goto", -, -, 11) \\ \text{relacional} \rightarrow \text{ relacional}_2 \\ \text{relacional.listtrue} = \text{newList}() \\ \text{l= newIndex}(), \text{ II} = \text{newIndex}() \\ \text{relacional.listfralse} = \text{newList}() \\ \text{l= newIndex}(), \text{ II} = \text{newIndex}() \\ \text{relacional.listfalse} = \text{add}(11) \\ \text{relacional.listfalse.add}(11) \\ \text{relacional.listfalse.add}(11) \\ \text{relacional.listfalse.add}(11) \\ \text{relacional.listfalse.add}(11) \\ \text{relacional.listfalse.add}(11) \\ \text{relacional.listfalse.add}(12) \\ \text{relacional.listfalse.add}(13) \\ r$		
$\begin{array}{c} \text{relacional.tipo} = \max(\text{relacional1.tipo}, \text{ relacional2.tipo}) \\ \alpha_1 = \text{ampliar}(\text{relacional1.dir}, \text{ relacional1.tipo}, \text{ relacional1.tipo}) \\ \alpha_2 = \text{ampliar}(\text{relacional2.dir}, \text{ relacional2.tipo}, \text{ relacional.tipo}) \\ \alpha_2 = \text{ampliar}(\text{relacional2.dir}, \text{ relacional2.tipo}, \text{ relacional.tipo}) \\ \text{add-quad}(\text{code}, "geto", -, -, II) \\ \text{relacional} \rightarrow \text{ relacional2} \\ \text{relacional.listfure} = \text{newList}() \\ \text{relacional.listfalse} = \text{newList}() \\ \text{l= newIndex}(), \text{ II} = \text{newIndex}() \\ \text{relacional.listfure.add}(I) \\ \text{relacional.listfupe} = \text{max}(\text{relacional2.tipo}, \text{ relacional2.tipo}) \\ \alpha_1 = \text{ampliar}(\text{relacional1.tipo}, \text{ relacional2.tipo}, \text{ relacional1.tipo}) \\ \alpha_2 = \text{ampliar}(\text{relacional2.dir}, \text{ relacional2.tipo}, \text{ relacional.tipo}) \\ \alpha_2 = \text{ampliar}(\text{relacional2.dir}, \text{ relacional2.tipo}, \text{ relacional.tipo}) \\ \text{add-quad}(\text{code}, ">>", \alpha_1, \alpha_2, I) \\ \text{add-quad}(\text{code}, "goto", -, -, II) \\ \text{relacional} \rightarrow \text{ expresion} \\ \text{expresion} \rightarrow \text{ expresion.tipo} \\ \text{expresion.dir} = \text{ expresion.tipo} \\ \text{expresion.dir} = \text{ expresion.tipo}, \text{ expresion2.tipo}) \\ \text{expresion.dir} = \text{ newTemp}() \\ \alpha_1 = \text{ ampliar}(\text{expresion2.dir}, \text{ expresion.tipo}, \text{ expresion.tipo}) \\ \text{add-quad}(\text{code}, "+", \alpha_1, \alpha_2, \text{ expresion.tipo}) \\ \text{expresion.dir} = \text{ newTemp}() \\ \alpha_1 = \text{ ampliar}(\text{expresion1.tipo}, \text{ expresion.tipo}) \\ \text{expresion.dir} = \text{ newTemp}() \\ \alpha_1 = \text{ ampliar}(\text{expresion1.tipo}, \text{ expresion.tipo}) \\ \text{expresion.dir} = \text{ newTemp}() \\ \alpha_1 = \text{ ampliar}(\text{ expresion1.tipo}, \text{ expresion.tipo}) \\ \text{expresion.dir} = \text{ newTemp}() \\ \alpha_1 = \text{ ampliar}(\text{ expresion1.tipo}, \text{ expresion.tipo}) \\ \text{expresion.dir} = \text{ newTemp}() \\ \alpha_1 = \text{ ampliar}(\text{ expresion1.tipo}, \text{ expresion.tipo}) \\ \text{expresion.dir} = \text{ newTemp}() \\ \alpha_2 = \text{ ampliar}(\text{ expresion1.tipo}, \text{ expresion.tipo}) \\ \text{expresion.dir} = \text{ expresion.dir}, \text{ expresion.tipo}, \text{ expresion.tipo}) \\ \text{expresion.dir} = \text{ expresion.dir}, \text{ expresion.tipo}) \\ \text{expresion.dir} = expresi$		
$\begin{array}{c} \alpha_1 = \operatorname{ampliar}(\operatorname{relacional_1.dir}, \operatorname{relacional_1.tipo}, \operatorname{relacional_1.tipo}) \\ \alpha_2 = \operatorname{ampliar}(\operatorname{relacional_2.dir}, \operatorname{relacional_2.tipo}, \operatorname{relacional.tipo}) \\ \operatorname{add.quad}(\operatorname{code}, "=="\alpha_1, \alpha_2, I) \\ \operatorname{add.quad}(\operatorname{code}, "=\operatorname{ord}, \alpha_1, \alpha_2, I) \\ \operatorname{add.quad}(\operatorname{code}, "=\operatorname{ord}, \alpha_1, \alpha_2, I) \\ \operatorname{relacional}.\operatorname{listrue} = \operatorname{newList}() \\ \operatorname{l=newIndex}(), \operatorname{l1} = \operatorname{newIndex}() \\ \operatorname{relacional.listrue.add}(I) \\ \operatorname{relacional.lipo} = \operatorname{max}(\operatorname{relacional_1.tipo}, \operatorname{relacional_2.tipo}) \\ \alpha_1 = \operatorname{ampliar}(\operatorname{relacional_1.dir}, \operatorname{relacional_1.tipo}, \operatorname{relacional.tipo}) \\ \alpha_2 = \operatorname{ampliar}(\operatorname{relacional_1.dir}, \operatorname{relacional_2.tipo}, \operatorname{relacional.tipo}) \\ \operatorname{add.quad}(\operatorname{code}, ">>", \alpha_1, \alpha_2, I) \\ \operatorname{add.quad}(\operatorname{code}, ">>", \alpha_1, \alpha_2, I) \\ \operatorname{add.quad}(\operatorname{code}, ">>", \alpha_1, \alpha_2, I) \\ \operatorname{add.quad}(\operatorname{code}, "= \operatorname{ord}, \alpha_1, \alpha_2, \operatorname{cyresion_2.tipo}) \\ \operatorname{expresion} \to \operatorname{expresion_1} + \operatorname{expresion_2} \\ \operatorname{expresion.tipo} = \operatorname{max}(\operatorname{expresion_1.tipo}, \operatorname{expresion_2.tipo}) \\ \operatorname{expresion} \to \operatorname{expresion_1} - \operatorname{expresion_1} - \operatorname{expresion_2} \\ \operatorname{expresion.tipo} = \operatorname{max}(\operatorname{expresion_1.tipo}, \operatorname{expresion_2.tipo}) \\ \operatorname{expresion.dir} = \operatorname{newTemp}() \\ \alpha_1 = \operatorname{ampliar}(\operatorname{expresion_1.dir}, \operatorname{expresion_2.tipo}, \operatorname{expresion.tipo}) \\ \operatorname{expresion.dir} = \operatorname{newTemp}() \\ \alpha_1 = \operatorname{ampliar}(\operatorname{expresion_2.dir}, \operatorname{expresion_2.tipo}, \operatorname{expresion.tipo}) \\ \operatorname{expresion.dir} = \operatorname{newTemp}() \\ \alpha_1 = \operatorname{ampliar}(\operatorname{expresion_2.dir}, \operatorname{expresion_2.tipo}, \operatorname{expresion.tipo}) \\ \operatorname{expresion.dir} = \operatorname{newTemp}() \\ \alpha_2 = \operatorname{ampliar}(\operatorname{expresion_2.dir}, \operatorname{expresion_2.tipo}, \operatorname{expresion.tipo}) \\ \operatorname{expresion.dir} = \operatorname{expresion.dir}) \\ \operatorname{expresion.dir}$		
