Definición dirigida por sintaxis

PRODUCCIÓN	**REGLAS SEMÁNTICAS**
$E \rightarrow E_1 + E_2$	$E.val = E_1.val + E_2.val$
$E \rightarrow E_1 * E_2$	$E.val = E_1.val * E_2.val$
$E \rightarrow (E_1)$	$E.val = E_1.val$
$E \rightarrow num$	$E.val = num.val$

3*4+(5*6)+1
12+(30)+1
42+1
43
Un atributo

- El valor de una variable
- La dirección de memoria de una variable
- El tipo de una variable
- Los argumentos de una función (número y tipo)
- La categoría identificador(variable, función, clase, objeto, estructura, etc)
- El tipo retorno de una función
- El valor retorno de una función
- Las etiquetas del código intermedio
- El código intermedio

PRODUCCIÓN	**REGLAS SEMÁNTICAS**
$E \rightarrow E_1 + T$	$E.val = E_1.val + T.val$
$E \rightarrow T$	$E.val = T.val$
$T \rightarrow T_1 * F$	$T.val = T_1.val * F.val$
$T \rightarrow F$	$T.val = F.val$
$F \rightarrow (E)$	$F.val = E.val$
$F \rightarrow num$	$F.val = num.val$

Atributos

- Heredado $S \rightarrow ABC$ $B.h=F(S.a,A.a, C.a)$
- Sintetizados $S \rightarrow ABC$ $S.s = f(A.a, B.a, C.a)$

PRODUCCIÓN	REGLAS SEMÁNTICAS
E → TE’	E’.h = T.val E.val = E’.s
E’ → +TE’$_1$	E’$_1$.h = E’.h + T.val E’.s = E’$_1$.s
E’ →ε	E’.s = E’.h
T → FT’	T’.h = F.val T.val = T’.s
T’ → *FT’$_1$	T’$_1$.h = T’.h *F.val T’.s = T’$_1$.s
T’→ε	T’.s= T’.h
F →(E)	F.val = E.val
F →num	F.val = num.val

A → A$_1$ alfa | beta
A.s = f(A$_1$.s, alfa.s)
A.s =f(beta.s)

A →beta A’
A’.h = f(beta.s)
A.s = f(A’.s)

A’ →alfa A’$_1$
A’$_1$.h = f(A’.h, alfa.s)
A’.s =f(A’$_1$.s)

A’ →
A’.s = f(A’.h)

L → L$_1$, id
L.lista =L$_1$.lista
L.lista.add(id)

L → id
L.lista = nuevaLista()
L.lista.add(id)

x, y , z

L →idL’
L’ →, id L’
L’ →

$L \rightarrow L_1$, id	L.lista =L_1.lista L.lista.add(id)
L → id	L.lista = nuevaLista() L.lista.add(id)

x, y, z [x, y , z]

L →id L'	L'.h =nuevaLista() L'h.add(id) L.lista = L'.s
L' → , id L'_1	L'_1.h = L'.h L'_1.h.add(id) L'.s = L'_1.s
L' →ε	L'.s = L'.h

P → D N
D → T L ; D | ε
T → B A
B → int | float
A → [num] A | ε
L → L, id | id
N → N S | S
S →id = E; | Y = E; | if (E) S | if (E) S else S | while(E) S| do S while(E);
Y → id [E] | Y [E]
E → E + T | T
T → T * F | F
F → id | Y | (E) | num

int x, y;	int[3][2] a, b;
B.base = 0	B.base = 0
A.base = 0	A.base = 0
A.tipo = 0	A.tipo = 0
	A.tipo = 2
	A.tipo = 3

Tabla de símbolos

ID	DIR	TIPO	CATEGORÍA	LISTA ARGS
x	0	0	var	—
y	4	0	var	—
a	8	3	var	—
b	32	3	var	—

Tabla de tipos

ID	NOMBRE	TAMAÑO	#ELEMENTOS	TIPO BASE
0	int	4	—	—
1	float	4	—	—
2	array	8	2	0
3	array	24	3	2

