Compiladores 2021-1 Prof. Adrián Ulises Mercado Martínez Provecto Final Coronel Ruiz Aldair García Pérez Adrián Toporek Coca Eric 31512683-1 31513122-4 31428498-7 Introducción. El diseño del front-end para un compilador cubre las siguientes fases: Diseño de Expresiones regulares que definen la gramática del lenguaje. Amálisis Léxico y generación de tokens del lenguaje. Amálisis Mináctico. Amálisis Semántico. Generación de código intermedio. A partir del material proporcionado por el Profesor (la gramática proporcionada a continuación) se pudo ir construyendo paso a paso lo que consta el mismo front-end a partir de los contenidos vistos en el curso de Compiladores. PRODUCCIÓN programa — declaraciones funciones declaraciones — tipo lista var ; declaraciones | a programo — declaraciones funciones | e declaraciones | e declaraciones | e declaraciones | e declaraciones | e figo = boildo compused o | e figo = boildo | e figo = boil Análisis Léxico Primero, se tienen que identificar todos los tokens de programa, es decir, los símbolos terminales que fueron resaltados en la gramática proporcionada y darles algún valor numérico que lo identifique como token, siendo los tokens definidos como sigue: PCOMA 1002 //; CIZO 1003 // [CDER 1004 //] INT 1005 // int FLOAT 1006 // float NUM 1007 // double ID 1008 // id SS 1028 PIZO 1009 // PDER 1010 // PDER 1010 // CHAR 1011 // char DOUBLE 1012 // double VOID 1013 // void IF 1014 // if ELSE 1015 // else DO 1016 // do WHILE 1017 // while BREAK 1018 // break SWITCH 1019 // switch DEFAULT 1020 // default CASE 1021 // case LT 1034 V 1035 / D 1036 / T 1037 / TR 1035 // cadena DDOT 1036 // : USIG 1037 // = DR 1039 // || KEY 1040 // { KKEY 1041 // } RUE 1042 ALSE 1043 FUNC 1044 RETURN 1045 PRINT 1046 SCAN 1047 IN 9001 // Para la implementación del análisis léxico, primero fue definida una clase token, que guarda la clase y el valor de la expresión según sea leida, y nos respaldamos en lexer que según sea lo que se lea puede generar un token, ignorarlo (whitespes y comentarios) o mandar un error en caso de que no se proporcione algo definido por la gramática. Los detalles de implementación sobre el lexer fueron cubiertos en la práctica 5. Análisis Sintáctico Diseño de expresiones regulares. A partir de la gramática dada, para simplificar las expresiones regulares por en typos más sencillos, se escogió en orden una letra del alfabeto para la asignación. Quedando de la siguiente forma: declaraciones

