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 → declaraciones **\n** funciones
2. declaraciones → tipo lista_var **\n** declaraciones
   | tipo_registro lista_var **\n** declaraciones
   | ε
3. tipo_registro → **registro \n inicio** declaraciones **\n fin**
4. tipo → base tipo_arreglo
5. base → **ent** | **real** | **dreal** | **car** | **sin**
6. tipo_arreglo → **[num]** tipo_arreglo | ε
7. lista_var → lista_var **,** **id** | **id**
8. funciones → **func** tipo **id(** argumentos **)** **inicio \n** declaraciones sentencias **\n fin \n** funciones | ε
9. argumentos → listar_arg | **sin**
10. lista_arg → lista_arg arg | arg
11. arg → tipo_arg **id**
12. tipo_arg → base param_arr
13. param_arr → [ ] param_arr | ε
14. sentencias → sentencias**\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 >= relacional | relacional == relacional | relacional <> relacional | expresion
18. expresion → expresion + expresion | expresion **-** expresion
    | expresion ***** expresion | expresion / expresion
    | expresion % expresion | **(**expresion **)**
    | variable | **num** | **cadena** | **caracter** | **id(** parametros **)**
19. variable → **id** arreglo | **id.id**
20. arreglo → **id** [expresion] arreglo | ε
21. parametros → lista_param | ε
22. lista_param → lista_param **,** expresion | expresion

PRODUCCIÓN	REGLAS SEMÁNTICAS
programa → declaraciones **\n** funciones	dir = 0 StackTT = newStackTT() StackTS = newStackTS() ts = newSymTab() tt = newTypeTab() StackTT.push(tt) StackTS.push(ts) TablaDeCadenas = newTablaCadenas()
declaraciones → tipo lista_var **\n** declaraciones	type = tipo.tipo
declaraciones → tipo_registro lista_var **\n** declaraciones	type = tipo_registro.tipo
declaraciones → ε	
tipo_registro → **registro \n** **inicio** declaraciones **\n fin**	ts = newSymTab() tt = newTypeTab() StackDir.push(dir) dir = 0 StackTT.push(tt) StackTS.push(ts) dir = StackDir.pop() tt1 = StackTT.pop() StackTS.getCima().setTT(tt1) ts1 = StackTS.pop() dir = StackDir.pop() type = StackTT.getCima().addTipo("registro",0, ts1)
tipo → base tipo_arreglo	base = base.tipo tipo.tipo = tipo_arreglo.tipo
base → **ent**	base.tipo = ent
base → **real**	base.tipo = real
base → **dreal**	base.tipo = dreal
base → **car**	base.tipo = car
base → **sin**	base.tipo = sin
tipo_arreglo → [**num**] tipo_arreglo$_1$	**si** *num.tipo = ent* ***y*** *num.val > 0* **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") **fin**
tipo_arreglo → ε	tipo_arreglo.tipo = base
lista_var → lista_var **, 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**
lista_var → **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**

PRODUCCIÓN	REGLAS SEMÁNTICAS

funciones → **func** tipo **id(** argumentos **) inicio \n** declaraciones sentencias **\n** **fin \n** funciones	**si** *StackTS.getFondo().getId(id.lexval) ≠ -1* **entonces**
	  $StackTS.getFondo().addSym(id.lexval, tipo, --, "func")$
	  $StackDir.push(dir)$
	  $FuncType = tipo.tipo$
	  $FuncReturn = false$
	  $dir = 0$
	  $StackTT.push(tt)$
	  $StackTS.push(ts)$
	  $dir = StackDir.pop()$
	  $add\_quad(code, 'label', -, -, id.lexval)$
	  $L = newLabel()$
	  $backpatch(code, sentencias.next, L)$
	  $add\_quad(code, 'label', -, -, L)$
	  $StackTT.pop()$
	  $StackTS.pop()$
	  $dir = StackDir.pop()$
	  $StackTS.getCima().addArgs(id.lexval, argumentos.lista)$
	  si (tipo.tipo ≠ sin) **y** (FuncReturn = false) **entonces**
	  Error(la función no tiene valor de retorno)
	  endif
	en otro caso
	  Error("El identificador ya fue declarado")
	fin