Definición dirigida por sintaxis para análisis sintáctico ascendente

Producción	**Reglas semánticas**
P → D N	
D → T L ; D	L.tipo = T.tipo
D → ε	
T → B A	A.base = B.base T.tipo = A.tipo
B → int	B.base = 0
B → float	B.base = 1
A → [num] A_1	A_1.base = A.base **Si** num.tipo = int **Entonces** A.tipo = insertar(TT, num.val, A_1.tipo) **Sino**

	error(“Se esperaba un entero”) **Fin Si**
A →ε	A.tipo = A.base
L → L_1 , id	L_1.tipo = L.tipo **Si** !existe(TS, id) **Entonces** agregar(TS, id, dir, L.tipo) dir ← dir + getTam(TT, L.tipo) **Sino** error(“La variable ya fue declarada”) **Fin Si**
L → id	**Si** !existe(TS, id) **Entonces** agregar(TS, id, dir, L.tipo) dir ← dir + getTam(TT, L.tipo) **Sino** error(“La variable ya fue declarada”) **Fin Si**
N → N S	
N → S	
S →id = E;	**Si** existe(TS, id.val) **Entonces** tipo = getTipo(TS, id.val) **Si** equivalentes(tipo, E.tipo) **Entonces** dir = reducir(E.dir, E.tipo, tipo) genCod(id ‘=’ dir) **Sino** error(“Tipos incompatibles”) **FinSi** **Sino** error(“La variable no ha sido declarada”) **FinSi**
S → Y = E; int a[3]; a[i] = 5.3;	**Si** equivalentes(Y.tipo, E.tipo) **Entonces** op1 = reducir(E.dir , E.tipo, Y.tipo) genCod(Y.id ‘[‘ Y.dir ‘]’ ‘=’ op1) **Sino** error(“Incompatibilidad de tipos”) **Fin Si**
S → if (E) S_1	ltrue = nuevaEtiqueta() lfalse = nuevaEtiqueta() S.código = genCod(‘if’ E.dir ‘goto’ ltrue) + genCod(‘goto’ lfalse) + genCod(‘label’ ltrue) + S_1.código + genCod(‘label’ false)
S → if(E) S_1 else S_2	ltrue = nuevaEtiqueta() lfalse = nuevaEtiqueta() lfin = nuevaEtiqueta() S.código = genCod(‘if’ E.dir ‘goto’ ltrue)

	+ genCod('goto' lfalse) + genCod('label' ltrue) + S_1.código + genCod('goto' lfin) + genCod('label' false) + S_2.código + genCod('label' lfin)
S → while(E) S_1	linicio = nuevaEtiqueta() ltrue = nuevaEtiqueta() lfalse = nuevaEtiqueta() S.código = genCod('label' linicio) + genCod('if' E.dir 'goto' ltrue) + genCod('goto' lfalse) + genCod('label' ltrue) + S_1.código + genCod('goto' linicio) + genCod('label' lfalse)
S → do S_1 while(E);	ltrue = nuevaEtiqueta() lfalse = nuevaEtiqueta() S.código = genCod('label' ltrue) + S_1.código + genCod('if' E.dir 'goto' ltrue) + genCod('label' lfalse)
Y → id [E]	**Si** existe(TS, id) **Entonces** tipo = getTipo(TS, id) **si** getNombre(TT, tipo) = 'array' **Entonces** Y.tipo = getTipoBase(TT, tipo) Y.tam = getTam(TT, Y.tipo) Y.dir = nuevaTemporal() **Si** E.tipo != int **Entonces** error("El indice para un arreglo debe ser entero") **Fin Si** genCod(Y.dir '=' E.dir '*' Y.tam) Y.id = id **Sino** **Fin Si**
Y → Y_1 [E]	**Si** getNombre(TT, Y_1.tipo) = 'array' **Entonces** Y.tipo = getTipoBase(TT, Y_1.tipo) Y.tam = getTam(TT, Y.tipo) t = nuevaTemporal() Y.dir = nuevaTemporal() **Si** E.tipo != int **Entonces** error("El indice para un arreglo debe ser entero") **Fin Si** genCod(t '=' E.dir '*' Y.tam) genCod(Y.dir '=' Y_1.dir '+' t) Y.id = Y_1.id **Sino**