tipo

	básico compuesto		D									
	compuesto		_									
		-	E									
	lista_var	=	F									
	funciones	=	G									
	argumentos	=	H									
	lista_args	=	I									
	bloque	_	I									
	bioque		,									
	instrucciones	=	K									
	sentencia	=	L									
	casos	=	M									
	6360	-	N									
	caso	-										
	predeterminado	=	0									
	parte_izq	=	P									
	bool	_	Q									
	5001	-	Q									
	comb	=	R									
	igualdad	=	S									
	rel	=	T									
	exp	-	U									
	term		v									
		-										
	unario	=	W									
	factor	=	X									
	parámetros		Y									
	parametros											
	lista_param	=	Z									
	localización	=	AA									
Por lo tanto, la gramática qui	uedaría redefinida cómo sigue:											
sumo, m promatica qu												
- 41												
Gramática:												
	A ->	B G										
	B ->	C F;B	3									
	C ->	DE										
	D ->	int	float	char	double	void						
	E ->	[numero] E	3									
	F ->	F, id	id									
			ıa									
	G ->	func C id(H) J G		3								
	H ->	I	3									
	I->	id	I, Cid	3								
		(1,010	3								
	J ->	{BK}										
	K ->	KL	L									
	L ->	P = Q;	if(Q) L	if(Q) L else L	while(0) L	do L while(Q) break;	J					
		return U;	switch(Q) {M}	print U;	scan P	return;	- '					
	M ->	return 0,	0		Scall r	return,						
		N M	0	3								
	N ->	case numero: K										
	0 ->	default: K										
	P ->	id AA	id									
	Q->											
	` `	QIIR	R									
	R ->	R && S	S									
	R ->	R && S	S	Т								
	R -> S ->	R && S S'== T	S S!= T		11511							
	R -> S -> T ->	R && S S'== T U < U	S S!= T U <= U	U >= U	U > U	U						
	R -> S -> T -> U ->	R && S S'== T U < U U + V	S S != T U <= U U - V	U >= U V		U						
	R -> S -> T -> U -> V ->	R && S S'== T U < U U + V V * W	S S!= T U <= U U - V V/W	U >= U	U > U W	U						
	R -> S -> T -> U -> V ->	R && S S'== T U < U U + V V * W	S S!= T U <= U U - V V/W	U >= U V V % W		U						
	R -> S -> T -> U -> V ->	R && S S'== T U < U U + V V * W !W	S S != T U <= U U - V V/W W	U >= U V V % W X	w							
	R -> S -> T -> U -> V -> W -> X ->	R && S S'== T U < U U + V V * W (Q)	S S!= T U <= U U - V V/W -W id X'	U >= U V V % W		U true false						
	R-> S-> T-> U-> V-> W-> X->	R && S S'== T U < U U + V V * W (Q) Z	S S I= T U <= U U - V V / W -W id X' ε	U >= U V V % W X	w							
	R-> S-> T-> U-> V-> W-> X-> Y-> Z->	R && S S'== T U < U U + V V * W [W (Q) Z Z, Q	S S I= T U <= U U - V V / W -W id X' ε	U >= U V V % W X	w							
	R-> S-> T-> U-> V-> W-> X->	R && S S'== T U < U U + V V * W [W (Q) Z Z, Q	S S I= T U <= U U - V V / W -W id X' ε	U >= U V V % W X	w							
	R-> S-> T-> U-> V-> W-> X-> Y-> Z->	R && S S'== T U < U U + V V * W (Q) Z	S S!= T U <= U U - V V/W -W id X'	U >= U V V % W X	w							
	R-> S-> T-> U-> V-> W-> X-> Y-> Z->	R && S S'== T U < U U + V V * W [W (Q) Z Z, Q	S S I= T U <= U U - V V / W -W id X' ε	U >= U V V % W X	w							
	R -> S -> T -> U -> V -> W -> X -> Z -> AA ->	R && S S'== T U < U U + V V * W !W (Q) Z Z,Q AA [Q]	S SI=T U<=U U-V V/W -W id X' ε Q [Q]	U>= U V V % W X numero	W cadena	true false						
Notemos que las produccion	R-> S-> T-> U-> V-> W-> X-> Y-> Z->	R && S S'== T U < U U + V V * W !W (Q) Z Z,Q AA [Q]	S SI=T U<=U U-V V/W -W id X' ε Q [Q]	U>= U V V % W X numero	W cadena	true false						
Notemos que las produccion Gramática:	R -> S -> T -> U -> V -> W -> X -> Z -> AA ->	R && S S'== T U < U U + V V * W (Q) Z Z, Q AA [Q] soor la izquierda, por lo que al quitar la	S SI=T U<=U U-V V/W -W id X' ε Q [Q]	U>= U V V % W X numero	W cadena	true false						
Notemos que las produccion Gramática:	R -> S -> T -> U -> V -> W -> X -> Y -> AA -> mes F, K, Q, R, S, U, V, Zy AA son recursivas p	R && S S'== T U < U U + V V * W (Q) Z Z, Q AA [Q] soor la izquierda, por lo que al quitar la	S SI=T U<=U U-V V/W -W id X' ε Q [Q]	U>= U V V % W X numero	W cadena	true false						
Notemos que las produccion Gramática:	R -> S -> T -> U -> V -> W -> X -> Y -> AA -> A ->	R && S S == T U < U U + V V * W W (Q) Z Z,Q AA (Q) Door la izquierda, por lo que al quitar l.	S SI=T U<=U U-V V/W -W id X' € Q [Q] [Q]	U>= U V V % W X numero	W cadena	true false						
Notemos que las produccion Gramática:	R -> S -> T -> U -> V -> W -> X -> Y -> AA -> A -> B ->	R && S S'== T U < U U + V V + W (Q) Z Z,Q AA [Q] oor la izquierda, por lo que al quitar la B G C E; B	S SI=T U<=U U-V V/W -W id X' ε Q [Q]	U>= U V V % W X numero	W cadena	true false						
Notemos que las produccion Gramática:	R -> S -> T -> U -> V -> W -> X -> Y -> AA -> A -> B -> C ->	R && S S'== T U < U U + V Y * W W (Q) 2 Z, Q AA Q] oor la izquierda, por lo que al quitar l. B G C E; B D E	S S Is T U cold	U >= U V V % W X numero	W cadena iene lo siguien	true false						
Notemos que las produccion Gramática:	R -> S -> T -> U -> V -> W -> X -> Y -> AA -> A -> B -> C -> D ->	R && S S'== T U < U U + V V * W (Q) Z Z,Q AA [Q] or la izquierda, por lo que al quitar la B G CF;B D E int	S SI=T U<=U U-V V/W -W id X' € Q [Q] [Q]	U>= U V V % W X numero	W cadena	true false						
Notemos que las produccion Gramática:	R -> S -> T -> U -> V -> W -> X -> Y -> AA -> A -> B -> C -> D ->	R && S S'== T U < U U + V V * W (Q) Z Z,Q AA [Q] or la izquierda, por lo que al quitar la B G CF;B D E int	S S Is T U cold	U >= U V V % W X numero	W cadena iene lo siguien	true false						
Notemos que las produccion Gramática:	R -> S -> T -> U -> V -> W -> X -> A -> A -> B -> C -> D -> E ->	R && S S E = T U < U U + V V * W W (Q) Z Z Z Q AA Q AA Q B G E E E E E E E E E	S S I= T U <= U U -= V V / WW id X' c Q [Q] Imisma sobre las producciones mencion	U >= U V V % W X numero	W cadena iene lo siguien	true false						
Notemos que las produccion Gramática:	R -> S -> T -> U -> V -> W -> X -> Y -> AA -> B -> C -> D -> E -> F ->	R && S S == T U < U U + V V * W W (Q) Z Z,Q AA [Q] Door la izquierda, por lo que al quitar la B G CF;B DE int [numero] E id F'	S S t= T U <= U U - V V / W - W id X' ε Q [Q] I misma sobre las producciones mencion ε float ε	U >= U V V % W X numero	W cadena iene lo siguien	true false						
Notemos que las produccion Gramática:	R -> S -> T -> U -> V -> W -> X -> A -> A -> A -> B -> C -> D -> E -> F -> F ->	R &&S S'== T U < U U + V Y * W (Q) Z Z,Q AA(Q) AA(Q) BG CF;B DE int [numero] E id F' ,id F'	S S I= T U <= U U -= V V / WW id X' c Q [Q] Imisma sobre las producciones mencion	U>=U V V V%W X numero	W cadena iene lo siguien	true false						
Notemos que las produccion Gramática:	R -> S -> T -> U -> U -> V -> X -> X -> X -> A -> A -> B -> C -> D -> E -> F -> F -> F -> G ->	R && S S == T U < U U + V V * W W (Q) Z Z,Q AA [Q] Door la izquierda, por lo que al quitar la B G CF;B DE int [numero] E id F'	S S t= T U <= U U - V V / W - W id X' ε Q [Q] I misma sobre las producciones mencion ε float ε	U >= U V V % W X numero	W cadena iene lo siguien	true false						
Notemos que las produccion Gramática:	R -> S -> T -> U -> U -> V -> X -> X -> X -> A -> A -> B -> C -> D -> E -> F -> F -> F -> G ->	R &&S S'== T U < U U + V Y * W (Q) Z Z,Q AA(Q) AA(Q) BG CF;B DE int [numero] E id F' ,id F'	S S Is T U <= U U - V V / W - W id X' ε Q [Q] Imisma sobre las producciones mencion ε float ε	U>=U V V V%W X numero	W cadena iene lo siguien	true false						
Notemos que las produccion Gramática:	R -> S -> T -> U -> U -> V -> W -> X -> Y -> AA -> AA -> B -> C -> D -> E -> F -> F -> F -> H -> H ->	R && S S'== T U < U U + V V* W (Q) Z Z,Q AA (Q) Oor la izquierda, por lo que al quitar la BG CF;B DE int [numero] E id F' ,id F' func Cid(H)] G	$\begin{array}{c} S \\ S \Vdash T \\ U <= U \\ U <= U \\ V \vdash W \\ \vdash W \\ \text{id } X^* \\ & \varepsilon \\ Q \\ Q \\ Q \\ \end{array}$ misma sobre las producciones mencion ε float ε	U>=U V V V%W X numero	W cadena iene lo siguien	true false						
Notemos que las produccion Gramática:	R -> S -> T -> U -> U -> V -> X -> X -> Y -> AA -> A -> B -> C -> D -> E -> F -> F -> G -> H -> I ->	R && S S'== T U < U U + V V*W (Q) Z Z,Q AA (Q) BG CF,B DE int numero) E id F' , id F' func Cid (H) J G C H J C Cid I'	S S Is T U <= U U - V V / W - W id X' ε Q [Q] Imisma sobre las producciones mencion ε float ε	U>=U V V V%W X numero	W cadena iene lo siguien	true false						
Notemos que las produccion Gramática:	R -> S -> T -> U -> V -> W -> X -> Y -> AA -> A -> B -> C -> D -> E -> F -> G -> H -> I	R && S S'== T U < U U + V V * W (Q) Z Z,Q AA [Q] For la izquierda, por lo que al quitar la B G C E; B D E int [numero] E id F' id F' fenc C id [H]] G I C id I' , c id I' , c id I' , c id I'	$\begin{array}{c} S \\ S \Vdash T \\ U <= U \\ U <= U \\ V \vdash W \\ \vdash W \\ \text{id } X^* \\ & \varepsilon \\ Q \\ Q \\ Q \\ \end{array}$ misma sobre las producciones mencion ε float ε	U>=U V V V%W X numero	W cadena iene lo siguien	true false						
Notemos que las produccion Gramática:	R -> S -> T -> U -> U -> V -> W -> X -> Y -> A -> A -> B -> C -> D -> E -> F -> F -> G -> H -> I -> I -> I -> J ->	R && S S'== T U < U U + V V * W NW (Q) Z Z,Q AA Q BG CF,B DE int [numero] E idf' , idf' ffnec (idf) G Cidf' Cidf' (Ekf) [EK]	$\begin{array}{c} S \\ S \Vdash T \\ U <= U \\ U <= U \\ V \vdash W \\ \vdash W \\ \text{id } X^* \\ & \varepsilon \\ Q \\ Q \\ Q \\ \end{array}$ misma sobre las producciones mencion ε float ε	U>=U V V V%W X numero	W cadena iene lo siguien	true false						
Notemos que las produccion Gramática:	R -> S -> T -> U -> U -> V -> W -> X -> Y -> A -> A -> B -> C -> D -> E -> F -> F -> G -> H -> I -> I -> I -> J ->	R && S S'== T U < U U + V V * W NW (Q) Z Z,Q AA Q BG CF,B DE int [numero] E idf' , idf' ffnec (idf) G Cidf' Cidf' (Ekf) [EK]	$\begin{array}{c} S \\ S \Vdash T \\ U <= U \\ U <= U \\ V \vdash W \\ \vdash W \\ \text{id } X^* \\ & \varepsilon \\ Q \\ Q \\ Q \\ \end{array}$ misma sobre las producciones mencion ε float ε	U>=U V V V%W X numero	W cadena iene lo siguien	true false						
Notemos que las produccion Gramática:	R -> S -> T -> U -> U -> V -> W -> X -> Y -> AA -> AA -> A -> B -> C -> D -> E -> F -> F -> F -> F -> I -> I -> I -> I -> I -> K ->	R &&S S'== T U < U U + V V * W (Q) Z Z,Q AA [Q] For la izquierda, por lo que al quitar la BG CF;B DE int [numero] E id F' id F' func Cid (H)] G C dd I' C dd I' C dd I' C ld I'	$\begin{array}{c} S \\ S \Vdash T \\ U <= U \\ U <= U \\ V \vdash W \\ \vdash W \\ \text{id } X^* \\ & \varepsilon \\ Q \\ Q \\ Q \\ \end{array}$ misma sobre las producciones mencion ε float ε	U>=U V V V%W X numero	W cadena iene lo siguien	true false						
Notemos que las produccion Gramática:	R -> S -> T -> U -> U -> V -> W -> X -> Y -> AA -> AA -> B -> B -> C -> D -> E -> F -> G -> H -> I -> I -> I -> I -> K	R && S S'== T U < U U + V V * W NW (Q) Z Z,Q AA Q BG CF,B DE int [numero] E idf' fnnc (idf H) G I Cidf' Cidf' Cidf' (BK) LK' LK'	S S t= T U c= t U c= t U - V V/W -W id X	U > U V V V % W X numero	W cadena iene lo siguien double	true false						
Notemos que las produccion Gramática:	R -> S -> T -> U -> U -> V -> W -> X -> Y -> AA -> AA -> A -> B -> C -> D -> E -> F -> F -> F -> F -> I -> I -> I -> I -> I -> K ->	R && S S'== T U < U U + V V * W (Q) Z Z,Q AA [Q] For la izquierda, por lo que al quitar la BG CF;B DE int [numero] E id F' fuec Cid(H) JG Cid I' (BK) L K' P = Q;	S S !s T U < = U U - V V / W - W id X'	U>=U V V V %W X numero sadas anteriormente se t char	W cadena iene lo siguien double while(Q) L	true false	ı					
Notemos que las produccion Gramática:	R -> S -> T -> U -> U -> V -> W -> X -> Y -> AA -> AA -> B -> B -> C -> D -> E -> F -> G -> H -> I -> I -> I -> I -> K	R && S S'== T U < U U + V V * W (Q) Z Z,Q AA [Q] For la izquierda, por lo que al quitar la BG CF;B DE int [numero] E id F' fuec Cid(H) JG Cid I' (BK) L K' P = Q;	S S !s T U < = U U - V V / W - W id X'	U>=U V V V %W X numero adas anteriormente se t char	W cadena iene lo siguien double while(Q) L	te: void do L while(0) break;						
Notemos que las produccion Gramática:	R -> S -> T -> U -> U -> U -> V -> W -> X -> Y -> Z -> AA -> A -> B -> C -> D -> E -> F -> G -> H -> I -> I -> J -> K	R && S S'= T U < U U + V V * W NW (Q) Z Z,Q AA (Q) BG CF;B DE int [numero] E idF' func Cid(H) JG I Cid I' Cid I' (BK) LK' P = Q; return U;	S S I= T U ← U U − V V / W − W id X € Q [Q] consists a sobre las producciones mencion € float € € float € € if(O L switch(Q) [M]	U >= U V V % W X numero char ff(Q) L else L print U;	W cadena iene lo siguien double	true false						
Notemos que las produccion Gramática:	R -> S -> T -> U -> U -> V -> W -> X -> Y -> Z -> AA -> A -> B -> C -> D -> E -> F -> F -> F -> G -> H -> I -> I -> I -> I -> I -> I -> K	R && S S'== T U < U U + V V * W (Q) Z Z,Q AA [Q] For la izquierda, por lo que al quitar la	S S !s T U < = U U - V V / W - W id X'	U>=U V V V %W X numero adas anteriormente se t char	W cadena iene lo siguien double while(Q) L	te: void do L while(0) break;	J					
Notemos que las produccion Gramática:	R -> S -> T -> U -> U -> V -> W -> X -> Y -> Z -> AA -> B -> C -> D -> E -> F -> L -> L -> M -> N ->	R && S S'== T U < U U + V V * W NW (Q) Z Z,Q AA Q] BG CF;B DE int [numero] E idF' id F' func Cid(H) G I Cid I' Cid I' (Sid I' BK L K' P = Q return U; NM csen numero. K csen numero. K	S S I= T U ← U U − V V / W − W id X € Q [Q] consists a sobre las producciones mencion € float € € float € € if(O L switch(Q) [M]	U >= U V V % W X numero char ff(Q) L else L print U;	W cadena iene lo siguien double while(Q) L	te: void do L while(0) break;	1					
Notemos que las produccion Gramática:	R -> S -> T -> U -> U -> V -> W -> X -> Y -> Z -> AA -> B -> C -> D -> E -> F -> L -> L -> M -> N ->	R && S S'== T U < U U + V V * W (Q) Z Z,Q AA [Q] For la izquierda, por lo que al quitar la	S S I= T U ← U U − V V / W − W id X € Q [Q] consists a sobre las producciones mencion € float € € float € € if(O L switch(Q) [M]	U >= U V V % W X numero char ff(Q) L else L print U;	W cadena iene lo siguien double while(Q) L	te: void do L while(0) break;	ī					
Notemos que las produccion Gramática:	R -> S -> T -> U -> U -> V -> W -> X -> Y -> Z -> AA -> A -> B -> C -> D -> E -> F -> F -> F -> G -> H -> I	R && S S'== T U < U U+V V*W !W (Q) Z Z,Q AA (Q) BG CF;B DE int [numero] E id F' , id F' func Gid (H) G C I (H) G LK' P = Q; return U; NM case numero: K default K	S S L= T U <= U U - V V / W -W ids X	U >= U V V % W X numero char ff(Q) L else L print U;	W cadena iene lo siguien double while(Q) L	te: void do L while(0) break;	I					
Notemos que las produccion Gramática:	R -> S -> T -> U -> U -> V -> W -> X -> Y -> AA -> AA -> A -> B -> C -> D -> E -> F -> F -> F -> F -> F -> L -> A -> B -> C -> D -> D -> E -> F	R && S S'== T U < U U + V V * W NW (Q0) Z Z,Q AA(Q) AA(Q) Door la izquierda, por lo que al quitar lı BG CE;B DE int [numero] E id F' id F' id F' func C id(H) J G I C id I' C id	S S I= T U ← U U − V V / W − W id X € Q [Q] consists a sobre las producciones mencion € float € € float € € if(O L switch(Q) [M]	U >= U V V % W X numero char ff(Q) L else L print U;	W cadena iene lo siguien double while(Q) L	te: void do L while(0) break;	J					
Notemos que las produccion Gramática:	R -> S -> T -> U -> U -> U -> V -> W -> X -> Y -> Z -> AA -> A -> B -> C -> D -> E -> F -> F -> F -> I	R &&S S'==T U < U U+V V*W (Q) Z Z,Q AA (Q) BG CF;B DE int [numero] E id F' , id F' func Gid (H) G C di '' , Cid I' L K' L K' P = Q; return U; NM case numero: K default: K id AA R Q'	S S to T U <= U U <= V V / W - W id X'	U >= U V V % W X numero char ff(Q) L else L print U;	W cadena iene lo siguien double while(Q) L	te: void do L while(0) break;	J					
Notemos que las produccion Gramática:	R -> S -> T -> U -> U -> V -> W -> S -> A -> A -> A -> A -> B -> C -> D -> E -> F -> F -> F -> F -> L -> A -> A -> A -> A -> B -> C -> D -> E -> D -> E -> D -> E -> D -> E	R && S S'== T U < U U + V V * W (Q) Z Z,Q AA(Q) AA(Q) BG C'E',B DE int [numero] E id F' func Cid(H) JG I Cid I' Cid	S S L= T U <= U U - V V / W -W ids X	U >= U V V % W X numero char ff(Q) L else L print U;	W cadena iene lo siguien double while(Q) L	te: void do L while(0) break;						
Notemos que las produccion Gramática:	R -> S -> T -> U -> U -> U -> V -> W -> X -> Y -> Z -> AA -> A -> B -> B -> C -> D -> E -> F -> F -> F -> G -> H -> I	R && S S'== T U < U U+V V*W (Q) Z Z,Q AA Q AA Q BG CF;B DE int (numero) E idf' , idf' (Cid I' Cid I' Cid I' El K' LK' LK' LK' LK' LK' LK' NM Case numero. K default K id AA RQ' S R' S R' S S' S S' S S' S S' S S' S S	S S to T U <= U U <= V V / W - W id X'	U >= U V V % W X numero char ff(Q) L else L print U;	W cadena iene lo siguien double while(Q) L	te: void do L while(0) break;	J					
Notemos que las produccion Gramática:	R -> S -> T -> U -> U -> U -> V -> W -> X -> Y -> Z -> AA -> A -> B -> C -> D -> E -> F -> F -> F -> I	R && S S'== T U < U U+V V*W !W (Q) Z Z,Q AA (Q) AA (Q) BG CF;B DE int [numero] E id F' , id F' func Gid (H) G C id I' (, Cid I' (, E) LK' LK' LK' P = Q; return U; NM case numero: K default: K id AA R Q'	S S to T U <= U U - V V / W - W id X'	U >= U V V % W X numero char ff(Q) L else L print U;	W cadena iene lo siguien double while(Q) L	te: void do L while(0) break;						

