Universidad Nacional Autónoma de México
Facultad de Ingeniería
Compiladores, Grupo 3
2017-2
Definición Dirigida por Sintaxis
Proyecto

PRODUCCIÓN	REGLAS SEMÁNTICAS
P → D F	*dir* = 0 *symbolTable* = new SymbolTable() *typeTable* = new TypeTable() pilaSym.push(*symbolTable*) pilaType.push(*typeTable*)
D → T L **;** D	L.tipo = T.tipo L.*dim* = T.*dim*
D → ε	
T → B C	T.*tipo* = C.*tipo* T.*dim* = C.*dimr* C.*base* = B.*tipo* C.*dim* = B.*dim*
T → **struct { D }**	pilaDir.push(*dir*) pilaSym.push(*symbolTable*) pilaType.push(*typeTable*) *dir* = 0 *symbolTable* = new SymbolTable() *typeTable* = new TypeTable() T.*tipo* = pilaSym.getCima() T.*dim* = *dir* *dir* = pilaDir.pop() *symbolTable* = pilaSym.pop() *typeTable* = pilaType.pop()
B → **int**	B.*tipo* = int B.*dim* = 2
B → **float**	B.*tipo* = float B.*dim* = 8
B → **double**	B.*tipo* = double B.*dim* = 16
B → **char**	B.*tipo* = char B.*dim* = 1
B → **void**	B.*tipo* = void B.*dim* = 0
C → [**num**] C_1	C.*tipo* = insertType(C_1.*tipo*, num.*lexval*) C_1.*base* = C.*base*
C → ε	C.*tipo* = C.*base*
L → L_1 **, id**	L_1.*tipo* = L.*tipo* *simbolo* = "*variable*" **if** !*symbolTable*.existe(id) **then** *tipo* = insertType(L.*tipo*, L.*dim*)

PRODUCCIÓN	REGLAS SEMÁNTICAS

	insertSymbol(id.*lexval*, *tipo*, *dir*, *simbolo*)
	dir = *dir* + ancho(*tipo*)
	else if(!pilaSym.getGlobal().existe(id)) **then**
	tipo = insertTypeGlobal(L.*tipo*, L.*dim*)
	insertSymbolGlobal(id.*lexval*, *tipo*, *dir*, *simbolo*)
	dir = *dir* + ancho(*tipo*)
	else
	error("El id ya existe")
	endif

L → **id**	*simbolo* = "*variable*"
	if !*symbolTable*.existe(id) **then**
	tipo = insertType(L.*tipo*, L.*dim*)
	insertSymbol(id.*lexval*, *tipo*, *dir*, *simbolo*)
	dir = *dir* + ancho(*tipo*)
	else if(!pilaSym.getGlobal().existe(id)) **then**
	tipo = insertTypeGlobal(L.*tipo*, L.*dim*)
	insertSymbolGlobal(id.*lexval*, *tipo*, *dir*, *simbolo*)
	dir = *dir* + ancho(*tipo*)
	else
	error("El id ya existe")
	endif

F → **define** T **id(** A **){** D Y **}** F	pilaDir.push(*dir*)
	pilaSym.push(*symbolTable*)
	pilaType.push(*typeTable*)
	Y.*tipo* = "void"
	dir = 0
	symbolTable = new SymbolTable()
	typeTable = new TypeTable()
	simbolo = "*funcion*"
	if(!pilaSym.getGlobal().existe(id)) **then**
	if(T.*tipo* == Y.*tipo*) **then**
	tipo = insertTypeGlobal(L.*tipo*, L.*dim*)
	insertSymbolGlobal(id.*lexval*, *tipo*, -1, *simbolo*, A.*lista*)
	F.*code* = label(id) ‖ Y.*code* ‖ label(L.*next*) ‖ *'halt'*
	else
	error("El tipo de la función no coincide con el tipo retornado")
	endif
	else
	error("El id ya existe")
	endif

F → ε	
A → E	A.*lista* = E.*lista*
A → **void**	A.*lista* = null

E → E $_1$**,** T **id**	**if** !*symbolTable*.existe(id) **then**
	simbolo = "*param*"
	insertSymbol(id.*lexval*, T.*tipo*, *dir*, *simbolo*)
	dir = *dir* + T.*dim*
	else
	error("id duplicado")
	endif

E → T **id**	**if** !*symbolTable*.existe(id) **then** *simbolo* = "*param*"

PRODUCCIÓN	REGLAS SEMÁNTICAS

	insertSymbol(id.*lexval*, T.*tipo*, *dir*, *simbolo*)
	dir = *dir* + T.*dim*
	else
	error("id duplicado")
	endif