$\begin{array}{c} \alpha_2 = \operatorname{ampliar}(\operatorname{relacional_2.dir}, \operatorname{relacional_2.tipo}, \operatorname{relacional.tipo}) \\ \operatorname{add.quad}(\operatorname{code}, "==", \alpha_1, \alpha_2, 1) \\ \operatorname{add.quad}(\operatorname{code}, "\operatorname{goto"}, \neg, \neg, \operatorname{II}) \\ \end{array}$ $\begin{array}{c} \operatorname{relacional} \rightarrow \operatorname{relacional_1} <> \operatorname{relacional_2} \\ \end{array}$ $\begin{array}{c} \operatorname{relacional.listrue} = \operatorname{newList}() \\ \operatorname{relacional.listrue} = \operatorname{newList}() \\ \operatorname{l=newIndex}(), \operatorname{II} = \operatorname{newIndex}() \\ \operatorname{relacional.listrue} = \operatorname{add}(\operatorname{II}) \\ \operatorname{relacional.listrue} = \operatorname{add}(\operatorname{II}) \\ \operatorname{relacional.listrue} = \operatorname{add}(\operatorname{II}) \\ \operatorname{relacional.listpo} = \operatorname{max}(\operatorname{relacional_1.tipo}, \operatorname{relacional_2.tipo}) \\ \alpha_1 = \operatorname{ampliar}(\operatorname{relacional_1.tipo}, \operatorname{relacional_1.tipo}, \operatorname{relacional.tipo}) \\ \alpha_2 = \operatorname{ampliar}(\operatorname{relacional_2.dir}, \operatorname{relacional_2.tipo}, \operatorname{relacional.tipo}) \\ \alpha_2 = \operatorname{ampliar}(\operatorname{relacional_2.dir}, \operatorname{relacional_2.tipo}, \operatorname{relacional.tipo}) \\ \operatorname{add.quad}(\operatorname{code}, "o-", \alpha_1, \alpha_2, \operatorname{I}) \\ \operatorname{add.quad}(\operatorname{code}, "goto", \neg, \neg, \operatorname{II}) \\ \end{array}$ $\begin{array}{c} \operatorname{relacional} \rightarrow \operatorname{expresion} \\ \operatorname{relacional} \rightarrow \operatorname{expresion.dir} \\ \operatorname{expresion.dir} = \operatorname{expresion.dir} \\ \operatorname{expresion.dir} = \operatorname{expresion.dir} \\ \operatorname{expresion.dir} = \operatorname{newTemp}() \\ \alpha_1 = \operatorname{ampliar}(\operatorname{expresion_2.tipo}, \operatorname{expresion_2.tipo}, \operatorname{expresion.tipo}) \\ \operatorname{expresion.dir} = \operatorname{newTemp}() \\ \alpha_1 = \operatorname{ampliar}(\operatorname{expresion_2.tipo}, \operatorname{expresion_2.tipo}, \operatorname{expresion.tipo}) \\ \operatorname{expresion.dir} = \operatorname{newTemp}() \\ \alpha_1 = \operatorname{ampliar}(\operatorname{expresion_2.tipo}, \operatorname{expresion.tipo}, \operatorname{expresion.tipo}) \\ \operatorname{expresion.dir} = \operatorname{newTemp}() \\ \alpha_1 = \operatorname{ampliar}(\operatorname{expresion_2.tipo}, \operatorname{expresion.tipo}, \operatorname{expresion.tipo}) \\ \operatorname{expresion.dir} = \operatorname{newTemp}() \\ \alpha_1 = \operatorname{ampliar}(\operatorname{expresion_2.tipo}, \operatorname{expresion.tipo}, \operatorname{expresion.tipo}) \\ \operatorname{expresion.dir} = \operatorname{expresion_2.tipo}, \operatorname{expresion.tipo}, \operatorname{expresion.tipo}) \\ \operatorname{expresion.dir} = \operatorname{expresion_2.tipo}, \operatorname{expresion.tipo}, \operatorname{expresion.tipo}, \operatorname{expresion.2.tipo}, \operatorname{expresion.tipo}, \operatorname{expresion.2.tipo}, $		
$ \begin{array}{c} \operatorname{add.quad(code,"==",\alpha_1,\alpha_2,I)} \\ \operatorname{add.quad(code,"goto",\cdot,\cdot,II)} \\ \\ \operatorname{relacional} \to \operatorname{relacional}_1 <> \operatorname{relacional}_2 \\ \\ \operatorname{relacional.listfalse} = \operatorname{newList}() \\ \operatorname{I=newIndex}(),II = \operatorname{newIndex}() \\ \operatorname{relacional.listfalse} = \operatorname{newList}() \\ \operatorname{I=newIndex}(),II = \operatorname{newIndex}() \\ \operatorname{relacional.listfalse.add}(II) \\ \operatorname{relacional.listfalse.add}(II) \\ \operatorname{relacional.listpo} = \operatorname{max}(\operatorname{relacional.tipo},\operatorname{relacional.tipo}) \\ \alpha_1 = \operatorname{ampliar}(\operatorname{relacional}_1,\operatorname{dir},\operatorname{relacional.tipo},\operatorname{relacional.tipo}) \\ \alpha_2 = \operatorname{ampliar}(\operatorname{relacional}_2,\operatorname{dir},\operatorname{relacional.tipo},\operatorname{relacional.tipo}) \\ \alpha_2 = \operatorname{ampliar}(\operatorname{relacional}_2,\operatorname{dir},\operatorname{relacional.tipo},\operatorname{relacional.tipo}) \\ \operatorname{add.quad}(\operatorname{code},"\circ,"\circ, \alpha_1,\alpha_2,I) \\ \operatorname{add.quad}(\operatorname{code},"\operatorname{goto"}, \cdot, \cdot,\operatorname{II}) \\ \operatorname{relacional} \to \operatorname{expresion} \to \operatorname{expresion.tipo} \\ \operatorname{relacional.dir} = \operatorname{expresion.tipo} \\ \operatorname{expresion.dir} = \operatorname{expresion.tipo} = \operatorname{max}(\operatorname{expresion.tipo},\operatorname{expresion.2,tipo}) \\ \operatorname{expresion.dir} = \operatorname{newTemp}() \\ \alpha_1 = \operatorname{ampliar}(\operatorname{expresion}_1,\operatorname{dir},\operatorname{expresion.tipo},\operatorname{expresion.tipo}) \\ \operatorname{add.quad}(\operatorname{code},"+",\alpha_1,\alpha_2,\operatorname{expresion.tipo},\operatorname{expresion.tipo}) \\ \operatorname{expresion.tipo} = \operatorname{max}(\operatorname{expresion}_1,\operatorname{tipo},\operatorname{expresion.tipo}) \\ \operatorname{expresion.tipo} = \operatorname{max}(\operatorname{expresion}_1,\operatorname{tipo},\operatorname{expresion.tipo}) \\ \operatorname{add.quad}(\operatorname{code},"-",\alpha_1,\alpha_2,\operatorname{expresion.tipo},\operatorname{expresion.tipo}) \\ \operatorname{add.quad}(\operatorname{code},"-",\alpha_1,\alpha_2,\operatorname{expresion.dir}) \\ \operatorname{add.quad}(\operatorname{code},"-",\alpha_1,\alpha_$		