## 1													
Total State Stat		S' ->	== T S'	!= T S'	3								
1			U < U	U <= U	U>= U	U > U	U						
Company Comp		Has	VIII	0 1-0	0,20	-70							
V				****									
Yes				- V U'	3								
Windows Section Sect													
Windows State St		V' ->	* W V'	/ W V¹	% W V'	3							
Second Process Column Second Process Second Proce		W->		-w	У	1							
1									1300				
## 1		X ->	(Q)	id AA	numero	cadena	true	false	id(Y)	id			
## 1		Y ->	Z	3									
A			0.7'										
AA			0.01										
Interes, section up on it produce in Lew constraints in finite con		Z>	,QZ	3									
Semicon seminos por la producirio La veneziano di factativo como segundo del producirio P.P.Y., pie la gradi di factativa la p													
Semicon seminos por la producirio La veneziano di factativo como segundo del producirio P.P.Y., pie la gradi di factativa la p		AA' ->	[O] AA [3									
Marie Barrel Ba													
Marie Barrel Ba													
Marie Barrel Ba													
Marie Barrel Ba	nalmente, notemos que er	n la producción L se encuentran dos factores cor	munes por la izquierda, ad	¿más en la producción T,P y X, por lo que al fa	ctorizar, la gramática dei	initiva está defir	nida cómo sigue:						
A-9 BO C C C C C C C C C													
A-9 BO C C C C C C C C C	ramática:												
C	ramatica:												
C		A ->	B G										
C		B ->	CF;B	3									
D													
F ->				float	char	double	void						
F ->		E ->	[numero] E	3									
F ->		F->	id F'							1			
G -> Inter-Child File													
H-2				ε									
1-2		G ->			3								
1-2		H ->	I	3									
1 -			Cid I'							- 1			
1 -		TI ->	Ciari										
K K K K K K K K K K				3									
K K K K K K K K K K		J ->	{ B K }							1			
K -> L K' c white(D) L' white(D) do thile(D) white(D)		K ->	LK'							1			
L-> P-Q		K1 = ~	1 1/1										
Switch(10 Max Switch(10				3	10								
C		L =>	P = Q;	if(Q) L L'	while(Q) L		break;	J					
C			return L''	switch(Q) {M}	print U:	scan P							
U		II-S											
M > O E C C C C C C C C C				3									
N >		L'' ->	U;	j ,									
O-> default K P-> MI P'		M ->	N M	0	3								
O-> default K P-> MI P'		N =>	case numero: K										
P->			default. V										
P'-> AA													
Q-> RQ'													
Q-> RQ'		P' ->	AA	3									
R ->		3-7	KQ										
R'-> S-> TS'		Q' ->	II R Q	3									
S -> T S'		R ->	S R'										
S -> T S'			8-8-5 P1										
S'->		K ->	m ol										
T-> UI' T-> < U		5->											
T'->		S' ->	== T S'	!= T S'	3								
T'->		T ->	U T'										
U-> VU' -VU' -VU' -VU' -VU' -VU' -VU' -VU'		mi_s		e= II	N=11	s II							
U'-> +vU' -vU' c				\ 0	/-0	/ / /	3						
V->													
V -> Wv' /WV' /WV' % WV c		U' ->		- V U'	3								
V' -> *WV' /WV' %WV' ε W -> NV -W X		V =>	W V¹										
W -> fw -W X numero cadena true false				(mark	0/ 77774								
X -> (Q) id X' numero cadena true false X' -> AA (Y) Y -> Z Z -> QZ' AA -> (Q) AA' AA -> (Q) AA' ε (Q) AA'		V· ->				3							
X' -> AA (Y) ε Y -> Z ε Z -> QZ' AA -> (Q)A' ε AA -> (Q)AA' ε (Q)AA' ε (Q)AA' ε		W ->	!w		X								
X' -> AA (Y) ε Y -> Z ε Z -> QZ' AA -> (Q)A' ε AA -> (Q)AA' ε (Q)AA' ε (Q)AA' ε		X ->	(0)	id X'	numero	cadena	true	false					
Y-> Z		V1 =>	AA										
Z-> QZ' Z'-> QZ' ε AA-> [Q]AA' AA'-> [Q]AA' ε													
2'-> Q2'				3									
2'-> Q2'		Z ->	QZ'										
AA -> [Q] AA'		Z' ->		£									
AA'-> [Q]AA'			101 44										
		nn ->	(Q) AA										
of in taking Divinit de Deut Cinetania		AA' ->	(Q) AA'	3									
Similation District de Deut Cinetante													
State of the Printed As Pour Circles of													
Similatión Divisida Day Cintavia													
	ofinición Di-	gida Dor Cintarria											
		a dada antariammenta cala resta adamenta las res			a latera las símbalas na t	amminalas) Dayl							
		a dada ameriormente, solo resta adecuar las regi	gias semanticas a la primer	. gramatica generaua (es decir, la que traduce	a letras los sillibolos lio t	erminales). Por i	io tanto, tenemos	s.					
ándonos en la gramática dada anteriormente, solo resta adecuar las reglas semánticas a la primera gramática generada (es decir, la que traduce a letras los símbolos no terminales). Por lo tanto, tenemos:	sándonos en la gramática												
	sándonos en la gramática												
	sándonos en la gramática												
	sándonos en la gramática												
	ándonos en la gramática			Davis Com									
ándonos en la gramática dada anteriormente, solo resta adecuar las reglas semánticas a la primera gramática generada (es decir, la que traduce a letras los símbolos no terminales). Por lo tanto, tenemos:	sándonos en la gramática	Producción											
ándonos en la gramática dada anteriormente, solo resta adecuar las reglas semánticas a la primera gramática generada (es decir, la que traduce a letras los símbolos no terminales). Por lo tanto, tenemos: Producción Regla Semántica	ándonos en la gramática	Producción											
ándonos en la gramática dada anteriormente, solo resta adecuar las reglas semánticas a la primera gramática generada (es decir, la que traduce a letras los símbolos no terminales). Por lo tanto, tenemos: Producción Regla Semántica	ándonos en la gramática												
ándonos en la gramática dada anteriormente, solo resta adecuar las reglas semánticas a la primera gramática generada (es decir, la que traduce a letras los símbolos no terminales). Por lo tanto, tenemos: Producción Regla Semántica	ándonos en la gramática												
Producción Regla Semántica A - > B G PilaTT push(nuevaTablaTS()) PilaTT push(nuevaTablaTS())	iándonos en la gramática			PilaTS.push(nuevaTablaTS()) PilaTT.push(nuevaTablaTT())									
Producción Producción Regla Semántica A -> B G PilaTS push (mevezTablaTS()) PilaTr push nuevaTablaTT()) dir = 0	iándonos en la gramática	A -> B G		PilaTS.push(nuevaTablaTS()) PilaTT.push(nuevaTablaTT()) lir = 0									
Producción Producción Regla Semántica A ~> B G PilaTS.push(nuevaTablaTS()) dir = 0 B >> CF; B E, tipo = C (tipo)	iándonos en la gramática	A -> B G B -> C F; B		PilaTS.push(nuevaTablaTS()) PilaTT.push(nuevaTablaTT()) lir = 0									
Producción Producción Regla Semántica A - > B G PilaTTS push (nueva TablaTTC)) diri = 0 B -> CF; B Etipo = Ctipo B -> Cepsilon	sándonos en la gramática	A -> B G B -> C F; B		PilaTS.push(nuevaTablaTS()) PilaTT.push(nuevaTablaTT()) lir = 0 E.tipo = C.tipo									
Producción Producción Regla Semántica A - > B G PilaTTS push (nueva TablaTTC)) diri = 0 B -> CF; B Etipo = Ctipo B -> Cepsilon	rándonos en la gramática	A -> B G B -> C F; B B -> epsilon		PilaTS.push(nuevaTablaTS()) PilaTT.push(nuevaTablaTT()) lir = 0 E.tipo = C.tipo									
Producción Producción Regla Semántica A - > B G PilaTTS push (nueva TablaTTC)) diri = 0 B -> CF; B Etipo = Ctipo B -> Cepsilon	sándonos en la gramática	A -> B G B -> C F; B B -> epsilon		PilaTS.push(nuevaTablaTS()) PilaTT.push(nuevaTablaTT()) lir = 0 E.tipo = C.tipo									
Producción Producción Regla Semántica A -> BG PilaTS push (nevezTablaTS()) PilaTT push nuevaTablaTT()) dir = 0 B -> CF; B B -> Ct; po B -> Ct; po C -> DE C (tipo = E, tipo	rándonos en la gramática	A -> B G B -> C F; B B -> epsilon C -> D E											
Producción Regla Semántica A -> B PilaTS, push (nuevaTablaTS()) dir = 0 B -> CF; B E, tipo = C, tipo B -> epsilon C -> DE C, tipo = Lipo C, tipo = Lipo C, tipo = Lipo D, -> int D, ->	sándonos en la gramático	A -> B G B -> C F; B B -> epsilon C -> DE D -> int		PilaTS, push(nuevaTablaTS()) PilaTT: push(nuevaTablaTT()) Iffi = 0 Etipo = C.tipo E.base = D.tipo Litpo = E.tipo Jipo = int									
Producción Regla Semántica A -> B PilaTS, push (nuevaTablaTS()) dir = 0 B -> CF; B E, tipo = C, tipo B -> epsilon C -> DE C, tipo = Lipo C, tipo = Lipo C, tipo = Lipo D, -> int D, ->	sándonos en la gramátic:	A -> B G B -> C F; B B -> epsilon C -> DE D -> int		PilaTS, push(nuevaTablaTS()) PilaTT: push(nuevaTablaTT()) Iffi = 0 Etipo = C.tipo E.base = D.tipo Litpo = E.tipo Jipo = int									
Producción Regla Semántica A -> B G PilaTS push (nueva TablaTS()) pilaTs push (nueva TablaTS()) dit = 0 B -> CF, B B -> CF, B B -> c psilon C -> DE C tipo = E tabe = D tipo D -> int D -> float	sándonos en la gramático	A -> B G B -> C F; B B -> epsilon C -> D E D -> int D -> float		PilaTs.push(nuevaTablaTS()) PilaTT.push(nuevaTablaTT()) fit = 0 Lipo = C.tipo E.base = D.tipo Lipo = Lipo Lipo = Int									
Producción Regla Semántica A -> B G Platr Spush (nuevaTablaTSI)) B -> CF; B Etipo - Ctipo B -> epsilon B -> epsilon D -> int D -> float D -	sándonos en la gramático	A -> B G B -> CF; B B -> epsilon C -> DE D -> int D -> float D -> char		PilaTS.push(nuevaTablaTS()) PilaTT.push(nuevaTablaTT()) Int - 0 S.tipo = C.tipo Lipo = D.tipo Lipo = Lipo Lipo = Into Lipo = Int									
indonos en la gramática dada anteriormente, solo resta adecuar las reglas semánticas a la primera gramática generada (es decir, la que traduce a letras los símbolos no terminales). Por lo tanto, tenemos: Producción Regla Semántica	sándonos en la gramática	A -> B G B -> C F; B B -> epsilon C -> D E D -> int D -> float D -> char D -> double		PilaTS.push(nuevaTablaTS()) PilaTT.push(nuevaTablaTT()) Iti = 0 Lipo = C.Lipo Lipo = D.Lipo Lipo = Lipo Lipo = Lipo Lipo = Into									
A -> B Producción Regla Semántica A -> B PliaTS push (muevaTablaTSI)) B -> CE; B Etipo - Cipo B -> sepsilon C -> DE Cityo - Etipo C -> DE Cityo - Etipo D -> int D jipo - int D -> float D jipo - int D -> float D jipo - float D -> char D -> double D -> oduble D -> void D -> jipo - double D -> void D -> void D -> void D -> void	sándonos en la gramático	A -> B G B -> C F; B B -> epsilon C -> D E D -> int D -> float D -> char D -> double		### ### ##############################									
A -> B Producción Regla Semántica A -> B PliaTS push (muevaTablaTSI)) B -> CE; B Etipo - Cipo B -> sepsilon C -> DE Cityo - Etipo C -> DE Cityo - Etipo D -> int D jipo - int D -> float D jipo - int D -> float D jipo - float D -> char D -> double D -> oduble D -> void D -> jipo - double D -> void D -> void D -> void D -> void	sándonos en la gramática	A -> B G B -> CF; B B -> epsilon C -> D E D -> int D -> float D -> char D -> double D -> void		### ### ##############################	II, ELtipo)								
indonos en la gramática dada anteriormente, solo resta adecuar las reglas semánticas a la primera gramática generada (es decir, la que traduce a letras los símbolos no terminales). Por lo tanto, tenemos: Producción Regla Semántica	sándonos en la gramática	A -> B G B -> CF; B B -> epsilon C -> D E D -> int D -> float D -> char D -> double D -> void		### ### ##############################	I, E.tipo)								