Y → Z	Y.*tipo* = Z.*tipo* Y.*code* = Z.*code* ‖ Z.*next*
Z → Z_1 X	Z.*next* = X.*next* Z.*code* = Z_1.*code* ‖ label(Z_1.*next*) ‖ X.*code* ‖ asig(Z.*code*, Z_1.*next*, newLabel())
Z → X	Z.*next* = X.*next* Z.*code* = X.*code*
X → **if**(W) X_1 V	X.*next* = V.*next* X.*code* = W.*code* ‖ label(W.*true*) ‖ X_1.*code* ‖ V.*code* comb(X.*code*, X_1.*next*, V.*next*) comb(X.*code*, W.$false$, V.*next*) asig(X.*code*, W.*true*, newLabel() comb(X.*code*, W.$false$, newLabel()
X → **while(** W **)** X_1	X.*next* = W.$false$ X.*code* = label(X_1.*next*) ‖ W.*code* ‖ label(W.*true*) ‖ X_1.*code* ‖ genCode('goto' X_1.*next*) asig(X.*code*, X_1.*next*,newLabel()) asig(X.*code*, W.*true*,newLabel())
X → **do** X_1 **while(** W **);**	X.*next* = W.$false$ X.*code* = label(W.*true*) ‖ X_1.*code* ‖ label(X_1.*next*) ‖ W.*code* asig(X.*code*, X_1.*next*,newLabel()) asig(X.*code*, W.*true*,newLabel())
X → **for(** X_1**;** W**;** X_1**)** X_3	X.*next* = ".$false$ X.*code* = X_1.*code* ‖ label(X_1.*next*)‖ label(X_3.*next*) ‖ W.*code* ‖ label(W.*true*) ‖ X_3.*code* ‖ X_2.*code* ‖ label(X_1.*next*) asig(X.*code*, X_1.*next*,newLabel()) asig(X.*code*, X_2.*next*,newLabel()) asig(X.*code*, X_3.*next*,newLabel()) asig(X.*code*, W.*true*,newLabel())

X → U **=** G **;**	X.*next* = newLabel()
	if (U.*tipo* = G.*tipo*) **then**
	X.*code* = G.*code* ‖ genCode(U.*dir* '=' G.*dir*)
	else
	error("los tipos no coinciden"
	endif

X → **return** G**;**	X.*next* = newLabel() X.*code* = G.*code* ‖ genCode('return' G.*dir*) X.*tipo* = G.*tipo*

X → **id(** M **);**	**if**(SymbolTableGlobal.existe(id)) **then**
	if(id.simbolo == "funcion") **then**
	if(id.lista == M.lista) **then**
	G.*dir* = id
	G.*tipo* = id.*tipo*
	G.*code* = M.*code* ‖ M.*params* ‖ genCode('call' id ',' id.lista.size)

PRODUCCIÓN	REGLAS SEMÁNTICAS

	else
	error("El número y tipo de parámetros no coinciden")
	endif
	else
	error("El id no es una función ")
	endif
	else
	error("El id no fue declarado")
	endif

X → **{** Z **}**	X.*next* = Z.*next* X.*code* = Z.*code*
V → **else** X	V.*next* = newLabel() V.$false$ = newLabel() V.code = genCode('goto' V.*next*) ‖ label(V.$false$) ‖ X.*code*
V → ε	V.*next* = newLabel() V.$false$ = newLabel()

U → **id**	**if**(symbolTable.existe(id) **or** symTableGlobal.existe(id)) **then**
	U.*tipo* = id.*tipo*
	U.*dir* = id
	U.*code* =""
	else
	error("El id no ha sido declarado ")
	endif

U → S	U.*tipo* = S.*tipo* U.*dir* = S.*dir* U.*code* = S.*code*
U → **id.id**	

S → **id**[G]	**if**(symbolTable.existe(id) **or** symTableGlobal.existe(id)) **then**
	if(id.*tipo* == "array")**then**
	if(G.*tipo* == "int")**then**
	S.*base* = id
	S.*tipo* = TypeTable.getTypeBase(id.*tipo*)
	S.*dim* = TypeTable.getDim(id.*tipo*)
	S.*dir* = newTemp()
	S.*code* = genCon(S.*dir* '=' G.*dir**S.*dim*)
	else
	error("La expresion entre corchetes debe ser tipo entero")
	endif
	else
	error("El id debe ser del tipo array")
	endif
	else
	error("El id no fue declarado")
	endif

S → S_1 [G]	S_1.*base* = S.*base* S.*tipo* = TypeTable.getTypeBase(S_1.*tipo*) **if**(S.*tipo* != -1) **then** **if**(G.*tipo* == "int")**then** S.*dim* = TypeTable.getDim(id.*tipo*) t = newTemp() S.*dir* = newTemp()