$ \begin{array}{c} \operatorname{add.quad(code, "goto", -, -, 11)} \\ \operatorname{relacional.} = \operatorname{rewList}() \\ \operatorname{relacional.} \operatorname{listfalse} = \operatorname{newList}() \\ \operatorname{le newIndex}(), 11 = \operatorname{newIndex}() \\ \operatorname{relacional.} \operatorname{listfalse} = \operatorname{newList}() \\ \operatorname{le newIndex}(), 11 = \operatorname{newIndex}() \\ \operatorname{relacional.} \operatorname{listfalse} = \operatorname{newIndex}() \\ \operatorname{relacional.} \operatorname{listfalse} \operatorname{add}(11) \\ \operatorname{relacional.} \operatorname{listfalse} \operatorname{add}(11) \\ \operatorname{relacional.} \operatorname{listp} = \operatorname{max}(\operatorname{relacional.} \operatorname{lipp}, \operatorname{relacional.} \operatorname{tipp}) \\ \alpha_1 = \operatorname{ampliar}(\operatorname{relacional.} \operatorname{dir}, \operatorname{relacional.} \operatorname{tipp}, \operatorname{relacional.} \operatorname{tipp}) \\ \alpha_2 = \operatorname{ampliar}(\operatorname{relacional.} \operatorname{dir}, \operatorname{relacional.} \operatorname{tipp}, \operatorname{relacional.} \operatorname{tipp}) \\ \alpha_2 = \operatorname{ampliar}(\operatorname{relacional.} \operatorname{dir}, \operatorname{relacional.} \operatorname{tipp}, \operatorname{relacional.} \operatorname{tipp}) \\ \operatorname{add.quad}(\operatorname{code}, "goto", -, -, 11) \\ \operatorname{relacional.} \operatorname{dir} = \operatorname{expresion.} \operatorname{dir} \\ \operatorname{expresion} \to \operatorname{expresion}_1 + \operatorname{expresion}_2 \\ \operatorname{expresion.} \operatorname{tipp} = \operatorname{max}(\operatorname{expresion.} \operatorname{tipp}) \\ \operatorname{expresion.} \operatorname{dir} = \operatorname{expresion.} \operatorname{tipp}, \operatorname{expresion.} \operatorname{tipp}) \\ \operatorname{add.quad}(\operatorname{code}, "a, a, a$		add_quad(code, "==", α_1 , α_2 , I)
$\begin{array}{c} \text{relacional.listfalse} = \text{newList}() \\ \text{I=} \text{newIndex}(), \text{II} = \text{newIndex}() \\ \text{relacional.listfalse.add}(\text{I}) \\ \text{relacional.listfalse.add}(\text{I}) \\ \text{relacional.lipo} = \text{max}(\text{relacional1.tipo}, \text{relacional2.tipo}) \\ \alpha_1 = \text{ampliar}(\text{relacional1.dir}, \text{relacional1.tipo}, \text{relacional.tipo}) \\ \alpha_2 = \text{ampliar}(\text{relacional2.dir}, \text{relacional2.tipo}, \text{relacional.tipo}) \\ \text{add.quad}(\text{code}, \text{``}<\text{``}, \alpha_1, \alpha_2, \text{I}) \\ \text{add.quad}(\text{code}, \text{``goto'}, \text{``}, \text{II}) \\ \text{relacional} \rightarrow \text{expresion} \\ \text{expresion} \rightarrow \text{expresion.tipo} \\ \text{expresion.dir} = \text{expresion.dir} \\ \text{expresion} \rightarrow \text{expresion}_1 + \text{expresion}_2 \\ \text{expresion.dir} = \text{newTemp}() \\ \alpha_1 = \text{ampliar}(\text{expresion2.dir}, \text{expresion1.tipo}, \text{expresion.tipo}) \\ \alpha_2 = \text{ampliar}(\text{expresion2.dir}, \text{expresion2.tipo}, \text{expresion.tipo}) \\ \text{add.quad}(\text{code}, \text{``}+\text{``}, \alpha_1, \alpha_2, \text{expresion1.tipo}, \text{expresion2.tipo}) \\ \text{expresion} \rightarrow \text{expresion1} - \text{expresion2} \\ \text{expresion.dir} = \text{newTemp}() \\ \alpha_1 = \text{ampliar}(\text{expresion1.tipo}, \text{expresion2.tipo}, \text{expresion.tipo}) \\ \alpha_2 = \text{ampliar}(\text{expresion1.dir}, \text{expresion3.tipo}, \text{expresion.tipo}) \\ \alpha_2 = \text{ampliar}(\text{expresion2.dir}, \text{expresion3.tipo}, \text{expresion.tipo}) \\ \alpha_2 = \text{ampliar}(\text{expresion3.dir}, \text{expresion3.dir}, \text{expresion.tipo}) \\ \alpha_2 = \text{ampliar}(\text{expresion3.dir}, \text{expresion3.dir}, \text{expresion.tipo}) \\ \alpha_3 = \text{ampliar}(\text{expresion3.dir}, \text{expresion3.dir}, \text{expresion.tipo}) \\ \alpha_4 = \text{ampliar}(\text{expresion3.dir}, \text{expresion3.dir}, \text{expresion.tipo}) \\ \alpha_2 = \text{ampliar}(\text{expresion3.dir}, \text{expresion.dir}) \\ \alpha_3 = \text{expresion3.dir}, \alpha_3 = \text{expresion.dir}, \alpha_3 = \text{expresion3.dir}, \alpha_3 = \text{expresion.dir}) \\ \alpha_4 = \text{expresion3.dir}, \alpha_3 = expresi$		
$I = \operatorname{newIndex}(), I1 = \operatorname{newIndex}()$ $\operatorname{relacional.listtrue.add}(I)$ $\operatorname{relacional.listfalse.add}(I1)$ $\operatorname{relacional.lipo} = \operatorname{max}(\operatorname{relacional1.tipo}, \operatorname{relacional2.tipo})$ $\alpha_1 = \operatorname{ampliar}(\operatorname{relacional1.dir}, \operatorname{relacional1.tipo}, \operatorname{relacional.tipo})$ $\alpha_2 = \operatorname{ampliar}(\operatorname{relacional2.dir}, \operatorname{relacional2.tipo}, \operatorname{relacional.tipo})$ $\operatorname{add.quad}(\operatorname{code}, "<>", \alpha_1 , \alpha_2, I)$ $\operatorname{add.quad}(\operatorname{code}, "\operatorname{goto"}, -, -, 11)$ $\operatorname{relacional} \to \operatorname{expresion}$ $\operatorname{relacional.dir} = \operatorname{expresion.tipo}$ $\operatorname{relacional.dir} = \operatorname{expresion.tipo}$ $\operatorname{expresion} \to \operatorname{expresion1.tipo}, \operatorname{expresion2.tipo})$ $\operatorname{expresion.dir} = \operatorname{newTemp}()$ $\alpha_1 = \operatorname{ampliar}(\operatorname{expresion2.dir}, \operatorname{expresion1.tipo}, \operatorname{expresion.tipo})$ $\alpha_2 = \operatorname{ampliar}(\operatorname{expresion2.dir}, \operatorname{expresion2.tipo}, \operatorname{expresion.tipo})$ $\operatorname{add.quad}(\operatorname{code}, "+", \alpha_1, \alpha_2, \operatorname{expresion2.tipo})$ $\operatorname{expresion.dir} = \operatorname{newTemp}()$ $\alpha_1 = \operatorname{ampliar}(\operatorname{expresion1.tipo}, \operatorname{expresion2.tipo})$ $\operatorname{expresion.dir} = \operatorname{newTemp}()$ $\alpha_1 = \operatorname{ampliar}(\operatorname{expresion1.tipo}, \operatorname{expresion2.tipo})$ $\operatorname{expresion.dir} = \operatorname{newTemp}()$ $\alpha_1 = \operatorname{ampliar}(\operatorname{expresion1.tipo}, \operatorname{expresion1.tipo}, \operatorname{expresion.tipo})$ $\alpha_2 = \operatorname{ampliar}(\operatorname{expresion2.dir}, \operatorname{expresion2.tipo}, \operatorname{expresion.tipo})$ $\alpha_3 = \operatorname{ampliar}(\operatorname{expresion2.dir}, \operatorname{expresion2.tipo}, \operatorname{expresion.tipo})$ $\alpha_2 = \operatorname{ampliar}(\operatorname{expresion2.dir}, \operatorname{expresion2.tipo}, \operatorname{expresion.tipo})$ $\alpha_3 = \operatorname{ampliar}(\operatorname{expresion2.dir}, \operatorname{expresion2.tipo}, \operatorname{expresion.tipo})$ $\alpha_4 = \operatorname{ampliar}(\operatorname{expresion2.dir}, \operatorname{expresion2.tipo}, \operatorname{expresion.tipo})$ $\alpha_4 = \operatorname{ampliar}(\operatorname{expresion2.dir}, \operatorname{expresion2.tipo}, \operatorname{expresion.tipo})$ $\alpha_4 = \operatorname{ampliar}(\operatorname{expresion2.dir}, \operatorname{expresion2.tipo}, \operatorname{expresion2.tipo}, \operatorname{expresion2.tipo}, \operatorname{expresion3.tipo}, \operatorname{expresion3.tipo}, \operatorname$	$relacional \rightarrow relacional_1 \iff relacional_2$	
$\begin{array}{c} \text{relacional.list} \text{true.add}(I) \\ \text{relacional.listfalse.add}(II) \\ \text{relacional.tipo} = \max(\text{relacional1.tipo}, \text{ relacional2.tipo}) \\ \alpha_1 = \text{ampliar}(\text{relacional1.tipo}, \text{ relacional1.tipo}, \text{ relacional1.tipo}) \\ \alpha_2 = \text{ampliar}(\text{relacional2.dir}, \text{ relacional2.tipo}, \text{ relacional.tipo}) \\ \alpha_2 = \text{ampliar}(\text{relacional2.dir}, \text{ relacional2.tipo}, \text{ relacional.tipo}) \\ \text{add.quad}(\text{code}, "\text{sof}", \neg, \neg, \text{II}) \\ \text{relacional} \rightarrow \text{expresion} \\ \text{relacional.dir} = \text{expresion.tipo} \\ \text{relacional.dir} = \text{expresion.dir} \\ \text{expresion} \rightarrow \text{expresion}_1 + \text{expresion}_2 \\ \text{expresion.tipo} = \max(\text{expresion1.tipo}, \text{expresion.tipo}) \\ \text{expresion} \rightarrow \text{expresion}_1 - \text{expresion}_2 \\ \text{expresion} \rightarrow \text{expresion}_1 - \text{expresion}_2 \\ \text{expresion.tipo} = \max(\text{expresion1.tipo}, \text{expresion2.tipo}) \\ \text{expresion.dir} = \text{newTemp}() \\ \text{expresion.tipo} = \max(\text{expresion1.tipo}, \text{expresion2.tipo}) \\ \text{expresion.dir} = \text{newTemp}() \\ \text{expresion.tipo} = \max(\text{expresion1.tipo}, \text{expresion2.tipo}) \\ \text{expresion.dir} = \text{newTemp}() \\ \text{expresion.dir} = \text{newTemp}() \\ \text{expresion.dir} = \text{expresion1.tipo}, \text{expresion1.tipo}, \text{expresion.tipo}) \\ \text{expresion.dir} = \text{newTemp}() \\ \text{expresion.dir} = \text{newTemp}$		relacional.listfalse = newList()
$\begin{array}{c} \text{relacional.listfalse.add}(\overline{11}) \\ \text{relacional.tipo} & \max(\text{relacional1.tipo}, \text{relacional2.tipo}) \\ \alpha_1 & = \operatorname{ampliar}(\text{relacional1.dir}, \text{relacional1.tipo}, \text{relacional.tipo}) \\ \alpha_2 & = \operatorname{ampliar}(\text{relacional2.dir}, \text{relacional2.tipo}, \text{relacional.tipo}) \\ \alpha_2 & = \operatorname{ampliar}(\text{relacional2.dir}, \text{relacional2.tipo}, \text{relacional.tipo}) \\ \operatorname{add.quad}(\operatorname{code}, "<>", \alpha_1, \alpha_2, I) \\ \operatorname{add.quad}(\operatorname{code}, "goto", \neg, \neg, I1) \\ \\ \text{relacional} & \to \operatorname{expresion} \\ \text{expresion} & \to \operatorname{expresion.dir} \\ \text{expresion.dir} & = \operatorname{expresion.dir} \\ \text{expresion.dir} & = \operatorname{expresion1.tipo}, \operatorname{expresion2.tipo}) \\ \operatorname{expresion.dir} & = \operatorname{newTemp}() \\ \alpha_1 & = \operatorname{ampliar}(\operatorname{expresion1.dir}, \operatorname{expresion2.tipo}, \operatorname{expresion.tipo}) \\ \alpha_2 & = \operatorname{ampliar}(\operatorname{expresion2.dir}, \operatorname{expresion2.tipo}, \operatorname{expresion.tipo}) \\ \operatorname{add.quad}(\operatorname{code}, "+", \alpha_1, \alpha_2, \operatorname{expresion1.tipo}, \operatorname{expresion2.tipo}) \\ \operatorname{expresion.dir} & = \operatorname{newTemp}() \\ \alpha_1 & = \operatorname{ampliar}(\operatorname{expresion1.tipo}, \operatorname{expresion2.tipo}, \operatorname{expresion.tipo}) \\ \operatorname{expresion.dir} & = \operatorname{newTemp}() \\ \alpha_1 & = \operatorname{ampliar}(\operatorname{expresion1.tipo}, \operatorname{expresion1.tipo}, \operatorname{expresion.tipo}) \\ \operatorname{add.quad}(\operatorname{code}, "-", \alpha_1, \alpha_2, \operatorname{expresion.tipo}, \operatorname{expresion.tipo}) \\ \operatorname{add.quad}(\operatorname{code}, "-", \alpha_1, \alpha_2, \operatorname{expresion.dir}) \\ \text{add.quad}(\operatorname{code}, "-", \alpha_1, \alpha_2, \operatorname{expresion.dir}) \\ \text{add.quad}($		I = newIndex(), I1 = newIndex()
$\alpha_1 = \operatorname{ampliar}(\operatorname{relacional1.tipo}, \operatorname{relacional2.tipo})$ $\alpha_1 = \operatorname{ampliar}(\operatorname{relacional1.dir}, \operatorname{relacional1.tipo}, \operatorname{relacional1.tipo})$ $\alpha_2 = \operatorname{ampliar}(\operatorname{relacional2.dir}, \operatorname{relacional2.tipo}, \operatorname{relacional.tipo})$ $\alpha_2 = \operatorname{ampliar}(\operatorname{relacional2.dir}, \operatorname{relacional2.tipo}, \operatorname{relacional.tipo})$ $\operatorname{add_quad}(\operatorname{code}, "\operatorname{goto"}, -, -, \operatorname{II})$ $\operatorname{relacional} \to \operatorname{expresion}$ $\operatorname{relacional.tipo} = \operatorname{expresion.tipo}$ $\operatorname{relacional.dir} = \operatorname{expresion.dir}$ $\operatorname{expresion} \to \operatorname{expresion1.tipo}, \operatorname{expresion2.tipo})$ $\operatorname{expresion.tipo} = \operatorname{max}(\operatorname{expresion1.tipo}, \operatorname{expresion1.tipo}, \operatorname{expresion.tipo})$ $\alpha_1 = \operatorname{ampliar}(\operatorname{expresion2.dir}, \operatorname{expresion2.tipo}, \operatorname{expresion.tipo})$ $\alpha_2 = \operatorname{ampliar}(\operatorname{expresion2.dir}, \operatorname{expresion2.tipo}, \operatorname{expresion1.tipo}, \operatorname{expresion2.tipo})$ $\operatorname{expresion} \to \operatorname{expresion1} \to \operatorname{expresion1.tipo}, \operatorname{expresion2.tipo})$ $\operatorname{expresion.dir} = \operatorname{newTemp}()$ $\alpha_1 = \operatorname{ampliar}(\operatorname{expresion1.dir}, \operatorname{expresion2.tipo}, \operatorname{expresion.tipo})$ $\alpha_2 = \operatorname{ampliar}(\operatorname{expresion2.dir}, \operatorname{expresion2.tipo}, \operatorname{expresion.tipo})$ $\alpha_2 = \operatorname{ampliar}(\operatorname{expresion2.dir}, \operatorname{expresion2.tipo}, \operatorname{expresion.tipo})$ $\alpha_2 = \operatorname{ampliar}(\operatorname{expresion2.dir}, \operatorname{expresion2.tipo}, \operatorname{expresion.tipo})$ $\alpha_3 = \operatorname{ampliar}(\operatorname{expresion2.dir}, \operatorname{expresion3.tipo}, \operatorname{expresion.tipo})$ $\alpha_4 = \operatorname{ampliar}(\operatorname{expresion3.dir}, \operatorname{expresion3.tipo}, \operatorname{expresion.tipo})$ $\alpha_4 = \operatorname{ampliar}(\operatorname{expresion3.dir}, \operatorname{expresion3.dir})$		ho = relacional.listtrue.add(I)
$\alpha_1 = \text{ampliar}(\text{relacional}_1.\text{dir}, \text{ relacional}_1.\text{tipo}, \text{ relacional}_1.\text{tipo})$ $\alpha_2 = \text{ampliar}(\text{relacional}_2.\text{dir}, \text{ relacional}_2.\text{tipo}, \text{ relacional}_1.\text{tipo})$ $\text{add_quad}(\text{code}, "<>", \alpha_1, \alpha_2, I)$ $\text{add_quad}(\text{code}, "\text{goto"}, -, -, I1)$ $\text{relacional} \rightarrow \text{expresion}$ $\text{relacional.tipo} = \text{expresion.tipo}$ $\text{relacional.dir} = \text{expresion.dir}$ $\text{expresion} \rightarrow \text{expresion}_1 + \text{expresion}_2$ $\text{expresion.tipo} = \text{max}(\text{expresion}_1.\text{tipo}, \text{ expresion}_2.\text{tipo})$ $\text{expresion.dir} = \text{newTemp}()$ $\alpha_1 = \text{ampliar}(\text{expresion}_1.\text{dir}, \text{ expresion}_1.\text{tipo}, \text{ expresion.tipo})$ $\alpha_2 = \text{ampliar}(\text{expresion}_2.\text{dir}, \text{ expresion}_2.\text{tipo}, \text{ expresion.tipo})$ $\alpha_2 = \text{ampliar}(\text{expresion}_1.\text{tipo}, \text{ expresion}_2.\text{tipo})$ $\text{expresion.tipo} = \text{max}(\text{expresion}_1.\text{tipo}, \text{ expresion}_2.\text{tipo})$ $\text{expresion.dir} = \text{newTemp}()$ $\alpha_1 = \text{ampliar}(\text{expresion}_1.\text{dir}, \text{ expresion}_1.\text{tipo}, \text{ expresion.tipo})$ $\alpha_2 = \text{ampliar}(\text{expresion}_1.\text{dir}, \text{ expresion}_1.\text{tipo}, \text{ expresion.tipo})$ $\alpha_2 = \text{ampliar}(\text{expresion}_2.\text{dir}, \text{ expresion}_1.\text{tipo}, \text{ expresion.tipo})$ $\alpha_2 = \text{ampliar}(\text{expresion}_2.\text{dir}, \text{ expresion}_1.\text{tipo}, \text{ expresion.tipo})$ $\alpha_1 = \text{ampliar}(\text{expresion}_2.\text{dir}, \text{ expresion}_1.\text{tipo}, \text{ expresion.tipo})$ $\alpha_2 = \text{ampliar}(\text{expresion}_2.\text{dir}, \text{ expresion}_1.\text{tipo}, \text{ expresion.tipo})$ $\alpha_2 = \text{ampliar}(\text{expresion}_2.\text{dir}, \text{ expresion}_1.\text{tipo}, \text{ expresion.tipo})$ $\alpha_1 = \text{ampliar}(\text{expresion}_2.\text{dir}, \text{ expresion}_1.\text{tipo}, \text{ expresion.tipo})$ $\alpha_2 = \text{ampliar}(\text{expresion}_2.\text{dir}, \text{ expresion}_2.\text{tipo}, \text{ expresion.tipo})$ $\alpha_1 = \text{ampliar}(\text{expresion}_2.\text{dir}, \text{ expresion}_2.\text{tipo}, \text{ expresion.tipo})$ $\alpha_2 = \text{expresion}_2.\text{dir}, \text{ expresion}_2.\text{dir}, \text{ expresion.tipo}$ $\alpha_1 = \text{expresion}_2.\text{dir}, \text{ expresion}_2.\text{dir}, \text{ expresion.tipo}$ $\alpha_2 = \text{expresion}_2.\text{dir}, \text{ expresion}_2.\text{dir}, \text{ expresion}_2.\text{dir}, \text{ expresion.tipo}$ $\alpha_1 = \text{expresion}_2.dir$		m relacional.listfalse.add(I1)
$\alpha_2 = \text{ampliar}(\text{relacional}_2.\text{dir}, \text{ relacional}_2.\text{tipo}, \text{ relacional}_1.\text{tipo})$ $\text{add_quad}(\text{code}, "<>", \alpha_1, \alpha_2, I)$ $\text{add_quad}(\text{code}, "\text{goto"}, \neg, \neg, I1)$ $\text{relacional} \rightarrow \text{expresion}$ $\text{relacional.tipo} = \text{expresion.tipo}$ $\text{relacional.dir} = \text{expresion.dir}$ $\text{expresion} \rightarrow \text{expresion}_1 + \text{expresion}_2$ $\text{expresion.tipo} = \text{max}(\text{expresion}_1.\text{tipo}, \text{expresion}_2.\text{tipo})$ $\text{expresion.dir} = \text{newTemp}()$ $\alpha_1 = \text{ampliar}(\text{expresion}_2.\text{dir}, \text{expresion}_2.\text{tipo}, \text{expresion}.\text{tipo})$ $\alpha_2 = \text{ampliar}(\text{expresion}_2.\text{dir}, \text{expresion}_2.\text{tipo}, \text{expresion}.\text{tipo})$ $\text{add_quad}(\text{code}, "+", \alpha_1, \alpha_2, \text{expresion}_2.\text{tipo})$ $\text{expresion.dir} = \text{newTemp}()$ $\alpha_1 = \text{ampliar}(\text{expresion}_1.\text{dir}, \text{expresion}_1.\text{tipo}, \text{expresion}.\text{tipo})$ $\text{expresion.dir} = \text{newTemp}()$ $\alpha_1 = \text{ampliar}(\text{expresion}_1.\text{dir}, \text{expresion}_1.\text{tipo}, \text{expresion}.\text{tipo})$ $\alpha_2 = \text{ampliar}(\text{expresion}_2.\text{dir}, \text{expresion}_2.\text{tipo}, \text{expresion}.\text{tipo})$ $\alpha_2 = \text{ampliar}(\text{expresion}_2.\text{dir}, \text{expresion}_2.\text{tipo}, \text{expresion}.\text{tipo})$ $\text{add_quad}(\text{code}, "-", \alpha_1, \alpha_2, \text{expresion}.\text{dir})$		$relacional.tipo = max(relacional1.tipo, relacional_2.tipo)$