funciones → ε	
argumentos → listar_arg	argumentos.lista = lista_arg.lista
argumentos → **sin**	argumentos.lista = nulo
lista_arg → lista_arg arg_1	lista_arg.lista = $lista\_arg_1$.lista lista_arg.lista.add(arg.tipo)
lista_arg → arg	lista_arg.lista = newListaParam() lista_arg.lista.add(arg.tipo)

arg → 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 → base param_arr	base = base.tipo tipo_arg.tipo = param_arr.tipo
param_arr → [ ] $param\_arr_1$	param_arr.tipo = StackTT.getCima().addTipo("array", -, $param\_arr_1$.tipo)
param_arr → ε	param_arr.tipo = base
sentencias → sentencias**\n** sentencia	L = newLabel() backpatch(code, sentecias.listnext, L) sentencias.listnext = sentencia.listnext
sentencias → sentencia	sentencias.listnext = sentencia.listnext
sentencia → **si** expresion_booleana **entonces \n** sentencias **\n fin**	L = newLabel() backpatch(code, expresion_booleana.listtrue, L) sentencia.listnext = combinar(expresion_booleana.listfalse, sentencias.listnext)
sentencia → **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)

PRODUCCIÓN	REGLAS SEMÁNTICAS
sentencia → **mientras** \n expresion_booleana **hacer** \n sentencias \n **fin**	L = newLabel() L1 = newLabel() backpatch(code, sentencias.listnext, L) backpatch(code, expresion_boolean.listtrue, L1) sentencia.listnext = expresion_booleana.listfalse add_quad(code, "goto", -, -, L)
sentencia → **hacer** \n sentencias$_2$ \n **mientras que** expresion_booleana	L = newLabel() backpatch(code, expresion_boolean.listtrue, L) backpatch(code, sentencias.listnext, L1) sentencia.listnext = expresion_booleana.listfalse add_quad(code, "label", -, -, L)
sentencia → **id :=** expresion	**si** *StackTS.getCima().getId(id.lexval)* ≠ *-1* **entonces** $t = StackTS.getCima().getTipo(id.lexval)$ $d = StackTS.getCima().getDir(id.lexval)$ $\alpha = reducir(expresion.dir, expresion.tipo, t)$ $add\_quad(code, " = ", \alpha, -, "Id" + d)$ **en otro caso** Error("El identificador no ha sido declarado") **fin** sentecia.listnext= nulo
sentencia → variable **:=** expresion	α = reducir(expresion.dir, expresion.tipo, variable.tipo) $add\_quad(code, " = ", \alpha, -, variable.base[variable.dir])$ sentecia.listnext= nulo
sentencia → **escribir** expresion	add_quad(code, "print", expresion.dir, -,-) sentecia.listnext= nulo
sentencia → **leer** variable	add_quad(code, "scan", -, -,variable.dir) sentecia.listnext= nulo
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	**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.add(I)
expresion_booleana → expresion_booleana$_1$ **oo** expresion_booleana$_2$	L = newLabel() 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 add_quad(code, "label", -, -, L)