	Fitipo = Fitipo F
	Si! Piarts top() buscar(id) Entonces
F -> id F'	rnatxop(,nisetat(u), xup(, ui, var, vol.0)) dir = dir + FilatThou(, setTam(Chio))
	error ("El id no está declarado") Finsi
	F1.tipo = F1.tipo S1! PilaTS.top().buscar(id) Entonces
	Si!PilaTS.top().buscar(id) Entonces
F' -> , id F'1	FirlarStop().insertari(d, Fitipo, dir, "var", NULO)) dir = dir + PilaTT.top().gerTam(Ctipo)
,	SiNO
	SINO error ("El id no está declarado") Finsi
F' -> epsilon	e lini
r -> epsiioii	ListaRetorno = NULL
	Listaneoutio = vol.2. PilaTS_upin(meeraTablaSimbolos) PilaTT_push(neevaTablaTipos)
	PilaTT.yus/(nuevaTablaTipos) PilaDir.yus/(dir)
	ernatur, pusaquu) dir = 0
	Si! PilarIS.top().buscar(id) Entonces
	St equivalentes.lstal_ListaRetormo, C.tippo Ent District stops Inserts of C.tipo Stops District stops I
	PilaiDir_push(dir) dir = 0 Si [PiD Stop) huscari(d) Entonces Si equivalentes/Lista(Listafetorno, Citipo) Ent PilaiTS.(p() insertatar(d, C. tipo, -, func' Hista) gencod(label(idi))
G -> func C id (H) J G	genCod(label(id)) J.sig = nuevate() genCod(label(isg))
G->tuneCia(H))G	J.Sig = nuewatu() genCod([abel(].sig))
	error("Retomo no coincide") Finsi
	Sino error("ID no declarado")
	error("ID no declarado")
	Finsi Plaffx.pop() Plafft.pop()
	PilaTT.pop()
G -> epsilon	dir = PilaDir,pop()
H->I	H.lista = Llista
H -> epsilon	H lista = MIII I
	Sil PiliaTX.top().buscar(id) Entonces PilaTX.top().insertar(id, Ctipo, dir,
	PilaTS.top().insert_n(af, Ctipo, dir.
	dir = dir + PlaTT.to()_getTam(C.tipo)
I -> C id I'	
	error ("ID no está declarado") Finsi
	I.lista = I'.lista
	Lista agregar (Ctipo)
	Si!PilaTS.top().buscar(id) Entonces PilaTS.top().insertar(id, C.tipo, dir, "param", NULL) dir dir + PilaTT.top().getTam(C.tipo) (#1)
	dir - dir + PilaTT.top(),getTam(C.tipo) Sino
I' -> , C id I'1	Sino error ("ID no está declarado") Finsi
	FinSi
	I lista = 1 L lista
I¹ -> epsilon	l'lista = nuevalistadags()
i -> epsiioii	I'.lista.agregar(C.tipo)
J -> { B K }	K.sig = l.sig genCod(label(K.sig)
	K. sigH = K.sig L.sig = nuevaEtq()
K -> L K'	L.sig = nuevaEtq() genCode(labelLsig)
	L sig = nuevaEtq() gercol(label(Lsg)) [K'ssgif = K'sigif
K' -> L K'1	genCod(label(L.sig))
	R. Lager = S. Auger K. Kisge = Kristing
K' -> epsilon	K'.sig = K'.sigH
	Si equivalentes (Ptipo, Qtipo) Entonces d = reduci(Qdir, Qtipo, Ptipo)
	di = reducti(Qdir, Qtipo, P.tipo) genCod(Pdir = di)
L -> P = Q;	Sino
	error("Tipos incompatibles") Finsi
	Q.vddr = nuevaEtq()
	Q.fls = nuevoIndice()
L -> if(Q) L1 L'	Lisig - Lsig
2 / 11(2) 11 1	L. Sig = L.
	Li_lista_indices.agregar(Qfls)
	genou(aten(yuun)) I sia "I sia"
L' -> else L	genCod(goto Lsig)
n reach	Survivor(unity) Lisje Lisy Lisy genCod('goto' Lisy) genCod((goto' Lisia_indices(o))) remplazarIndices(L'lista_indices, nuevaEtiqueta(), cuadruplas)
L' -> epsilon	remplazarinores (L. insta_ motives, neweartqueed), (cuanripas) remplazarindices (L. insta_ indices, L. insta
	Li sig = nievaEta()
	Q.vddr = msevaEig()
L -> while(Q) L1	Q.vddr = msevaEtq() Q.fls = L.sig gencOd(label(L.sig))
	genCod(label(Q.vddr))
	genCod(goto ' L1.sig)
	Li sig = nuevaEtq() O wdf = nuevaEtq()
L -> do L1 while(Q)	Q.vdd: = nuevaEtq() Q.fds = Lsig genCot(label(L.sig))
	gencod(label(Lisig)) gencod(label(xydr))
L -> break;	gencou(notes)(void)) gencou(not is/g)
L->J	Survivougove maps
L -> return L''	
L'' -> U;	ListaRetorno.agregar(void)
· ·	gencod (return) " ListaRetorno.agregar(void)
L" ->;	Listatecomo.agregativous) genCodi (retum)