$ \begin{array}{c} \operatorname{add_quad(code,\ "<>",\alpha_1\ ,\alpha_2,\ I)} \\ \operatorname{add_quad(code,\ "goto",\ -,\ -,\ I1)} \\ \\ \operatorname{relacional} \to \operatorname{expresion} \\ \\ \operatorname{expresion} \to \operatorname{expresion1.tipo} = \operatorname{expresion.tipo} \\ \operatorname{relacional.dir} = \operatorname{expresion.dir} \\ \\ \operatorname{expresion} \to \operatorname{expresion1} + \operatorname{expresion2} \\ \\ \operatorname{expresion.tipo} = \operatorname{max}(\operatorname{expresion1.tipo}, \operatorname{expresion2.tipo}) \\ \operatorname{expresion.dir} = \operatorname{newTemp}() \\ \alpha_1 = \operatorname{ampliar}(\operatorname{expresion2.dir}, \operatorname{expresion2.tipo}, \operatorname{expresion.tipo}) \\ \alpha_2 = \operatorname{ampliar}(\operatorname{expresion2.dir}, \operatorname{expresion2.tipo}, \operatorname{expresion.tipo}) \\ \operatorname{add_quad}(\operatorname{code},\ "+",\alpha_1\ ,\alpha_2, \operatorname{expresion3.tipo}, \operatorname{expresion2.tipo}) \\ \operatorname{expresion.dir} = \operatorname{newTemp}() \\ \alpha_1 = \operatorname{ampliar}(\operatorname{expresion1.tipo}, \operatorname{expresion1.tipo}, \operatorname{expresion.tipo}) \\ \operatorname{expresion.dir} = \operatorname{newTemp}() \\ \alpha_2 = \operatorname{ampliar}(\operatorname{expresion2.dir}, \operatorname{expresion2.tipo}, \operatorname{expresion.tipo}) \\ \operatorname{add_quad}(\operatorname{code},\ "-",\alpha_1\ ,\alpha_2, \operatorname{expresion.dir}) \\ \operatorname{add_quad}(\operatorname{code},\ "-",\alpha_1\ ,\alpha_2, \operatorname{expresion.dir}) \\ \end{array}$		$\alpha_1 = \text{ampliar}(\text{relacional}_1.\text{dir}, \text{relacional}_1.\text{tipo}, \text{relacional}.\text{tipo})$
$ \begin{array}{c} \operatorname{add_quad}(\operatorname{code}, "\operatorname{goto"}, \text{-}, \text{-}, \operatorname{I1}) \\ \operatorname{relacional} \to \operatorname{expresion} \\ \operatorname{relacional.tipo} = \operatorname{expresion.tipo} \\ \operatorname{relacional.dir} = \operatorname{expresion.dir} \\ \operatorname{expresion} \to \operatorname{expresion}_1 + \operatorname{expresion}_2 \\ \operatorname{expresion.tipo} = \operatorname{max}(\operatorname{expresion1.tipo}, \operatorname{expresion2.tipo}) \\ \operatorname{expresion.dir} = \operatorname{newTemp}() \\ \alpha_1 = \operatorname{ampliar}(\operatorname{expresion2.dir}, \operatorname{expresion2.tipo}, \operatorname{expresion.tipo}) \\ \alpha_2 = \operatorname{ampliar}(\operatorname{expresion2.dir}, \operatorname{expresion2.tipo}, \operatorname{expresion.tipo}) \\ \operatorname{add_quad}(\operatorname{code}, "+", \alpha_1, \alpha_2, \operatorname{expresion2.tipo}, \operatorname{expresion2.tipo}) \\ \operatorname{expresion} \to \operatorname{expresion1.tipo} = \operatorname{max}(\operatorname{expresion1.tipo}, \operatorname{expresion2.tipo}) \\ \operatorname{expresion.dir} = \operatorname{newTemp}() \\ \alpha_1 = \operatorname{ampliar}(\operatorname{expresion2.dir}, \operatorname{expresion3.tipo}, \operatorname{expresion.tipo}) \\ \alpha_2 = \operatorname{ampliar}(\operatorname{expresion2.dir}, \operatorname{expresion3.tipo}, \operatorname{expresion.tipo}) \\ \operatorname{add_quad}(\operatorname{code}, "-", \alpha_1, \alpha_2, \operatorname{expresion.dir}) \\ \end{array}$		$\alpha_2 = \text{ampliar}(\text{relacional}_2.\text{dir}, \text{relacional}_2.\text{tipo}, \text{relacional}.\text{tipo})$
$ \begin{array}{lll} \text{relacional} \rightarrow \text{expresion} & \text{relacional.tipo} = \text{expresion.tipo} \\ \text{relacional.dir} = \text{expresion.dir} \\ \text{expresion} \rightarrow \text{expresion}_1 + \text{expresion}_2 & \text{expresion.tipo} = \text{max}(\text{expresion1.tipo}, \text{expresion2.tipo}) \\ \text{expresion.dir} = \text{newTemp}() \\ \alpha_1 = \text{ampliar}(\text{expresion2.dir}, \text{expresion2.tipo}, \text{expresion.tipo}) \\ \alpha_2 = \text{ampliar}(\text{expresion2.dir}, \text{expresion2.tipo}, \text{expresion.tipo}) \\ \text{add_quad}(\text{code}, "+", \alpha_1, \alpha_2, \text{expresion.dir}) \\ \text{expresion} \rightarrow \text{expresion}_1 - \text{expresion2} \\ \text{expresion.dir} = \text{max}(\text{expresion1.tipo}, \text{expresion2.tipo}) \\ \text{expresion.dir} = \text{newTemp}() \\ \alpha_1 = \text{ampliar}(\text{expresion1.dir}, \text{expresion1.tipo}, \text{expresion.tipo}) \\ \alpha_2 = \text{ampliar}(\text{expresion2.dir}, \text{expresion2.tipo}, \text{expresion.tipo}) \\ \text{add_quad}(\text{code}, "-", \alpha_1, \alpha_2, \text{expresion.dir}) \\ \text{add_quad}(\text{code}, "-", \alpha_1, \alpha_2, \text{expresion.dir}) \\ \end{array}$		add_quad(code, " $<>$ ", α_1 , α_2 , I)
$ \begin{array}{c} \operatorname{relacional.dir} = \operatorname{expresion.dir} \\ \operatorname{expresion} \to \operatorname{expresion}_1 + \operatorname{expresion}_2 \\ \operatorname{expresion.tipo} = \operatorname{max}(\operatorname{expresion1.tipo}, \operatorname{expresion2.tipo}) \\ \operatorname{expresion.dir} = \operatorname{newTemp}() \\ \alpha_1 = \operatorname{ampliar}(\operatorname{expresion}_1.\operatorname{dir}, \operatorname{expresion1.tipo}, \operatorname{expresion.tipo}) \\ \alpha_2 = \operatorname{ampliar}(\operatorname{expresion2.dir}, \operatorname{expresion2.tipo}, \operatorname{expresion.tipo}) \\ \operatorname{add_quad}(\operatorname{code}, "+", \alpha_1, \alpha_2, \operatorname{expresion.dir}) \\ \operatorname{expresion} \to \operatorname{expresion1} - \operatorname{expresion2} \\ \operatorname{expresion.dir} = \operatorname{newTemp}() \\ \alpha_1 = \operatorname{ampliar}(\operatorname{expresion1.tipo}, \operatorname{expresion.tipo}, \operatorname{expresion.tipo}) \\ \alpha_2 = \operatorname{ampliar}(\operatorname{expresion2.dir}, \operatorname{expresion2.tipo}, \operatorname{expresion.tipo}) \\ \alpha_2 = \operatorname{ampliar}(\operatorname{expresion2.dir}, \operatorname{expresion2.tipo}, \operatorname{expresion.tipo}) \\ \operatorname{add_quad}(\operatorname{code}, "-", \alpha_1, \alpha_2, \operatorname{expresion.dir}) \\ \end{array}$		add_quad(code, "goto", -, -, I1)