	Fin Si
$E \rightarrow E_1 + T$	**Si** equivalentes(E_1.tipo, T.tipo) **Entonces** E.dir = nuevaTemporal() E.tipo = máximo(E_1.tipo , T.tipo) op1 = ampliar(E_1.dir , E_1.tipo, E.tipo) op2 = ampliar(T.dir , T.tipo, E.tipo) genCod(E.dir '=' op1 '+' op2) **Sino** error("Incompatibilidad de tipos") **Fin**
$E \rightarrow T$	E.dir = T.dir E.tipo = T.tipo
$T \rightarrow T_1 * F$	**Si** equivalentes(T_1.tipo, F.tipo) **Entonces** T.dir = nuevaTemporal() T.tipo = máximo(T_1.tipo , F.tipo) op1 = ampliar(T_1.dir , T_1.tipo, T.tipo) op2 = ampliar(F.dir , F.tipo, T.tipo) genCod(T.dir '=' op1 '*' op2) **Sino** error("Incompatibilidad de tipos") **Fin**
$T \rightarrow F$	T.dir = F.dir T.tipo = F.tipo
$F \rightarrow (E)$	F.dir = E.dir F.tipo = E.tipo
$F \rightarrow id$	**Si** existe(TS, id) **Entonces** F.dir = id.val F.tipo = getTipo(TS, id) **Sino** error("La variable no fue declarada") **FinSi**
$F \rightarrow num$	F.tipo = num.tipo (lexer) F.dir = num.val (lexer)
$F \rightarrow Y$	F.dir = nuevaTemporal() F.tipo = Y.tipo genCod(F.dir '=' Y.id '[' Y.dir ']')

CÓDIGO INTERMEDIO

- CÓDIGO DE TRES DIRECCIONES
- CÓDIGO DE PILA (CÓDIGO P)

UNA DIRECCIÓN

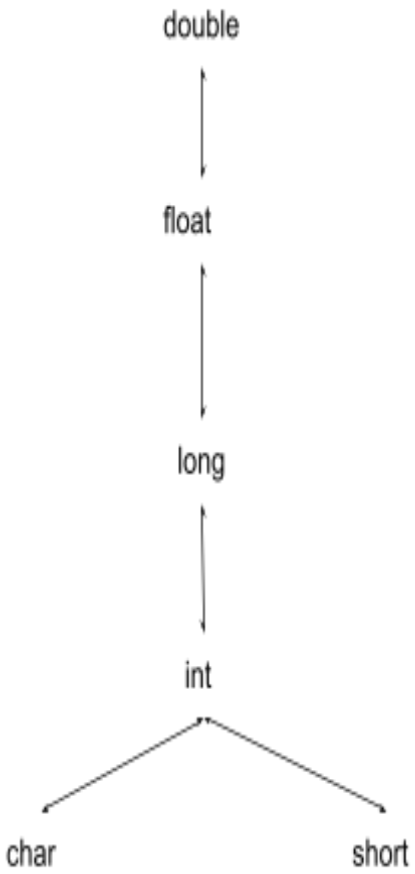
- UNA VARIABLE
- UNA CONSTANTE
- UNA TEMPORAL

INSTRUCCIONES

1. Asignación x= y op z
2. Asignación x = op y
3. Copia x = y
4. Salto condicional
    a. if x goto L
    b. ifFalse x goto L
    c. if x oprel y goto L
    d. ifFalse x oprel y goto L
5. Salto incondicional goto L
6. Copia indexada
    a. x = y[i]
    b. x[i] = y
7. Operaciones con apuntadores
    a. x = &y
    b. x = *y
    c. *x = y