PRODUCCIÓN	REGLAS SEMÁNTICAS
expresion_booleana → expresion_booleana$_1$ **yy** expresion_booleana$_2$	L = newLabel() backpatch(code, expresion_booleana$_1$.listtrue, L) expresion_booleana.listtrue =expresion_booleana$_2$.listtrue expresion_booleana.listfalse =combinar(expresion_booleana$_1$.listfalse, expresion_booleana$_2$.lisstfalse) add_quad(code, "label", -, -, L)
expresion_booleana → **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() expresion_booleana.listtrue = newList() expresion_booleana.listtrue.add(I) add_quad(code, "goto", -, -, I) expresion_booleana.listfalse = nulo
expresion_booleana → **falso**	I = newIndex() expresion_booleana.listtrue = nulo expresion_booleana.listfalse = newList() expresion_booleana.listfalse.add(I) add_quad(code, "goto", -, -, I)
relacional → relacional$_1$ < relacional$_2$	relacional.listtrue = newList() relacional.listfalse = newList() I= newIndex(), I1 = newIndex() relacional.listtrue.add(I) relacional.listfalse.add(I1) relacional.tipo = max(relacional1.tipo, relacional$_2$.tipo) α_1 = ampliar(relacional$_1$.dir, relacional$_1$.tipo, relacional.tipo) α_2 = ampliar(relacional$_2$.dir, relacional$_2$.tipo, relacional.tipo) add_quad(code, "<",α_1 ,α_2, I) add_quad(code, "goto", -, -, I1)
relacional → relacional$_1$ > relacional$_2$	relacional.listtrue = newList() relacional.listfalse = newList() I= newIndex(), I1 = newIndex() relacional.listtrue.add(I) relacional.listfalse.add(I1) relacional.tipo = max(relacional1.tipo, relacional$_2$.tipo) α_1 = ampliar(relacional$_1$.dir, relacional$_1$.tipo, relacional.tipo) α_2 = ampliar(relacional$_2$.dir, relacional$_2$.tipo, relacional.tipo) add_quad(code, ">",α_1 ,α_2, I) add_quad(code, "goto", -, -, I1)
relacional → relacional$_1$ <= relacional$_2$	relacional.listtrue = newList() relacional.listfalse = newList() I= newIndex(), I1 = newIndex() relacional.listtrue.add(I) relacional.listfalse.add(I1) relacional.tipo = max(relacional1.tipo, relacional$_2$.tipo) α_1 = ampliar(relacional$_1$.dir, relacional$_1$.tipo, relacional.tipo) α_2 = ampliar(relacional$_2$.dir, relacional$_2$.tipo, relacional.tipo) add_quad(code, "<=",α_1 ,α_2, I) add_quad(code, "goto", -, -, I1)

PRODUCCIÓN	REGLAS SEMÁNTICAS
relacional → relacional$_1$ >= relacional$_2$	relacional.listtrue = newList() relacional.listfalse = newList() I= newIndex(), I1 = newIndex() relacional.listtrue.add(I) relacional.listfalse.add(I1) relacional.tipo = max(relacional1.tipo, relacional$_2$.tipo) α_1 = ampliar(relacional$_1$.dir, relacional$_1$.tipo, relacional.tipo) α_2 = ampliar(relacional$_2$.dir, relacional$_2$.tipo, relacional.tipo) add_quad(code, ">=",α_1 ,α_2, I) add_quad(code, "goto", -, -, I1)
relacional → relacional$_1$ == relacional$_2$	relacional.listtrue = newList() relacional.listfalse = newList() I= newIndex(), I1 = newIndex() relacional.listtrue.add(I) relacional.listfalse.add(I1) relacional.tipo = max(relacional1.tipo, relacional$_2$.tipo) α_1 = ampliar(relacional$_1$.dir, relacional$_1$.tipo, relacional.tipo) α_2 = ampliar(relacional$_2$.dir, relacional$_2$.tipo, relacional.tipo) add_quad(code, "==",α_1 ,α_2, I) add_quad(code, "goto", -, -, I1)
relacional → relacional$_1$ <> relacional$_2$	relacional.listtrue = newList() relacional.listfalse = newList() I= newIndex(), I1 = newIndex() relacional.listtrue.add(I) relacional.listfalse.add(I1) relacional.tipo = max(relacional1.tipo, relacional$_2$.tipo) α_1 = ampliar(relacional$_1$.dir, relacional$_1$.tipo, relacional.tipo) α_2 = ampliar(relacional$_2$.dir, relacional$_2$.tipo, relacional.tipo) add_quad(code, "<>",α_1 ,α_2, I) add_quad(code, "goto", -, -, I1)
relacional → expresion	relacional.tipo = expresion.tipo relacional.dir = expresion.dir
expresion → expresion$_1$ + expresion$_2$	expresion.tipo = max(expresion1.tipo, expresion$_2$.tipo) expresion.dir = newTemp() α_1 = ampliar(expresion$_1$.dir, expresion$_1$.tipo, expresion.tipo) α_2 = ampliar(expresion$_2$.dir, expresion$_2$.tipo, expresion.tipo) add_quad(code, "+",α_1 ,α_2, expresion.dir)
expresion → expresion$_1$ − expresion$_2$	expresion.tipo = max(expresion1.tipo, expresion$_2$.tipo) expresion.dir = newTemp() α_1 = ampliar(expresion$_1$.dir, expresion$_1$.tipo, expresion.tipo) α_2 = ampliar(expresion$_2$.dir, expresion$_2$.tipo, expresion.tipo) add_quad(code, "−",α_1 ,α_2, expresion.dir)
expresion → expresion$_1$ * expresion$_2$	expresion.tipo = max(expresion1.tipo, expresion$_2$.tipo) expresion.dir = newTemp() α_1 = ampliar(expresion$_1$.dir, expresion$_1$.tipo, expresion.tipo) α_2 = ampliar(expresion$_2$.dir, expresion$_2$.tipo, expresion.tipo) add_quad(code, "*",α_1 ,α_2, expresion.dir)
expresion → expresion$_1$ / expresion$_2$	expresion.tipo = max(expresion1.tipo, expresion$_2$.tipo) expresion.dir = newTemp() α_1 = ampliar(expresion$_1$.dir, expresion$_1$.tipo, expresion.tipo) α_2 = ampliar(expresion$_2$.dir, expresion$_2$.tipo, expresion.tipo) add_quad(code, "/",α_1 ,α_2, expresion.dir)