PRODUCCIÓN	REGLAS SEMÁNTICAS

	S.*code* = S_1.*code* ‖ genCode(t '=' G.*dir**S.*dim*) ‖
	genCode(S.*dir* '=' S_1.*dir* '+' t)
	else
	error("La expresion entre corchetes debe ser tipo entero")
	endif
	else
	error("El id debe ser del tipo array")
	endif

G → G_1 R G_2	G.*tipo* = max(G_1.*tipo*, G_2.*tipo*) α_1 = amp(G_1.*dir*, G_1.*tipo*, G.*tipo*) α_2 = amp(G_2.*dir*, G_2.*tipo*, G.*tipo*) G.*dir* = newTemp(); G.code = G.*dir* '=' α_1 R.*val* α_2
G → G_1 K G_2	G.*tipo* = max(G_1.*tipo*, G_2.*tipo*) α_1 = amp(G_1.*dir*, G_1.*tipo*, G.*tipo*) α_2 = amp(G_2.*dir*, G_2.*tipo*, G.*tipo*) G.*dir* = newTemp(); G.code = G.*dir* '=' α_1 K.*val* α_2
G → U	G.*tipo* = U.*tipo* G.*dir* = U.*dir* G.*code* = U.*code*
G → **cadena**	G.*dir* = cadena.*lexval* G.*tipo* = 'string' G.*code* = ""
G → **numero**	G.*dir* = numero.*lexval* G.*tipo* = numero.*lexavltipo* G.*code* = ""
G → **caracter**	G.*dir* = caracter.*lexval* G.*tipo* = 'char' G.*code* = ""

G → **id(** M **)**	**if**(SymbolTableGlobal.existe(id)) **then**
	if(id.simbolo == "funcion") **then**
	if(id.lista == M.lista) **then**
	G.*dir* = id
	G.*tipo* = id.*tipo*
	G.*code* = M.*code* ‖ M.*params* ‖ genCode('call' id ',' id.lista.size)
	else
	error("El número y tipo de parámetros no coinciden")
	endif
	else
	error("El id no es una función ")
	endif
	else
	error("El id no fue declarado")
	endif

R → +	R.*val* = +
R → **-**	R.*val* = -
K → *****	K.*val* = *

PRODUCCIÓN	REGLAS SEMÁNTICAS
K → /	K.*val* = /
K → **%**	K.*val* = %
M → M$_1$ **,** G	M.*code* = M$_1$.*code* ‖ G.*code* M.*params* = M$_1$.*params* ‖ genCode('param' G.*dir*) M.lista = M$_1$.*lista*.add(G.*tipo*)
M → G	M.*code* = G.*code* M.*params* = genCode('param' G.*dir*) M.lista = newLista.add(G.*tipo*)
W → W$_1$ **||** W$_2$	W.*true* = W$_2$.*true* W.*false* = W$_2$.*false* W.*code* = W$_1$.*code* ‖ label(W$_1$.*false*) ‖ W$_2$.*code* comb(W.*code*, W$_1$.*true*, W$_2$.*true*) asig(W.*code*, W$_1$.*false*, newLabel())
W → W$_1$ **&&** W$_2$	W.*true* = W$_2$.*true* W.*false* = W$_1$.*false* W.*code* = W$_1$.*code* ‖ label(W$_1$.*true*) ‖ W$_2$.*code* comb(W.*code*, W$_2$.*false*, W$_1$.*false*) asig(W.*code*, W$_1$.*true*, newLabel())
W → **!** W$_1$	W.*true* = W$_1$.*false* W.*false* = W$_1$.*true* W.*code* = W$_1$.*code*
W → G$_1$ **relop** G$_2$	W.*true* = newIndex() W.*false* = new Index() W.*code* = G$_1$.*code* ‖ G$_2$.*code* ‖ genCode('if' G$_1$.*dir* relop.*lexval* G$_2$.*dir* 'goto' W.*true*) ‖ genCode('goto' W.*false*)
W → **true**	W.*true* = newIndex() W.*false* = new Index() W.*code* = genCode('goto' W.*true*)
W → **false**	W.*true* = newIndex() W.*false* = new Index() W.*code* = genCode('goto' W.*false*)

Símbolo	Nombre	Símbolo	Nombre
P	programa	D	declacion
T	Tipo	B	tipo base
C	tipo arreglo	L	Lista identificadores
F	definición de funciones	A	lista definición parámetros
E	lista de parámetros	Y	cuerpo función
Z	sentencias	X	sentencia
V	sentencia else	U	identificadores
S	identificador de arreglo	G	expresión
R	operador de adición	K	operador de multiplicación
M	lista de argumentos	W	expresión booleana