0001 x =0003
0002
0003 10
0004 y = 10

Jerarquía de tipos en un lenguaje de programación



F → Y
tipo = Gtipo
base = Gbase

Producción	**Reglas semánticas**
P → D N	
D → T L ; D	Gtipo= T.tipo
D → ε	
T → B A	Gbase = B.base T.tipo = A.tipo
B → int	B.base = 0
B → float	B.base = 1
A → [num] A_1	**Si** num.tipo = int **Entonces**

	A.tipo = insertar(TT, num.val, A_1.tipo) **Sino** error("Se esperaba un entero") **Fin Si**
A →ε	A.tipo = Gbase
L → L_1 , id	L_1.tipo = L.tipo **Si** !existe(TS, id) **Entonces** agregar(TS, id, dir, L.tipo) dir ← dir + getTam(TT, L.tipo) **Sino** error("La variable ya fue declarada") **Fin Si**
L → id	**Si** !existe(TS, id) **Entonces** agregar(TS, id, dir, L.tipo) dir ← dir + getTam(TT, L.tipo) **Sino** error("La variable ya fue declarada") **Fin Si**
N → N S	
N → S	
S →id = E;	**Si** existe(TS, id.val) **Entonces** tipo = getTipo(TS, id.val) **Si** equivalentes(tipo, E.tipo) **Entonces** dir = reducir(E.dir, E.tipo, tipo) genCod(id '=' dir) **Sino** error("Tipos incompatibles") **FinSi** **Sino** error("La variable no ha sido declarada") **FinSi**
S → Y = E;	**Si** equivalentes(Y.tipo, E.tipo) **Entonces** op1 = reducir(E.dir , E.tipo, Y.tipo) genCod(Y.id '[' Y.dir ']' '=' op1) **Sino** error("Incompatibilidad de tipos") **Fin Si**
S → if (E) S_1	ltrue = nuevaEtiqueta() lfalse = nuevaEtiqueta() S.código = genCod('if' E.dir 'goto' ltrue) + genCod('goto' lfalse) + genCod('label' ltrue) + S_1.código + genCod('label' false)
S → if(E) S_1 else S_2	ltrue = nuevaEtiqueta() lfalse = nuevaEtiqueta()

	lfin = nuevaEtiqueta() S.código = genCod('if' E.dir 'goto' ltrue) + genCod('goto' lfalse) + genCod('label' ltrue) + S_1.código + genCod('goto' lfin) + genCod('label' false) + S_2.código + genCod('label' lfin)
S → while(E) S_1	linicio = nuevaEtiqueta() ltrue = nuevaEtiqueta() lfalse = nuevaEtiqueta() S.código = genCod('label' linicio) + genCod('if' E.dir 'goto' ltrue) + genCod('goto' lfalse) + genCod('label' ltrue) + S_1.código + genCod('goto' linicio) + genCod('label' lfalse)
S → do S_1 while(E);	ltrue = nuevaEtiqueta() lfalse = nuevaEtiqueta() S.código = genCod('label' ltrue) + S_1.código + genCod('if' E.dir 'goto' ltrue) + genCod('label' lfalse)
Y → id [E]	**Si** existe(TS, id) **Entonces** tipo = getTipo(TS, id) **si** getNombre(TT, tipo) = 'array' **Entonces** Y.tipo = getTipoBase(TT, tipo) Y.tam = getTam(TT, Y.tipo) Y.dir = nuevaTemporal() **Si** E.tipo != int **Entonces** error("El indice para un arreglo debe ser entero") **Fin Si** genCod(Y.dir '=' E.dir '*' Y.tam) Y.id = id **Sino** **Fin Si**
Y → Y_1 [E]	**Si** getNombre(TT, Y_1.tipo) = 'array' **Entonces** Y.tipo = getTipoBase(TT, Y_1.tipo) Y.tam = getTam(TT, Y.tipo) t = nuevaTemporal() Y.dir = nuevaTemporal() **Si** E.tipo != int **Entonces** error("El indice para un arreglo debe ser entero") **Fin Si** genCod(t '=' E.dir '*' Y.tam) genCod(Y.dir '=' Y_1.dir '+' t)