PRODUCCIÓN	REGLAS SEMÁNTICAS
expresion → expresion$_1$ % expresion$_2$	expresion.tipo = max(expresion1.tipo, expresion$_2$.tipo) expresion.dir = newTemp() α_1 = ampliar(expresion$_1$.dir, expresion$_1$.tipo, expresion.tipo) α_2 = ampliar(expresion$_2$.dir, expresion$_2$.tipo, expresion.tipo) add_quad(code, "%",α_1 ,α_2, expresion.dir)
expresion → (expresion$_1$)	expresion.dir = expresion$_1$.dir expresion.tipo = expresion$_1$.tipo
expresion → variable	epxreson.dir = newTemp() expresion.tipo = variable.tipo add_quad(code, "$*$", variable.base[variable.dir] , -, expresion.dir)
expresion → **num**	expresoin.tipo = num.tipo expresion.dir = num.val
expresion → **cadena**	expresion.tipo = cadena expresion.dir = TablaDeCadenas.add(cadena)
expresion → **caracter**	expresion.tipo = caracter expresion.dir = TablaDeCadenas.add(caracter)

expresion → **id(**parametros **)**	**si** *StackTS.getFondo().getId(id.lexval)* $\neq$ *-1* **entonces**
	  si *StackTS.getFondo().getVar(id.lexval) = "func"* **entonces**
	  lista = StackTs.getFondo().getArgs(id.lexval)
	  si *lista.getTam()* $\neq$ *parametros.getTam()* **entonces**
	  Error(El numero de argumentos no coincide)
	  fin
	  para *i=0, i < parametros.lista.getTam(), 1* **hacer**
	  si *parametros[i]* $\neq$ *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 → arreglo	variable.dir = arreglo.dir variable.base = arreglo.base variable.tipo = arreglo.tipo

variable → **id**$_1$**.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 identificador no ha sido declarado")
	fin

PRODUCCIÓN	REGLAS SEMÁNTICAS

arreglo → **id**[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.tam = StackTT.getCima().getTipo(arreglo.tipo)
	  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

arreglo → arreglo_1 [expresion]	**si** *StackTT.getCima().getTipoBase(*arreglo_1*.tipo)="array"* **entonces**
	  si *expresion.tipo = ent* **entonces**
	  arreglo.base = arreglo_1.base
	  arreglo.tipo = StackTT.getCima().getTipoBase(arreglo_1.tipo)
	  arreglo.tam = StackTT.getCima().getTipo(arreglo.tipo)
	  temp = newTemp()
	  arreglo.dir = newTemp()
	  add_quad(code, "*", expersion.dir, arreglo.tam, temp)
	  add_quad(code, "+", arreglo_1.dir, temp, arreglo.dir)
	  en otro caso
	  Error(La expresión para un índice debe ser de tipo entero)
	  fin
	fin
	en otro caso
	Error(El arreglo no tiene tantas dimensiones)
	fin

parametros → lista_param	parametros.lista = lista_param.lista
parametros → ε	parametros.lista = nulo
lista_param → 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 → expresion	lista_param.lista = newListaParam() lista_param.lista.add(expresion.tipo) add_quad(code, "param", expresion.dir, -, -)