	MetqPrueba = nuevaEtq() genCode(goto MetqPrueba) Msigt = Laig
	genCode(goto' M.etqPrueba)
L -> switch(Q) { M }	M.J. = L.SIG M.J. = O. dir
	Mid= 0.dir genode(label(MetqPrueba))
	genCode(M.prueba)
L -> print U;	genCod(print' U.val)
L -> scan P	genCod('scan' P.dir)
	M1.sig = M.sig
M -> N M1	N. sig = M.sig Mpruba Mt.prueba Mpruba Mt.prueba
	Apprecia includes in the preciain of the control of
M -> 0	aspaces - Optices Osig = Maig
M -> epsilon	Mprueba = "
**	Ninicio = nuevaEtq()
N -> case num: K	K.Sig = "Nig" K.prueba = genCod(if N.id" == "num.lexval goto M.inicio) genCode(label(N.inicio))
	goto M.inicio)
	Surviva (university) Onicio = nurvaEq()
O -> default: K	$K \sin z = 0 \sin z$
U -> derault: K	O prueba = genCod (goto O inicio) gencod (labello (unicio))
	genCod(label(0.inicio))
P -> id P'	P'base = idlexval pdi = P'dri P'dri
F => IU F	r.dm = r.dm P.tipo = P'.tipo
	AAbase = P'base
P1 -> AA	P'.dir = AA.dir
	P'.tipo = AA.tipo
	Si PilaTS.top().buscar(P'.base) Entonces
	P'.dir = P'.base P'.tipo = PilaTS.top(),gerTipo(P'.dir)
P' -> epsilon	
	error("El id no está declarado")
	FinSi
	R.vddr = Q.vddr R.fls = musc/indice()
	(N. 100 F - RIDO)
Q -> R Q'	Q'lista_indices = nuevaListatndices()
	Rits = muevoinance() (2 inpoit = Rits = muevoinance() (3 inpoit = Rits = muevoinance() (4 inpoit = Rits = muevoinance() (5 inst_indices = Rits = muevoinance() (5 inst_indices = Rits =
	Q.tips < (!tipoS genCod(label(R.fls))
	Scientification (First Prince)
	Si equivalentes(Q',tipoH, R,tipo) Entonces R,vddr = Q,vddr R,fls = nuevlondice()
	R.fls = nuevoIndice()
	Q1.tipoH = Atipo Q1.wide = Q1.wide
	C_Lingolis = Ringor C_Lingolis = Ringor C_Lingolis = Ringor C_Lingolis = Ringor C_Lingolis = Ringolis = Ringoli
Q' -> R Q'	Q'ilista indices = Q'ilista indices
	Q'ulista_indices.agregar(R.fls) Q'ujeoS = Q'utipoS
	Q.t.poS = Q!1.poS genCod(label(Q.fts))
	Sino
	error("Tipos incompatibles")
	FinSi
Q' -> epsilon	reemplazarIndices(Q'.lista_indices, Q'.fls, cuadruplas) (Q'.tjos = in
	Q. tipos = mt
	S.vddr = nuevoIndice() S.fls = R.fls
	E! 'tipofi = S. tipo F. S. ti
R -> S R'	R'lista_indices = nuevaListaIndices()
	P. llista_indices regreat(S viddr) P time P times
	R.tipo = R'tipos genod(labelis oddr))
	Si equivalentes(R' 170H, Stipo) Entonces Svddr = unworindrice()
	S.vddr = nuevoIndice()
	S. Sis = R'. fis (amque dice R., no R') Pt i visual: a Pt visual (a margina)
	R'.t.ipolt = R'.tipol (o mismo) R'.t.def = book.vider
	Pis fie = bool fie
R' -> && S R'	R'Ilista_indices_apregar(S.vddr)
	R'tinos = 1'tinos
	R'.tipoS -R'.tipoS genCod(label(S.vddr))
	Sino
	error("Tipos incompatibles") Finsi
The continu	reemplazatIndices(R'.lista indices, R'.wddr. cuadruplas)
R' -> epsilon	R'.tipoS = INT
	T.vddr = S.vddr T.fls = S.fls
	T fls = \$ fls 5' vddr = \$ vddr
	S' fis = S fis
S -> T S'	S.1.ipot = T.tipo S.1.dirH = T.dir
	S'.dirH = T.dir
	S.tipo = S'.tipo S.dir = S'.dir
	Suit = S, unt
	1.700 - 3.700 T.fls = S.fls
	Tifle = S'fls S'1wddr = S'wddr
	STEQUIVARIENCES, APPOT, LAUPO EMBRICOS S dir news7mm()
	Si Sequivalentes(\$'14p0H, Titpo) Entonces Si dir = nuevaTemp() tipoTemp = max(\$'14p0H,Titpo) di = amplint(\$'d.trls, \$11p0t, tipottemp)
	d1 = ampliar(S'.dirH, S'.tipoH, tipoTemp)
S' -> == T S'1	di = amplanty, dirri, s. ripori, tipo temp) di = amplanty, dirri, s. ripori, tipo temp) gentCod(S' dir '=' d' dir' ==' d' z. dir) gentCod ('S' dir '=' d' z. dir) gentCod ('S' dir '=' d' z. dir) gentCod (goto 'Tufs) S. tipo = S' tipo S. tipo = S' tipo
	genCod('if' S'.dir' goto' T.vddr)
	genCod('goto' T.fls')
	S'.tipo = S'.tipo S'.tipoH = Tripo
	S'LdirH = T.dir
	Sino
	Sino error("Tipos incompatibles")
	Pinsi