$\begin{array}{ll} \operatorname{expresion} \to \operatorname{expresion}_1 + \operatorname{expresion}_2 & \operatorname{expresion.tipo} = \operatorname{max}(\operatorname{expresion}_1.\operatorname{tipo}, \operatorname{expresion}_2.\operatorname{tipo}) \\ \operatorname{expresion.dir} = \operatorname{newTemp}() \\ \alpha_1 = \operatorname{ampliar}(\operatorname{expresion}_1.\operatorname{dir}, \operatorname{expresion}_1.\operatorname{tipo}, \operatorname{expresion.tipo}) \\ \alpha_2 = \operatorname{ampliar}(\operatorname{expresion}_2.\operatorname{dir}, \operatorname{expresion}_2.\operatorname{tipo}, \operatorname{expresion.tipo}) \\ \operatorname{add_quad}(\operatorname{code}, "+", \alpha_1, \alpha_2, \operatorname{expresion.dir}) \\ \operatorname{expresion.tipo} = \operatorname{max}(\operatorname{expresion}_1.\operatorname{tipo}, \operatorname{expresion}_2.\operatorname{tipo}) \\ \operatorname{expresion.dir} = \operatorname{newTemp}() \\ \alpha_1 = \operatorname{ampliar}(\operatorname{expresion}_1.\operatorname{dir}, \operatorname{expresion}_1.\operatorname{tipo}, \operatorname{expresion.tipo}) \\ \alpha_2 = \operatorname{ampliar}(\operatorname{expresion}_2.\operatorname{dir}, \operatorname{expresion}_2.\operatorname{tipo}, \operatorname{expresion.tipo}) \\ \operatorname{add_quad}(\operatorname{code}, "-", \alpha_1, \alpha_2, \operatorname{expresion.dir}) \\ \end{array}$	$relacional \rightarrow expresion$	relacional.tipo = expresion.tipo
$\begin{array}{c} \text{expresion.dir} = \text{newTemp}() \\ \alpha_1 = \text{ampliar}(\text{expresion}_1.\text{dir}, \text{expresion}_1.\text{tipo}, \text{expresion.tipo}) \\ \alpha_2 = \text{ampliar}(\text{expresion}_2.\text{dir}, \text{expresion}_2.\text{tipo}, \text{expresion.tipo}) \\ \text{add_quad}(\text{code}, "+", \alpha_1 , \alpha_2, \text{expresion.dir}) \\ \text{expresion} \rightarrow \text{expresion}_1 - \text{expresion}_2 \\ \text{expresion.tipo} = \text{max}(\text{expresion}_1.\text{tipo}, \text{expresion}_2.\text{tipo}) \\ \text{expresion.dir} = \text{newTemp}() \\ \alpha_1 = \text{ampliar}(\text{expresion}_1.\text{dir}, \text{expresion}_1.\text{tipo}, \text{expresion.tipo}) \\ \alpha_2 = \text{ampliar}(\text{expresion}_2.\text{dir}, \text{expresion}_2.\text{tipo}, \text{expresion.tipo}) \\ \text{add_quad}(\text{code}, "-", \alpha_1 , \alpha_2, \text{expresion.dir}) \\ \end{array}$		relacional.dir = expresion.dir
$\alpha_1 = \text{ampliar}(\text{expresion}_1.\text{dir}, \text{ expresion}_1.\text{tipo}, \text{ expresion.tipo})$ $\alpha_2 = \text{ampliar}(\text{expresion}_2.\text{dir}, \text{ expresion}_2.\text{tipo}, \text{ expresion.tipo})$ $\text{add_quad}(\text{code}, "+", \alpha_1, \alpha_2, \text{ expresion.dir})$ $\text{expresion} \rightarrow \text{expresion}_1 - \text{expresion}_2$ $\text{expresion.tipo} = \text{max}(\text{expresion}_1.\text{tipo}, \text{ expresion}_2.\text{tipo})$ $\text{expresion.dir} = \text{newTemp}()$ $\alpha_1 = \text{ampliar}(\text{expresion}_1.\text{dir}, \text{ expresion}_1.\text{tipo}, \text{ expresion.tipo})$ $\alpha_2 = \text{ampliar}(\text{expresion}_2.\text{dir}, \text{ expresion}_2.\text{tipo}, \text{ expresion.tipo})$ $\alpha_2 = \text{ampliar}(\text{expresion}_2.\text{dir}, \text{ expresion}_2.\text{tipo}, \text{ expresion.tipo})$ $\alpha_1 = \text{ampliar}(\text{expresion}_2.\text{dir}, \text{ expresion}_2.\text{tipo}, \text{ expresion.tipo})$ $\alpha_2 = \text{ampliar}(\text{expresion}_2.\text{dir}, \text{ expresion}_2.\text{tipo}, \text{ expresion.tipo})$ $\alpha_1 = \text{ampliar}(\text{expresion}_2.\text{dir}, \text{ expresion}_2.\text{tipo}, \text{ expresion.tipo})$ $\alpha_2 = \text{ampliar}(\text{expresion}_2.\text{dir}, \text{ expresion}_2.\text{tipo}, \text{ expresion.tipo})$ $\alpha_3 = \text{expresion}_3.\text{dir}, \text{ expresion}_3.\text{dir}, \text{ expresion.tipo}$	$expresion \rightarrow expresion_1 + expresion_2$	
$\alpha_2 = \text{ampliar}(\text{expresion}_2.\text{dir}, \text{ expresion}_2.\text{tipo}, \text{ expresion.tipo})$ $\text{add_quad}(\text{code}, "+", \alpha_1, \alpha_2, \text{ expresion.dir})$ $\text{expresion} \rightarrow \text{expresion}_1 - \text{expresion}_2$ $\text{expresion.tipo} = \text{max}(\text{expresion}_1.\text{tipo}, \text{ expresion.tipo})$ $\text{expresion.dir} = \text{newTemp}()$ $\alpha_1 = \text{ampliar}(\text{expresion}_1.\text{dir}, \text{ expresion}_1.\text{tipo}, \text{ expresion.tipo})$ $\alpha_2 = \text{ampliar}(\text{expresion}_2.\text{dir}, \text{ expresion}_2.\text{tipo}, \text{ expresion.tipo})$ $\alpha_2 = \text{ampliar}(\text{expresion}_2.\text{dir}, \text{ expresion}_2.\text{tipo}, \text{ expresion.tipo})$ $\alpha_1 = \text{ampliar}(\text{expresion}_2.\text{dir}, \text{ expresion}_2.\text{tipo}, \text{ expresion.tipo})$ $\alpha_2 = \text{ampliar}(\text{expresion}_2.\text{dir}, \text{ expresion}_2.\text{tipo}, \text{ expresion.tipo})$ $\alpha_1 = \text{ampliar}(\text{expresion}_2.\text{dir}, \text{ expresion}_2.\text{tipo}, \text{ expresion.tipo})$ $\alpha_2 = \text{ampliar}(\text{expresion}_2.\text{dir}, \text{ expresion}_3.\text{dir}, \text{ expresion.tipo})$ $\alpha_3 = \text{ampliar}(\text{expresion}_3.\text{dir}, \text{ expresion.tipo})$		
$ add_quad(code, "+", \alpha_1, \alpha_2, expresion.dir) $ $ expresion \rightarrow expresion_1 - expresion_2 $ $ expresion.tipo = max(expresion_1.tipo, expresion_2.tipo) $ $ expresion.dir = newTemp() $ $ \alpha_1 = ampliar(expresion_1.dir, expresion_1.tipo, expresion.tipo) $ $ \alpha_2 = ampliar(expresion_2.dir, expresion_2.tipo, expresion.tipo) $ $ add_quad(code, "-", \alpha_1, \alpha_2, expresion.dir) $		$\alpha_1 = \text{ampliar}(\text{expresion}_1.\text{dir}, \text{expresion}_1.\text{tipo}, \text{expresion}.\text{tipo})$
expresion \rightarrow expresion ₁ - expresion ₂ expresion.tipo = max(expresion1.tipo, expresion ₂ .tipo) expresion.dir = newTemp() $\alpha_1 = \text{ampliar}(\text{expresion}_1.\text{dir}, \text{expresion}_1.\text{tipo}, \text{expresion}.\text{tipo})$ $\alpha_2 = \text{ampliar}(\text{expresion}_2.\text{dir}, \text{expresion}_2.\text{tipo}, \text{expresion}.\text{tipo})$ add_quad(code, "-", α_1 , α_2 , expresion.dir)		
expresion.dir = newTemp() $\alpha_1 = \text{ampliar}(\text{expresion}_1.\text{dir}, \text{expresion}_1.\text{tipo}, \text{expresion}.\text{tipo})$ $\alpha_2 = \text{ampliar}(\text{expresion}_2.\text{dir}, \text{expresion}_2.\text{tipo}, \text{expresion}.\text{tipo})$ $\text{add_quad}(\text{code}, "-", \alpha_1, \alpha_2, \text{expresion}.\text{dir})$		