	Y.id = Y_1.id **Sino** **Fin Si**
E → E_1 + T	**Si** equivalentes(E_1.tipo, T.tipo) **Entonces** E.dir = nuevaTemporal() E.tipo = máximo(E_1.tipo , T.tipo) op1 = ampliar(E_1.dir , E_1.tipo, E.tipo) op2 = ampliar(T.dir , T.tipo, E.tipo) genCod(E.dir '=' op1 '+' op2) **Sino** error("Incompatibilidad de tipos") **Fin**
E → T	E.dir = T.dir E.tipo = T.tipo
T → T_1 * F	**Si** equivalentes(T_1.tipo, F.tipo) **Entonces** T.dir = nuevaTemporal() T.tipo = máximo(T_1.tipo , F.tipo) op1 = ampliar(T_1.dir , T_1.tipo, T.tipo) op2 = ampliar(F.dir , F.tipo, T.tipo) genCod(T.dir '=' op1 '*' op2) **Sino** error("Incompatibilidad de tipos") **Fin**
T → F	T.dir = F.dir T.tipo = F.tipo
F → (E)	F.dir = E.dir F.tipo = E.tipo
F → id	**Si** existe(TS, id) **Entonces** F.dir = id.val F.tipo = getTipo(TS, id) **Sino** error("La variable no fue declarada") **FinSi**
F → num	F.tipo = num.tipo (lexer) F.dir = num.val (lexer)
F → Y	F.dir = nuevaTemporal() F.tipo = Y.tipo genCod(F.dir '=' Y.id '[' Y.dir ']')

S→ if(E) S_1 H
H→
H→else S

Gramática sin recursividad izquierda

Producción Original	**Producción sin recursividad izquierda**
P → D N	P → D N
D → T L ; D | ε	D → T L D_I D_ I → T L D_I | ε*
T → B A	T → B A
B → int | float	B → int | float
A → [num] A | ε	A → [num] A | ε
L → L, id | id	L → id L_I L_I → , id L_I | ε
N → N S | S	N → S N_I N_I → , S N_I | ε
S →id = E; | Y = E; | if (E) S | if (E) S else S | while(E) S| do S while(E)	S →id = E; | Y = E; | if (E) S | if (E) S else S | while(E) S| do S while(E)
Y → id [E] | Y [E]	Y → id [E] | Y [E]
E → E + T | T	E → + T E_I E_I → +T E_I | ε
T → T * F | F	T → * F T_I T_I → *F T_I | ε
F → id | Y | (E) | num	F → id | Y | (E) | num

Producción	**Reglas semánticas**
P → D N	

D → T L ; D	Ltipo= T.tipo
D → ε	
T → B A	Abase = B.base T.tipo = A.tipo
B → int	B.base = 0
B → float	B.base = 1
A → [num] A_1	**Si** num.tipo = int **Entonces** A.tipo = insertar(TT, num.val, A_1.tipo) **Sino** error("Se esperaba un entero") **Fin Si**
A →ε	A.tipo = Gbase
L → id L_I	L_1_I. tipo = L.tipo **Si** !existe(TS, id) **Entonces** agregar(TS, id, dir, L.tipo) dir ← dir + getTam(TT, L.tipo) **Sino** error("La variable ya fue declarada") **Fin Si**
L_I → , id L_I_1 | ε	L_1_I. tipo = L_I.tipo **Si** !existe(TS, id) **Entonces** agregar(TS, id, dir, L_I.tipo) dir ← dir + getTam(TT, L_I.tipo) **Sino** error("La variable ya fue declarada") **Fin Si**
N → S N_I	
N_I → , S N_I | ε	
S →id S_I	**Si** existe(TS, id.val) **Entonces** S_I.tipo = getTipo(TS, id.val) S_I.dir = id.val **Sino** error("La variable no ha sido declarada") **FinSi**