S' -> l= TS'1	Title S'vidit Title S'vidit S'the S'vide S'vide S'the S'vide S'vi
	Sino error("Tipos incompatibles") FinSi S'tipo = S'tipoH
S' -> epsilon	S'.dir = S'.dirH
T->UT	Tipo = Tipo Tdir = T'dir T'ipoH = Utipo T'itipoH = Utipo T'idit = Udir T'vidt = Twidt T'vidt = Twidt
T'->< U	Si equivalentes (T' it) pot, Utipo) Entonces T' tipo : PIN T' dir = nueval'entil (1) tipo (T' entil + T' it) pot, tipo (T' entil + T' it) pot (T' entil +
T'-> <= U	Si equivalentes(T'i)pot. Uipo) Entonces T'itipo = INT T'dir = nuev1remporal() tipo Tempe = maxT'itipoti, Uipo) tipo Uiportemp = maxT'itipoti, Uiportemp) d2 = ampliar(Udir, Udipo, tipo/temp) genCod(T'dir 'e'dind'; c'e'd dir) genCod(T'dir 'e'dind'; c'e'd dir) genCod(T'dir 'e'dind'; c'e'd) semCod(T'dir 'e'dir 'e'd') semCod(T'dir 'e
T'->>≥U	Si equivalentes (T'.tipoH, Utipo) Entonces T'.tipo = INT T'.dir = newTripoH, Utipo) tipoTemp = max(T'.tipoH, Utipo) da = max(T'.tipoH, Utipo) da = max(T'.tipoH, Utipo) da = max(T'.tipoH, Utipo) genCod(T'.dir ' t'.dir', p., to t'.dir) genCod(T'.dir ' t'.dir', p., to t'.dir) genCod(T'.dir ' t'.dir', p., to t'.dir) genCod(T'.dir' t'.dir', p., to t'.dir) genCod(T'.dir' t'.dir') sono error("Tipos incompatibles") FinSI FinSI
T'->> U	Si equivalentes (T'.tipoH, Utipo) Entonces T'.tipo = INT T'.dir = new T'.tipoH, Utipo) Entonces UtipoTeomy = max(T'.tipoH, Utipo) dis = max(T'.tipoH, Utipo) dis = max(T'.tipoH, Utipo) dis = max(T'.tipoH, UtipoH, Utipo) dis = max(T'.tipoH, UtipoH,
T'-> epsilon	T' dip = T' dip H T' dir = T' dir H
U->VU	U.tipo = U.tipo U.dir = U'.dir U'.tipo H = V.tipo U'.tipo H = V.tipo U'.tipt = V.tipo
U' -> + V U'1	Sequivalents(V lipol, V lipo) Entonces Sequivalents(V lipol, V lipo) Entonces U dir v U dir v U dir U tipol = max(V lipol, V lipo) U't dirl = nuex/emporal d = max(V dirl, V lipol, V lipol) sequence sequen
U' -> - V U'1	Stequivalentes(U'tipoH, Vitipo) Entonces
U¹ -> epsilon	U'.tipo = U'.tipoH U'.dir = U'.dirH
V -> W V ^s	Vitpo = V.tipo V.dir = V.dir V.tipo H = W.tipo V.diri = W.dir V.diri = W.dir

V' -> * W V'1	Si equivalentes(V', tipoH, Witpo) Entonces V', tipo = V', tipo V', tipo = V', tipo V', tipo = V', t						
V'->/WV'1	Siequivalentes(V'.tipoH, W.tipo) Entonces V.tipo = V'.tipo = V'.tipo V'.tipo = V'.tipo V'.tipo = V'.tipo V'.tipoH = max(V'.tipoH, W.tipo) V'.titipH = max(V'.tipoH, W.tipo) d' = max(V'.tif, V'.tipoH, V'.tipoH) d' = max(W.tif, V'.tipoH, V'.tipoH) genCodV'.tifh' = 'd' t' /d.zih' SenCodV'.tifh' = 'd' t' /d.zih' Sencod'.Tipo incompatibles") Finsi Finsi						
V' -> % W V'1	Si equivalentes(V',tipoH, Witpo) Entonces V',tipo = V',tipo = V',tipo V',tipo = V',tipo V',tipo = V',tipo V',tipoH = V',t						
V¹ -> epsilon	V'.tipo = V'.tipoH V'.dir = V'.dirH						
W->! W1	W.dir = nuevaTemporal() W.tipo = Wt.tipo genCod(W.dir '= '' ' Wt.dir)						
W->- W1	W.dir = nuevaTemporal() W.tipo = W1.tipo genCod(W.dir '=' '-' W1.dir)						
W -> X	W.dir = X.dir W.tipo = X.tipo						
X -> (Q)	X.dir = Q.dir X.tipo = Q.tipo						
X -> id X'	X' base = id.lexval X.dir = X'.dir X.dir = X'.tipo						
X -> numero	X.dir = numero.lexval X.tipo = numero.lextipo						
X -> cadena	TablaCadenas.agregar(cadena.lexval) X.dir = TablaCadenas.getUltimaPos() X.tipo = cadena						
X -> true	X.dir = 'true' X.tipo = int						
X -> false	X.dir = 'false' X.tipo = int						
X'-> AA	AA.base = X'.base X'.dir = nuevaTemp() X'.tip = AA.tipo genCod(X'.dir '= ' X'.base '['AA.dir']')						
X' -> (Y)	SiPilar's Condo() buscart(") base) Entonces SiPilar's Condo() (getVar(") base) - 'func' Entonces Si equivalentes Listas("pilar's fondo() getArgs(") base), Y.lista) Entonces X' tipe - pilar's To(p) (getTipo(") base) X' tile - nuevaTemp() X' tile - call X' base ', 'Y.lista.tam) Simo error("El número o tipo de parámetros no coincide) Finsi Sino Tipo ("El id no esta declarado")						
X' -> epsilon	X'.dir = X'.base X'.tipo = PilaTS.top.getTipo(X'.dir)						
Y -> Z Y -> epsilon	Y.lista = Z.lista Y.lista = NULO						
Z -> QZ'	Zlista = Zlista Zlista = Zlista Zlista agregar(Q,tipo) genCode(param' Q,dir)						
Z¹->, QZ¹1	Z'.lista = Z'1.lista = Z'1.lis						
Z¹ -> epsilon	Z'.lista = nuevaListaArgs()						
AA -> [Q] AA'	Si PilarSt top() buscar(AA.base) Entonces Si Qtipo = INTE tatonces tipOrmp = PilarStop(), getTipo(Qhase) Si PilarTtop() getVombret(tiporImp) = 'array' Entonces AA' tipo = PilarTtop(), getTiporDase(tiporImp) genCod(AA.dir' = 'Qdir'** AA.tam) AA.dir = AA' tipo AA' tipo = AA' tipos Sino Finot" El idn oe su narreglo") Finsi Sino error("El idnice del arreglo debe ser entero") Finsi Sino ("El id no está declarado")						