$\alpha_1 = \text{ampliar}(\text{expresion}_1.\text{dir}, \text{expresion}_1.\text{tipo}, \text{expresion}.\text{tipo})$ $\alpha_2 = \text{ampliar}(\text{expresion}_2.\text{dir}, \text{expresion}_2.\text{tipo}, \text{expresion}.\text{tipo})$ $\text{add_quad}(\text{code}, "-", \alpha_1, \alpha_2, \text{expresion}.\text{dir})$	$expresion \rightarrow expresion_1 - expresion_2$	
$\alpha_2 = \text{ampliar}(\text{expresion}_2.\text{dir}, \text{expresion}_2.\text{tipo}, \text{expresion.tipo})$ add_quad(code, "-", α_1 , α_2 , expresion.dir)		
$add_quad(code, "-", \alpha_1, \alpha_2, expresion.dir)$		
$ add_quad(code, "-", \alpha_1, \alpha_2, expresion.dir) $ $ expresion \rightarrow expresion_1 * expresion_2 $ $ expresion.tipo = max(expresion1.tipo, expresion_2.tipo) $		
$\exp \operatorname{resion} \to \operatorname{expresion}_1 * \operatorname{expresion}_2 = \max(\operatorname{expresion}_1 \operatorname{tipo}, \operatorname{expresion}_2 \operatorname{tipo})$	<u> </u>	add_quad(code, "-", α_1 , α_2 , expresion.dir)
	$expresion \rightarrow expresion_1 * expresion_2$	
expresion.dir = newTemp()		1
$\alpha_1 = \text{ampliar}(\text{expresion}_1.\text{dir}, \text{expresion}_1.\text{tipo}, \text{expresion}.\text{tipo})$		
$\alpha_2 = \text{ampliar}(\text{expresion}_2.\text{dir}, \text{expresion}_2.\text{tipo}, \text{expresion.tipo})$		
add_quad(code, "*", α_1 , α_2 , expresion.dir)		
$expresion \rightarrow expresion_1 / expresion_2$ $expresion.tipo = max(expresion1.tipo, expresion_2.tipo)$	$expresion \rightarrow expresion_1 / expresion_2$	
expresion.dir = newTemp()		
$\alpha_1 = \text{ampliar}(\text{expresion}_1.\text{dir}, \text{expresion}_1.\text{tipo}, \text{expresion}.\text{tipo})$		
$\alpha_2 = \text{ampliar}(\text{expresion}_2.\text{dir}, \text{expresion}_2.\text{tipo}, \text{expresion}.\text{tipo})$		
add_quad(code, "/", α_1 , α_2 , expresion.dir)	_	add_quad(code, "/", α_1 , α_2 , expresion.dir)

PRODUCCIÓN	REGLAS SEMÁNTICAS
expresion \rightarrow expresion ₁ % expresion ₂	expresion.tipo = max(expresion1.tipo, expresion2.tipo) expresion.dir = newTemp() α_1 = ampliar(expresion1.dir, expresion1.tipo, expresion.tipo) α_2 = ampliar(expresion2.dir, expresion2.tipo, expresion.tipo)
$expresion \rightarrow (expresion_1)$	add_quad(code, "%", α_1 , α_2 , expresion.dir) expresion.dir = expresion_1.dir expresion.tipo = expresion_1.tipo
expresion \rightarrow variable	epxreson.dir = newTemp() expresion.tipo = variable.tipo add_quad(code, "*", variable.base[variable.dir], -, expresion.dir)
$expresion \rightarrow num$	expresoin.tipo = num.tipo expresion.dir = num.val
$expresion \rightarrow cadena$	expresion.tipo = cadena expresion.dir = TablaDeCadenas.add(cadena)
$ ext{expresion} o ext{caracter}$	expresion.tipo = caracter expresion.dir = TablaDeCadenas.add(caracter)
$\operatorname{expresion} \to \operatorname{\mathbf{id}}(\operatorname{parametros})$	si StackTS.getFondo().getId(id.lexval) ≠ -1 entonces si StackTS.getFondo().getVar(id.lexval) = "func" entonces lista = StackTs.getFondo().getArgs(id.lexval) si lista.getTam() ≠ parametros.getTam() entonces Error(El numero de argumentos no coincide) fin para i=0, i < parametros.lista.getTam(), 1 hacer si parametros[i] ≠ lsita[i] entonces Error("El tipo de los parametros no coindide") fin fin expresion.dir = newTemp() expresion.tipo = StackTs.getFondo().getTipo(id.lexval) add_quad(code, "=", "call", id.lexval, expresion.dir) en otro caso Error("El identificador no ha sido declarado") fin
$variable \rightarrow arreglo$	variable.dir = arreglo.dir variable.base = arreglo.base variable.tipo = arreglo.tipo
$ ext{variable} o ext{id}_1. ext{id}_2$	si $StackTS.getFondo().getId(id.lexval) \neq -1$ entonces $t = StackTS.getFondo().getTipo(id.lexval)$ $t1 = StackTT.getFondo().getTipo(t)$ $si t1 = "registro"$ entonces $tipoBase = StackTT.getFondo().getTipoBase(t)$ $si tiposBase.getId(id_2) \neq -1$ entonces $variable.tipo = tipoBase.getType(id_2)$ $variable.dir = id_2$ $variable.base = id_1$ $en otro caso$ $ Error("el id no existe en la estructura)$ fin $en otro caso$ $ Error("El id no es una estructura")$ fin $en otro caso$ $ Error("El id no es una estructura")$ fin $en otro caso$ $ Error("El identificador no ha sido declarado")$
	fin

PRODUCCIÓN	REGLAS SEMÁNTICAS
$\operatorname{arreglo} o \operatorname{id}[\operatorname{expresion}]$ $\operatorname{arreglo} o \operatorname{arreglo}_1 [\operatorname{expresion}]$	si StackTS.getCima().getId(id.lexval)≠-1 entonces t = StackTS.getCima().getTipo(id.lexval) si StackTT.getCima().getTipo(t) == "array" entonces si expresion.tipo = ent entonces arreglo.base = id.lexval arreglo.tipo = StackTT.getCima().getTipoBase(t) arreglo.dir = newTemp() add_quad(code, "*", expersion.dir, arreglo.tam, arreglo.dir) en otro caso Error(La expresión para un índice debe ser de tipo entero) fin en otro caso Error(El identificador no es un arreglo) fin fin en otro caso Error(El idendificador no ha sido declarado) fin si StackTT.getCima().getTipoBase(arreglo₁.tipo)="array" entonces arreglo.base = arreglo₁.base arreglo.base = arreglo₁.base arreglo.tam = StackTT.getCima().getTipoBase(arreglo₁.tipo) temp = newTemp() arreglo.dir = newTemp() add_quad(code, "*", expersion.dir, arreglo.tam, temp) add_quad(code, "+", arreglo₁.dir, temp, arreglo.dir) en otro caso Error(El arreglo no tiene tantas dimensiones) fin fin en otro caso Error(El arreglo no tiene tantas dimensiones)
$parametros \rightarrow lista_param$	parametros.lista = lista_param.lista
parametros $\rightarrow \varepsilon$	parametros.lista = nulo
$lista_param \rightarrow lista_param_1$, expresion	$lista_param.lista = lista_param_1.lista$
	lista_param.lista.add(param.tipo)
	add_quad(code, "param", expresion.dir, -, -)
$lista_param \rightarrow expresion$	lista_param.lista = newListaParam()
	lista_param.lista.add(expresion.tipo)
	add_quad(code, "param", expresion.dir, -, -)