S_I → Y = E;	Y.id = S_I.dir
	Y.id_tipo = S_I.tipo
	//equivalentes puede simplemente comparar el tipo de ambas
	// expresiones
	Si equivalentes(Y.tipo, E.tipo) **Entonces**
	op1 = reducir(E.dir , E.tipo, Y.tipo)
	genCod(Y.id '[' Y.dir ']' '=' op1)
	Sino
	error("Incompatibilidad de tipos")

	Fin Si
S_I → =E;	**Si** equivalentes(S_I.tipo, E.tipo) **Entonces** dir = reducir(E.dir, E.tipo, S_I.tipo) genCod(S_I.dir '=' dir) **Sino** error("Tipos incompatibles") **FinSi**
S → if (E) S_1	ltrue = nuevaEtiqueta() lfalse = nuevaEtiqueta() S.código = genCod('if' E.dir 'goto' ltrue) + genCod('goto' lfalse) + genCod('label' ltrue) + S_1.código //usen nuevamente su función s() + genCod('label' false)
S → if(E) S_1 else S_2	ltrue = nuevaEtiqueta() lfalse = nuevaEtiqueta() lfin = nuevaEtiqueta() S.código = genCod('if' E.dir 'goto' ltrue) + genCod('goto' lfalse) + genCod('label' ltrue) + S_1.código + genCod('goto' lfin) + genCod('label' false) + S_2.código + genCod('label' lfin)
S → if (E) S_1_I	ltrue = nuevaEtiqueta() lfalse = nuevaEtiqueta() S.código = genCod('if' E.dir 'goto' ltrue) + genCod('goto' lfalse) + genCod('label' ltrue) + S_1_I.código + genCod('label' false)
S_1_I →ε	
S_1_I → else S_2	ltrue = nuevaEtiqueta() lfalse = nuevaEtiqueta() lfin = nuevaEtiqueta() S.código = genCod('if' E.dir 'goto' ltrue) + genCod('goto' lfalse) + genCod('label' ltrue) + S_1.código + genCod('goto' lfin) + genCod('label' false) + S_2.código + genCod('label' lfin)
S → while(E) S_1	linicio = nuevaEtiqueta() ltrue = nuevaEtiqueta() lfalse = nuevaEtiqueta()

	S.código = genCod(‘label’ linicio) + genCod(‘if’ E.dir ‘goto’ ltrue) + genCod(‘goto’ lfalse) + genCod(‘label’ ltrue) + S_1.código + genCod(‘goto’ linicio) + genCod(‘label’ lfalse)
S → do S_1 while(E);	ltrue = nuevaEtiqueta() lfalse = nuevaEtiqueta() S.código = genCod(‘label’ ltrue) + S_1.código + genCod(‘if’ E.dir ‘goto’ ltrue) + genCod(‘label’ lfalse)
Y → id [E]	**Si** existe(TS, id) **Entonces** tipo = getTipo(TS, id) **si** getNombre(TT, tipo) = ‘array’ **Entonces** Y.tipo = getTipoBase(TT, tipo) Y.tam = getTam(TT, Y.tipo) Y.dir = nuevaTemporal() **Si** E.tipo != int **Entonces** error(“El indice para un arreglo debe ser entero”) **Fin Si** genCod(Y.dir ‘=’ E.dir ‘*’ Y.tam) Y.id = id **Sino** **Fin Si**
Y → [E] Y_I	**si** getNombre(TT, Y_.tipo) = ‘array’ **Entonces** Y_I.tipoH = getTipoBase(TT, tipo) Y_I.tamH = getTam(TT, Y.tipo) Y_I.dirH = nuevaTemporal() **Si** E.tipo != int **Entonces** error(“El indice para un arreglo debe ser entero”) **Fin Si** genCod(Y_I.dirH ‘=’ E.dir ‘*’ Y_I.tamH) Y.dir = Y_I.dirS Y.tipo = Y_I.tipoS Y.tam = Y_I.tamS **Sino** **Fin Si**
Y_I → [E] Y_I_1 | ε	**Si** getNombre(TT, Y_I.tipo) = ‘array’ **Entonces** Y_I_1.tipoH = getTipoBase(TT, Y_I.tipo) Y_I_1.tamH = getTam(TT, Y.tipo) t = nuevaTemporal() Y_I_1.dir = nuevaTemporal() **Si** E.tipo != int **Entonces** error(“El indice para un arreglo debe ser entero”) **Fin Si** genCod(t ‘=’ E.dir ‘*’ Y_I_1.tamH) genCod(Y_I_1.dirH ‘=’ Y_I.dirH ‘+’ t) Y_I.dirS = Y_I_1.dirS