AA' -> [Q] AA'1	Si Qtipo = INT Entoness Si PilaTTLO(p) getNombre(AA'.tipo) = 'array' Entoness AA'.tipo = PilaTTLO(p), getTipoBase(AA'.tipo) dirTmp = newa'Temporal() AA'.tig = newa'Temporal() AA'.tig = newa'Temporal() AA'.tig = newa'Temporal() getCod(dirTmp = 'Q da'.*AA'.tipo) getCod(dirTmp = 'Q da'.*AA'.tipo) getCod(AA'.tig = 'z 'AA'.tim'.tipo) AA'.tig = AA'.tim'.tipo Sino cror("El id no es un arreglo") Ensi Sino error("El indice del arreglo debe ser un entero") Finsi							
	AA'.dirS=AA'.dir AA'.tipoS = AA'.tipo							

Esquema de traducción			
A -> { PilaTS.push(nuevaTablaTS()) PilaTT.push(nuevaTablaTT())			
$\operatorname{dir} = 0$			
BG			
B-C Star City			
{ F.tipo = C.tipo } F; B			
B -> epsilon			
C->D			
{ E.base = D.base }			
E { C.tipo = E.tipo }			
D -> int			
{D.tipo = int }			
D -> float			
{D.tipo = float}			
D -> char {D.tipo = char}			
D -> double			
{D.tipo = double}			
D -> void			
{D.tipo = void}			
E -> [numero] { E1.base = E.base }			
E1			
{ E.tipo = PilaTT.top().insertar("array", num.val, E1.tipo) }			
E -> epsilon { E.tipo = E.base }			
{ E.tipo = E.base } F -> id			
{ F'.tipo = F.tipo }			
F' { Si! PilaTS.top().buscar(id) Entonces			
PilaTS.top().insertar(id, F.tipo, dir, "var", NULO))			
dir = dir + PilaTT.top().getTam(C.tipo) SiNo			
SINO error ("El id no está declarado")			
FinSi }			
F' -> , id			
{ F'1.tipo = F'.tipo } F'1			
{ Si! PilaTS.top().buscar(id) Entonces			
PilaTS.top().insertar(id, F'.tipo, dir, "var", NULO))			
dir = dir + PilaTT.top().getTam(C.tipo) SiNo			
error ("El id no está declarado")			
FinSi }			
F'-> epsilon			

G -> func C id (H) { ListaRetorno = NULL PilaTS.push(nuevaTablaSimbolos) PilaTT.push(nuevaTablaTipos) PilaDir.push(dir) dir = 0 Si!PilaTS.top().buscar(id) Entonces Si equivalentesLista(ListaRetorno, C.tipo) Ent PilaTS.top().insertar(id, C.tipo, -, 'func' H.lista) genCod(label(id)) J.sig = nuevaEtq() } JG { genCod(label(J.sig)) Sino error("Retorno no coincide") FinSi				
Sino error("ID no declarado") FinSi PilaTS.pop() PilaTT.pop() dir = PilaDir.pop() }				
G -> epsilon				
H -> I {H.lista = I.lista}				
H -> epsilon {H.lista = NULL}				
I -> C id I' { Si l PilaTS.top().buscar(id) Entonces PilaTS.top().insertar(id, C.tipo, dir, "param",NULL) dir = dir + PilaTT.top().getTam(F.tipo) Sino error ("ID no está declarado")				
FinSi Llista = I'.lista Llista.agregar(C.tipo) }				
I' -> , Cid { Si ! PilaTS.top().buscar(id) Entonces PilaTS.top().insertar(id, C.tipo, dir, "param",NULL) dir = dir + PilaTT.top().getTam(C.tipo) Sino error ("ID no está declarado") FinSi } I'1 { I'.lista = I'.l.lista I.lista.agregar(C.tipo) }				
<pre>I' -> epsilon { I'.lista = nuevaListaArgs()</pre>				
<pre>J -> { B { K.sig = J.sig} K } { genCod(label(K.sig)) }</pre>				
K -> { L.sig = K'.sig } L { K'.sigH = K.sig } K'				
<pre>K' -> { L.sig = K'l.sig } L { K'l.sigH = K'.sigH } K'1 { K'.sig = nuevaEtq() genCod(label(K'.sig)) }</pre>				
K' -> epsilon { K'.sig = K'.sigH }				

L -> P = Q; { Si equivalentes(P.tipo, Q.tipo) Entonces
error("Tipos incompatibles") FinSi} L->if({ Qvddr = nuevaEtq() Qfls = nuevaEtq() } Q) { LL.sig = L.sig } L1 { genCod(label(Qvddr)) } L->if({ Qvddr = nuevaEtq() Qfls = nuevaEtq() Qfls = nuevaEtq() } Q,fls = nuevaEtq() } Q) { LL.sig = L.sig }
L -> if(
<pre>Qfls = nuevaEtq() { Q) {Ll.sig = L.sig } L1 {genCod(label(Q,vddr)) } L -> if({Q,vddr = nuevaEtq() Q,fls = nuevaEtq() } Q) {Ll.sig = L.sig }</pre>
{ L1.sig = L.sig } L1 { genCod(label(Q.vddr)) } L -> if({ Q.vddr = nuevaEtq() } Q.fls = nuevaEtq() } Q) { L1.sig = L.sig }
{ genCod(label(Q.vddr)) } L -> if({ Q.vddr = nuevaEtq() } Q.fls = nuevaEtq() } Q) { Ll.sig = L.sig }
{ Q,vddr = nuevaEtq() Q,ffs = nuevaEtq() } Q) { Ll.sig = L.sig }
{ L1.sig = L.sig }
{ L2.sig = L.sig } L2
{ genCod(label(Q.vddr)) genCod(`goto` L.sig) genCod(label(Q.fls)) }
L -> while(
Q) { L1.sig = nuevaEtq() } L1
{ genCod(label(L1.sig)) genCod('goto` L1.sig) }
L> do { L1.sig = nuevaEtq() }
L1 while({ Q.vddr = nuevaEtq() Q.fls = L.sig }
Q) { genCod(label(L1.sig)) genCod(label(Q.vddr)) }
L -> break; { genCod(goto L.sig) }
L -> { J.sig = L.sig }
L -> return U; { ListaRetorno.agregar(U.tipo) genCod(`return` U.dir) }
L -> return; { ListaRetorno.agregar(void) genCod('return')}
L -> switch(Q) { { M.etqPrueba = nuevaEtq() } genCode(`goto` M.etqPrueba)
M.sig = L.sig M.id = Q.dir }
M } { genCode(label(M.etqPrueba)) genCode(M.prueba) }
L -> print U; { genCod(`print` U.val) }
L -> scan P { genCod(`scan` P.dir) }

	_		
M -> { N.sig = M.sig }			
N { M1.sig = M.sig }			
M1 { M.prueba = M.prueba M1.prueba }			
M -> { O.sig = M.sig }			
O { M.prueba = O.prueba }			
M -> epsilon { M.prueba = " }			
N -> case num: { K.sig = M.sig }			
K			
{ N.inicio = nuevaEtq() N.prueba = genCod(if N.id`==`num.lexval`goto` M.inicio) genCode(label(N.inicio)) }			
O -> default: { K.sig = 0.sig }			
K			
{ 0.inicio = nuevaEtq()			
P -> id P'			
{ P.base = id.lexval P.dir = P'.dir			
P.tipo = P'.tipo }			
P' -> { AA.base = P'.base } AA			
{ P'.dir = AA.dir P'.tipo = AA.tipo }			
P' -> epsilon { Si PilaTS.top().buscar(P'.base) Entonces			
P'.dir = P ['] .base			
P'.tipo = PilaTS.top().getTipo(P'.dir) Sino			
error("El id no está declarado") FinSi }			
Q-> { R.vddr = Q.vddr R.fls = nuevoIndice() }			
R			
{ Q'.tipoH = R.tipo Q'.lista_indices = nuevaListaIndices()			
Q'.lista_indices.agregar(R.fls) }			
{ Q.tipo = Q'.tipoS			
genCod(label(R.fls)) } Q' ->			
{ Si equivalentes(Q'.tipoH, R.tipo) Entonces R.vddr = Q.vddr			
R.fls = nuevoIndice() }			
R { Q'1.tipoH = R.tipo			
{ Q'1.tipoH = R.tipo Q'1.vddr = Q.vddr Q'1.fls = Q.fls			
Q ¹ 1.lista indices = Q ¹ 1.lista indices			
Q ⁱ 1.lista_indices.agregar(R.fls) } Q'1			
{ Q'.tipoS = Q'1.tipoS genCod(label(Q1.fls))			
Sino			
error("Tipos incompatibles") FinSi}			
O' -> epsilon			
{ reemplazarIndices(Q'.lista_indices, Q'.fis, cuadruplas) Q'.tipoS = int }			
× A			

R-> { S.vddr = nuevoIndice() S.fls = R.fls } S			
{ R'.tipoH = S.tipo R'.lista_indices = nuevaListaIndices() R'.lista_indices.agregar(S.vddr) }			
R' { R.tipo = R'.tipoS genCod(label(S.vddr)) }			
R' -> &&			
{ Si equivalentes(R'.tipoH, S.tipo) Entonces S.vddr = nuevoIndice() S.fls = R'.fls (aunque dice R, no R') } S			
{ R'1.tipoH = R'.tipo (lo mismo) R'1.vddr = bool.vddr R'1.fls = bool.fls R'1.lista indices = bool'1.lista indices			
R'1.lista_indices.agregar(S.vddr) } R'1 { R'.tipoS = R'1.tipoS			
genCod(label(s.vddr)) Sino error("Tipos incompatibles")			
FinSi }			
R' -> epsilon { reemplazarIndices(R'.lista_indices, R'.vddr, cuadruplas) R'.tipoS = INT }			
S -> { T.vddr = S.vddr T.fls = S.fls } T			
{ S'.vddr = S.vddr S'.fls = S.fls S'.tipoH = T.tipo			
S'.dirH = T.dir } S' { S.tipo = S'.tipo			
$S.d\hat{r} = S'.d\hat{r}$			
S' -> ==			
{ T.vddr = S'.vddr T.fls = S'.fls } T			
{ S'1.vddr = S'.vddr S'1.fls = S'.fls } S'1			
{ Si equivalentes(S'.tipoH, T.tipo) Entonces S'.dir = nuevaTemp()			
tipoTemp = max(\$\s^1\tipoH,T.tipo) d1 = ampliar(\$'\tipoH, S'\tipoH, tipoTemp) d2 = ampliar(T.dir, T.tipo, tipoTemp) genCod(\$'\tip '=' dt.dir' '==' d2\tipoTemp)			
genCod('if' S'.dir 'goto' T.vddr) genCod('goto' T.fls) S'.tipo = S'1.tipo			
S'1.tîpoH = T.tîpo S'1.dîrH = T.dîr Sino			
error("Tipos incompatibles") FinSi }			

S'->!= { T.vddr = S'.vddr T.fls = S'.fls } T { S'1.vddr = S'.vddr S'!.fls = S'.fls } S'1 { Si equivalentes(S'.tipoH, T.tipo) Entonces S'.dir = nuevaTemp() tipoTemp = max(S'.tipoH,T.tipo) d1 = ampliar(S'.dirH, S'.tipoH, tipoTemp) d2 = ampliar(T.dir, T.tipo, tipoTemp) genCod(S'.dir '=' d.dir '!=' d.dir) genCod('idr 's' d.dir '!=' d.dir) genCod('idr 's'.tipo - T.vddr) genCod('idr 's'.tipo - T.vddr) genCod('idr 's'.tipo - T.vddr) genCod('idr - T.tipo - T.tipo - S'1.tipo - T.tipo - S'1.tipo - T.tipo - S'1.tipo - T.tipo - T.ti				
S' -> epsilon { S'.tipo = S'.tipoH S'.dir = S'.dirH }				
T-> U { T'.tipoH = U.tipo				
T'->< U { Si equivalentes(T'.tipoH, U.tipo) Entonces				
T'-><= U { Si equivalentes(T'.tipoH, U.tipo) Entonces				

[See genotement, 'Lipol', Warpol Entonous 'Very					
Stephywletter(at' type), Wilpo () Vilipol	V'.tipo = V'1.tipo V'.dir = V'1.dir V'1.tipoH = max(V'.tipoH, W.tipo) V'1.dirH = nuevaTemporal() d1 = max(V'.dirH, V'.tipoH, V'1.tipoH) d2 = max(W.dir, W.tipo, V'1.tipoH) genCod(V'1.dirH '=' d1 '*' d2.dir) Sino error("Tipos incompatibles")				
Sequivolentes(Y-tipo), Mutpo) Entonces	{ Si equivalentes(V'.tipoH, W.tipo) Entonces V'.tipo = V'1.tipo V'.dir = V'1.dir V'1.tipoH = max(V'.tipoH, W.tipo) V'1.dirH = nuevaTemporal() d1 = max(V'.dirH, V'.tipoH, V'1.tipoH) d2 = max(W.dir, W.tipo, V'1.tipoH) genCod(V'1.dirH '=' d1 '/' d2.dir) Sino error("Tipos incompatibles") FinSi }				
\{\text{\text{V:lipo} = \text{V:lipo} \} \\ \text{V:lipo} = \text{V:lipo} \} \\ \text{V:lipo} \\ V:lip	{ Si equivalentes(V'.tipoH, W.tipo) Entonces V'.tipo = V'1.tipo V'.dir = V'1.dir V'1.tipoH = INT V'1.dirH = nuevaTemporal() genCod(V'1.dirH '=' V' '%' W.dir) Sino error("Tipos incompatibles")				
{ W.dir = nuevaTemporal() W.tipo genCod(W.dir '=' '' W.tdir)}	{V'.tipo = V'.tipoH				
\{ \ \ \ \ \ \ \ \ \ \ \ \ \ \ \ \ \ \	{ W.dir = nuevaTemporal() W.tipo = W1.tipo				
{ W.dir = X.dir W.tipo = X.tipo } X -> (Q) { X.dir = Q.dir	{ W.dir = nuevaTemporal() W.tipo = W1.tipo				
X.dir = Q,dir	{ W.dir = X.dir				
{ X'.base = id.lexval } X' { X.dir = X'.dir X.tipo = X'.tipo } X - numero { X.dir = numero.lexval X.tipo = numero.lextipo } X - scadena { TablaCadenas.agregar(cadena.lexval) X.dir = TablaCadenas.getUltimaPos() X.tipo = cadena }					
X -> numero {	X' { X.dir = X'.dir				
{ TablaCadenas.agregar(cadena.lexval) X.dir = TablaCadenas.getUltimaPos() X.tipo = cadena }	X -> numero { X.dir = numero.lexval				
Y -s true	X -> cadena { TablaCadenas.agregar(cadena.lexval) X.dir = TablaCadenas.getUltimaPos()				
{ X.dir = 'true' X.tipo = int }					

X -> false { X.dir = 'false'				
X'-> { AA.base = X'.base }				
AA				
{ X'.dir = nuevaTemp()				
X'.tipo = AA.tipo				
genCod(X'.dir + X'.base '['AA.dir']') }				
$X' \rightarrow (Y)$				
{ Si PilaTS.fondo().buscar(X'.base) Entonces				
Si PilaTS.fondo().getVar(X'.base) = 'func' Entonces				
Si equivalentesListas(PilaTS.fondo().getArgs(X'.base), Y.lista) Entonces				
X'.tipo = PilaTS.top().getTipo(X'.base)				
X'.dir = nuevaTemp() genCode(X'.dir '=' 'call X'.base ',' Y.lista.tam)				
Sino				
error("El número o tipo de parámetros no coincide)				
FinSi				
Sino				
error("El id no es una función")				
FinSi				
Sino				
error("El id no está declarado") FinSi }				
X' -> epsilon { X'.dir = X'.base				
X'.tipo = PilaTS.top.getTipo(X'.dir) }				
Y -> Z				
{ Y.lista = Z.lista }				
Y -> epsilon				
{ Y.lista = NULO }				
Z -> Q Z'				
{ Z.lista = Z'.lista				
Z.lista.agregar(Q.tipo)				
genCode('param' Q.dir) }				
$Z' \rightarrow QZ'$				
{ Z'.lista = Z'1.lista				
Z.lista.agregar(Q.tipo) genCode('param' Q.dir) }				
Z' -> epsilon				
{ Z'.lista = nuevaListaArgs() }				
AA -> [Q]				
{ Si PilaTS.top().buscar(AA.base) Entonces				
Si Q.tipo = INT Entonces				
tipoTmp = PilaTS.top().getTipo(Q.base)				
Si PilaTT.top().getNombre(tipoTmp) = 'array' Entonces				
AA'.tipo = PilaTT.top().getTipoBase(tipoTmp) AA'.dir = nuevaTemporal()				
AA .dir = fildeva femporal() AA'.dir = PilaTT.top().getTam(AA'.tipo)				
genCod(AA.dir '=' Q.dir '*' AA.tam) }				
AA'				
{ AA.dir = AA'.dirS				
AA.tipo = AA'.tipoS				
Sino				
error("El id no es un arreglo") FinSi				
Sino				
error("El índice del arreglo debe ser entero")				
FinSi				
Sino				
error("El id no está declarado")				
FinSi }				

AA' -> [Q]			
{ Si Q,tipo = INT Entonces			
Si PilaTT.top(),getNombre(AA'.tipo) = 'array' Entonces			
AA'1.tipo = PilaTT.top().getTipoBase(AA'.tipo)			
dirTmp = nuevaTemporal()			
AA'1.dir = nuevaTemporal()			
AA'1.tam = PilaTT.top().getTam(AA'.tipo)			
genCod(dirTmp '=' 'Q.dir '*' AA'\.tam)			
genCod(AA'ı.dir = '='AA'.dir + dirTmp) }			
AA'1			
AA'.dir = AA'1.dirS			
AA'.tipo = AA'1.tipoS			
Sino			
error("El id no es un arreglo")			
Finst			
Sino			
error("El índice del arreglo debe ser un entero")			
FinSi }			
AA' -> epsilon			
AA'-z epsiloni AA'-dirS=AA'-dir			
{ AA .tipoS = AA' tipo }			
AH .upoo - AH .upo ;			