	Y_I.tamS = Y_I_1.tamS Y_I.tipoS = Y_I_1.tipoS **Sino** **Fin Si** Y_I.dirS = Y_I.dirH Y_I.tamS = Y_I.tamH Y_I.tipoS = Y_I.tipoH
E → G E_I	E_I.dirh = G.dir E_I.tipoh = G.tipo E.dir = E_I,dirS E.tipo= E_I.tipoS
E_I→ + G E_I_1	**Si** equivalentes(E_I.tipoH, G.tipo) **Entonces** E_I_1.dirH = nuevaTemporal() E_I_1.tipoH = máximo(E_I.tipoH , G.tipo) op1 = ampliar(E_I.dirH , E_I.tipoH, E_I_1.tipoH) op2 = ampliar(G.dir , G.tipo, E_I_1.tipo) genCod(E_I_1.dirH '=' op1 '+' op2) E_I.dirS = E_I_1.dirS E_I.tipoS = E_I_1.tipoS **Sino** error("Incompatibilidad de tipos") **Fin**
E_I→ ε	E_I.dirS = E_I.dirH E_I.tipoS = E_I.tipoH
G → F G_I	G_I.dirh = F.dir G_I.tipoh = F.tipo G.dir = G_I.dirS G.tipo = G_I.tipoS
G_I → * F G_I_1	**Si** equivalentes(G_I.tipoH, F.tipo) **Entonces** G_I_1.dirH = nuevaTemporal() G_I_1.tipoH = máximo(G_I.tipoH , F.tipo) op1 = ampliar(G_I.dirH , G_I.tipoH, G_I_1.tipoH) op2 = ampliar(F.dir , G.tipo, G_I_1.tipo) genCod(G_I_1.dirH '=' op1 '+' op2) G_I.dirS = G_I_1.dirS G_I.tipoS = G_I_1.tipoS **Sino** error("Incompatibilidad de tipos") **Fin**
G_I→ ε	G_I.dirS = G_I.dirH G_I.tipoS = G_I.tipoH
F → (E)	F.dir = E.dir

	F.tipo = E.tipo
F → id	**Si** existe(TS, id) **Entonces** F.dir = id.val F.tipo = getTipo(TS, id) **Sino** error(“La variable no fue declarada”) **FinSi**
F → num	F.tipo = num.tipo (lexer) F.dir = num.val (lexer)
F → id Y	**Si** existe(TS, id) **Entonces** Y.id = id.val Y.id_tipo = getTipo(TS, id) F.dir = nuevaTemporal() F.tipo = Y.tipo genCod(F.dir ‘=‘ Y.id ‘[‘ Y.dir ‘]’) **Sino** error(“La variable no fue declarada”) **FinSi**

Gramática finalizada:

P → D N
D → T L ; D
D → ε
T → B A
B → int
B → float

A → [num] A1
A →ε

L → id L_I
L_I → , id L_I1 | ε

N → S N_I
N_I → , S N_I | ε

S →id S_I
S_I → Y = E;
S_I → =E;

S → if (E) S1_I
S1_I →ε
S1_I → else S2

S → while(E) S1
S → do S1 while(E);

Y → id [E]
Y → [E] Y_I
Y_I → [E] Y_I1 | ε

E → G E_I
E_I→ + G E_I1
E_I→ ε

G → F G_I
G_I → * F G_I1
G_I→ ε

F → (E)
F → id
F → num
F → id Y